Software Requirements Specification

for

Relay Simulator

Prepared by음료수가 없어요
Angel Echevarria & Preethi Vaidyanathan

16.35 Real-Time Systems and Software

Approved By:

Signature:

Contents

Revision History

Name	**Date**	**Reason For Changes**	**Version**
Angel Echevarria and Preethi Vaidyanathan	5/10/15	Updated constructors and methods, removed real time component	2

1. Introduction

1.1 System overview

We will be creating an interactive, strategy-based simulation that simulates a 100-meter relay race. We will simulate two teams which each have four runners, and the teams will be competing against each other to cross the finish line first. The runners are spaced separately at 25 meter increments. Between consecutive runners (i.e. every 25 meters), there is a "hand off" , where both runners are moving, and then the baton is handed off to the next runner. The runners will have to keep track of each other's positioning and the baton so that they do not run into each other during the race, and are able to surpass each other if necessary. They must also ensure than only one runner per team has possession of.

1.2 Document overview

This document outlines the behavior of the Relay Simulation program.

1.3 Intended Audience

Our intended audience will be Professor Shah and the other students who are taking the class 16.35.

2. Requirements

2.1 Required states and modes

Identify states and modes (and if there are no state and modes required): There will be three states:

1. Runner speed specification state: The initial state that opens up on program launch. This is the state where the speed for each runner, a user-defined input. This will all be done via the command line.
2. Race state: As soon as the user exits the runner selection state by finalizing their selections, the program automatically enters the race state. This is where the program will carry out the relay race in real time based on the user inputs.
3. Winner state: After one team meets the criteria for winning a race, the winner state is entered, where the command line displays the winning team.

2.2 Capability requirements

1. **Simulator class**
    - 1.1. Variables
        - 1.1.1. team1y
            - 1.1.1.1. The Simulator class shall contain a specification for the y-coordinate on which the runners on team number 1 shall initialize
        - 1.1.2. team2y
            - 1.1.2.1. The Simulator class shall contain a specification for the y-coordinate on which the runners on team number 2 shall initialize
        - 1.1.3. runner1x
            - 1.1.3.1. The Simulator class shall contain a specification for the x-coordinate on which the runners who will run the first leg shall initialize
        - 1.1.4. runner2x
            - 1.1.4.1. The Simulator class shall contain a specification for the x-coordinate on which the runners who will run the second leg shall initialize
        - 1.1.5. runner3x
            - 1.1.5.1. The Simulator class shall contain a specification for the x-coordinate on which the runners who will run the third leg shall initialize
        - 1.1.6. runner4x
            - 1.1.6.1. The Simulator class shall contain a specification for the x-coordinate on which the runners who will run the fourth leg shall initialize
    - 1.2. Constructor
        - 1.2.1. the constructor shall not take any arguments
    - 1.3. Methods
        - 1.3.1.1. The Simulator shall create and add a total of eight Runner objects
        - 1.3.1.2. The Simulator shall create and add a total of eight RunnerController objects
2. **Runner class**
    - 2.1. Variables
        - 2.1.1. x
            - 2.1.1.1. The Runner shall contain a representation of the x-directional velocity of the runner
        - 2.1.2. y
            - 2.1.2.1. The Runner shall contain a representation of the y-directional velocity of the runner
        - 2.1.3. pose
            - 2.1.3.1. The Runner shall contain an array (pose) that represents the x-position (x) and the y-position (y) of the runner
        - 2.1.4. input_speed

2.1.4.1. The Runner shall contain representation of the speed of the runner during the race

2.1.4.1.1. Constraints

2.1.4.1.1.1. s shall be in the range [0,1]

