Anglia Ruskin University

Distributed Programming

Part 2

SID: 1526338

Date: 26-06-2020

Faculty: Science and Technology

Department: Computing & Technology

Module Code: MOD006128

Academic Year: 2019/20

Trimester: 2

Description

Part 2 was developed utilizing the existent structure created in Part 1. An application was developed that simulates a race between two cars on a racetrack. The two cars are controlled from the keyboard. Blue car is controlled using w-a-s-d and orange car using the arrows. Cars are moving with constant speed unless further input is provided by the user. The racetrack is created using an outside rectangle defining the outside boundaries, along with a second rectangle defining the inside boundaries. Collision detection between the racetrack boundaries and the cars is present. When a car collides with the racetrack boundaries the car's speed is multiplied by 0.1.

Back

A Collision Detector class was created and added to the Back folder. Collision is checked by checking whether the boundary rectangles contain the car location.

The car class was also altered to assist with the new functionality. Direction change coefficients are being calculated on both x and y axis. The movement of the car is performed using those two variables. Multiplication by 0.1 is performed to normalize the speed of the car.

```
this.xPosition += speed * this.xDirChange * 0.1;
this.yPosition += speed * this.yDirChange * 0.1;
```

The Update manager class was also modified to accommodate for the existence of the second car. Timer interval was changed from 50 to 10, to make the animation smoother.

Front

Changes were made on the front side code too. MyFrame was renamed to RaceFrame. Keypress event was created to allowing control of the cars via the keyboard. MyPanel was renamed to RaceTrack. Racetrack draws the racetrack on the paintComponent method.

User Guide

To run the software on Windows, follow the steps bellow:

- Start a command prompt and set the working directory to the directory of the Part1 directory. That can be performed by opening the Part1 directory of the project and typing cmd on the address bar.
- Type java Main.java to start the animation.

To play the game:

Orange Car Controls:
W – Increase Speed
S – Decrease Speed
A – Turn Left
D – Turn Right

Blue Car
↑ (Up Arrow) – Increase Speed
↓ (Down Arrow) – Decrease Speed
← (Left Arrow) – Turn Left
→ (Right Arrow) – Turn Right

Source Code

Main.java

```
import Back.Car;
import Back.CollisionDetector;
import Back.UpdateManager;
import java.awt.*;

public class Main {

    //Entry point
    public static void main(String[] args)
    {
        //Set up collider to match inner and outer box of the racetrack.
        CollisionDetector racetrackCollider = new CollisionDetector(new Rectangle(30,80,740,510)
, (new Rectangle(120,170,560,310)));
        //Initiate manager instance and the two cars.
        UpdateManager instance = new UpdateManager(new Car("orange",racetrackCollider), new Car(
"blue",racetrackCollider));
    }
}
```

Back/Car.java

```
package Back;

import javax.swing.*;
import java.util.ArrayList;
import java.util.List;

public class Car extends JLabel
{
    List<Icon> iconList;
    float speed = 0;
    int direction = 0;
    boolean directionChanged = false;
    int xDirChange = 0;
    int yDirChange = 0;
    double xPosition = 0;
    double yPosition = 0;
    CollisionDetector collisionDetector;

    //Constructor
    public Car (String colour, CollisionDetector collisionDetector)
    {
        //Create the list of Icons for the car
        this.iconList = getIconList(colour);
        //Set the initial car icon
        this.setIcon(iconList.get(0));
        this.setSize(this.getPreferredSize());
        this.collisionDetector = collisionDetector;
    }

    //Set the location of the car using coordinates x and y
    public void locationSet (int x, int y)
```

```
    {
        this.setLocation(x, y);
        this.xPosition = x;
        this.yPosition = y;
    }

    //Increase car speed. Upper limit set to 100. Step set to 5.
    public void increaseSpeed ()
    {
        speed += 5;
        if (speed > 100)
        {
            speed = 100;
        }
    }

    //Decrease car speed. Lower limit set to -100. Step set to 5.
    public void decreaseSpeed ()
    {
        speed -= 5;
        if (speed < -100)
        {
            speed = -100;
        }
    }

    //Turns the car left by adding 1 to the direction variable
    public void turnLeft()
    {
        direction += 1;
        //There are 16 direction icons, therefore when reaching the end of the list we need to r
eset the counter to go to the first icon
        if (direction > 15)
```

```
        {
            direction = 0;
        }
        directionChanged = true;
    }

    public void turnRight()
    {
        direction -= 1;
        if (direction < 0)
        {
            direction =15;
        }
        directionChanged = true;
    }

    //Calculate the anglia x and y coefficients based on the direction variable. The result is u
sed for moving the car.
    public void calculateAngles()
    {
        switch (direction)
        {
            case 0:
            xDirChange =4;
            yDirChange =0;
            break;

            case 1:
            xDirChange =3;
            yDirChange =-1;
            break;

            case 2:
```

```
xDirChange =2;
yDirChange =-2;
break;

case 3:
xDirChange =1;
yDirChange =-3;
break;

case 4:
xDirChange =0;
yDirChange =-4;
break;

case 5:
xDirChange = -1;
yDirChange =-3;
break;

case 6:
xDirChange =-2;
yDirChange =-2;
break;

case 7:
xDirChange =-3;
yDirChange =-1;
break;

case 8:
xDirChange =-4;
yDirChange =0;
break;
```

```
case 9:
xDirChange =-3;
yDirChange =1;
break;

case 10:
xDirChange =-2;
yDirChange =2;
break;

case 11:
xDirChange =-1;
yDirChange =3;
break;

case 12:
xDirChange =0;
yDirChange =4;
break;

case 13:
xDirChange =1;
yDirChange =3;
break;

case 14:
xDirChange =2;
yDirChange =2;
break;

case 15:
xDirChange =3;
```

```
            yDirChange =1;
            break;
        }
    }

    //Update car by checking if the direction was changed and setting the correct icon.
    //Move the car to the updated position using the x and y cooefficients of the cars direction
.
    //Apply collision detector speed modifications.
    public void Update()
    {
        if (directionChanged = true)
        {
            this.setIcon(iconList.get(direction));
            directionChanged = false;
        }
        calculateAngles();
        this.xPosition += speed * this.xDirChange * 0.1;
        this.yPosition += speed * this.yDirChange * 0.1;
        this.setLocation((int)this.xPosition,(int)this.yPosition);
        speed = speed*collisionDetector.detectColl(xPosition, yPosition, speed);
    }

    //Creates the icon list by accessing directory.
    public List<Icon> getIconList(String colour)
    {
        List<Icon> iconList = new ArrayList<>();
        for (int i = 1 ; i <= 16 ; i++)
        {
            //Relative path. Application must be run from the directory where main.java is locat
ed.
            iconList.add(new ImageIcon(String.format("%s/Resources/%s_%d.png",System.getProperty
("user.dir"),colour,i)));
```

```
        }
        return iconList;
    }
}
```

Back/CollisionDetector.java

```
package Back;

import java.awt.*;

public class CollisionDetector
{
    Rectangle outerBound;
    Rectangle innerBound;

    //Constructor. Takes two rectangles as arguements (inner and outer racetrack limits)
    public CollisionDetector(Rectangle outerBound, Rectangle innerBound)
    {
        this.outerBound = outerBound;
        this.innerBound = innerBound;
    }

    //Detects collisions of the car with the racetrack. If the two collide, speed is set to 10%
    public float detectColl (double x, double y, float speed)
    {
        if (outerBound.contains(x, y) && !innerBound.contains(x, y))
        {
            return  1f;
        }
        return 0.1f;
    }
}
```

Back/UpdateManager.java

```
package Back;
import Front.RaceFrame;
import Front.RaceTrack;
import javax.swing.*;

public class UpdateManager {
    Timer timer;
    Car car1;
    Car car2;

    public UpdateManager(Car car1, Car car2){
        this.car1 = car1;
        this.car2 = car2;
        car1.locationSet(375, 500);
        car2.locationSet(375,550);
        //Initiate timer and set delay between updates
        timer = new Timer(10, e-> Update());
        timer.start();
        //Set up panel and frame
        RaceTrack raceTrack = new RaceTrack(car1, car2);
        RaceFrame frame = new RaceFrame(raceTrack, car1, car2);
    }

    //Update method is run using the timer. Turns the car left and then updates.
    private void Update(){
        this.car1.Update();
        this.car2.Update();
    }
}
```

Front/RaceFrame.java

```
package Front;
import Back.Car;

import javax.swing.*;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import static java.awt.event.KeyEvent.VK_UP;

public class RaceFrame extends JFrame implements KeyListener {

    Car car1;
    Car car2;

    //Constructor. Sets the frame and adds the cars.
    public RaceFrame (RaceTrack raceTrack, Car car1, Car car2){

        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(850, 650);
        this.setVisible(true);
        this.setLocationRelativeTo(null);
        this.add(raceTrack);

        this.car1 = car1;
        this.car2 = car2;
        this.addKeyListener(this);
    }



    @Override
```

```
public void keyTyped(KeyEvent e) {

}

@Override
public void keyPressed(KeyEvent e) {
    switch (e.getKeyCode()){
        case KeyEvent.VK_UP:
            car1.increaseSpeed();
            break;
        case KeyEvent.VK_DOWN:
            car1.decreaseSpeed();
            break;
        case KeyEvent.VK_LEFT:
            car1.turnLeft();
            break;
        case KeyEvent.VK_RIGHT:
            car1.turnRight();
            break;
        case KeyEvent.VK_W:
            car2.increaseSpeed();
            break;
        case KeyEvent.VK_S:
            car2.decreaseSpeed();
            break;
        case KeyEvent.VK_A:
            car2.turnLeft();
            break;
        case KeyEvent.VK_D:
            car2.turnRight();
            break;
    }
}
```

```
    @Override
    public void keyReleased(KeyEvent e) {
    }
}
```

Front/RaceTrack.java

```
package Front;

import Back.Car;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class RaceTrack  extends JPanel implements ActionListener {

    //Constructor. Set up Panel and car locations.
    public RaceTrack(Car car1, Car car2){
        this.setLayout(null);
        car1.locationSet(375, 500);
        car2.locationSet(375, 550);
        this.add(car1);
        this.add(car2);
    }

    @Override
    public void paintComponent (Graphics g)
    {
        //Paint the racetrack.
        super.paintComponent(g);
        Color c1 = Color.green;
        g.setColor(c1);
        g.fillRect(150, 200, 550, 300);
        Color c2 = Color.black;
        g.setColor(c2);
        g.drawRect(50, 100, 750, 500); // outer edge
```

```
        g.drawRect(150, 200, 550, 300); //inner edge
        Color c3 = Color.yellow;
        g.setColor(c3);
        g.drawRect(100, 150, 650, 400); // mid-laner marker
        Color c4 = Color.white;
        g.setColor(c4);g.drawLine(425, 500, 425, 600); // start Line
    }

    @Override
    public void actionPerformed(ActionEvent e)
    {
        System.out.println(e.getActionCommand());
    }
}
```

Resources