2.1.4.1.1.2. The total speed added between a team shall not exceed 2

2.1.5. hasBaton

2.1.5.1. The Runner shall contain a Boolean which specifies whether the runner has the baton or not

2.1.6. start_x

2.1.6.1. The Runner shall contain a representation of the initial x-coordinate of the Runner

2.1.7. teamID

2.1.7.1. The Runner shall contain a representation of the team of the Runner

2.1.7.1.1. Contraints

2.1.7.1.1.1. The teamID shall be 0 and 1 respectively for the two teams

2.1.8. legID

2.1.8.1. The Runner shall contain a representation of the relay leg of the Runner

2.1.8.1.1. Constraints

2.1.8.1.1.1. The legID shall be 0, 1, 2, and 3 for each respective runner on each relay team

2.1.9. justRan

2.1.9.1. The Runner shall contain a Boolean which specifies whether the runner has already completed its relay leg

2.1.10. justPassed

2.1.10.1. The Runner shall contain a Boolean which specifies whether the runner is passing a competing runner

2.1.11. Won

2.1.11.1.1. The Runner shall contain a Boolean which specifies whether the Runner has crossed the finish line first

2.2. Constructor

2.2.1. The internal representation of pose, inputSpeed, hasBaton, start_x, justRan, teamID, legID, s, justPassed, and Won shall be initialized according to the constructor arguments

2.2.2. The constructor shall take 10 arguments

2.2.2.1. The constructor shall take pose as an argument

2.2.2.2. The constructor shall take inputSpeed as an argument

2.2.2.3. The constructor shall take hasBaton as an argument

2.2.2.4. The constructor shall take start_x as an argument

2.2.2.5. The constructor shall take justRan as an argument

2.2.2.6. The constructor shall take teamID as an argument

2.2.2.7. The constructor shall take legID as an argument

2.2.2.8. The constructor shall take s as an argument

2.2.2.9. The constructor shall take justPassed as an argument

2.2.2.10. The constructor shall take hasWon as an argument

2.2.3. Constraints

2.2.3.1. The constructor shall initialize the input speed according to the range specified in 2.1.4.1.1

2.3. Methods

2.3.1. addSimulator(Simulator sim) shall take in a Simulator as an object and assign a simulator to the Runner object

2.3.2. getVehicleID() method shall take no arguments and return an integer (VehicleID)

2.3.3. getStart_x() method shall take no arguments and return an integer (start_x)

2.3.4. getVelocity() method shall take no arguments and return an array (velocity)

2.3.5. getPosition() method shall take no arguments and return an array (pose)

2.3.6. getApproachTime() method shall take no arguments and return a double (ApproachTime)

2.3.7. controlRunner(control c) method shall take a Control object and use it to modify the internal velocities according to the specified velocity

2.3.8. getInputSpeed() method shall take no arguments and return a double (input_speed)

2.3.9. getHasBaton() method shall take no arguments and return a Boolean (hasBaton)

2.3.10. setHasBaton(Boolean hasBaton) shall take in a Boolean, and set the internal representation of the variable hasBaton

2.3.11. getJustRan() method shall take no arguments and return a boolean (justRan)

2.3.12. setJustRan(Boolean justRan) shall take in a Boolean, and set the internal representation of the variable justRan

2.3.13. getWon () method shall take no arguments and return a boolean (hasWon)

2.3.14. setWon(Boolean hasWon) shall take in a Boolean, and set the internal representation of the variable hasWon

2.3.15. getTeamID() method shall take no arguments and return an int (teamID)

2.3.16. getLegID() method shall take no arguments and return an int (legID)

2.3.17. getJustPassed() method shall take no arguments and return a boolean (justPassed)

2.3.18. setJustPassed(Boolean justPassed) shall take in a Boolean, and set the internal representation of the variable justPassed

3. Control class

3.1. Variables

3.1.1. Speed

3.1.1.1. The Control class shall contain a speed (s) that represents the magnitude of the velocity of the runner

3.1.2. Theta

3.1.2.1. The Control class shall contain a theta (t) that represents the angle of the runner

3.2. Constructor

3.2.1. The internal representation of s shall be initialized according to the constructor arguments.

3.2.2. The constructor shall take one arguments

3.2.2.1. The constructor shall take s as an argument

3.2.3. Constraints

3.2.3.1. s shall be initialized according to the constraints specified in 3.1.1.2.1

3.3. Methods

3.3.1. getSpeed-method shall return s

4. FirstRunnerController class

4.1. Variables

4.1.1. current_runner

4.1.1.1. The FirstRunnerController class contains a representation of the current runner in the relay

4.1.2. next_runner

4.1.2.1. The FirstRunnerController class contains a representation of the next runner in the relay

4.1.3. s

4.1.3.1. The FirstRunnerController class contains a Simulator object

4.2. Constructor

4.2.1. The internal representation of s, current_runner, and next_runner shall be initialized according to the constructor arguments

4.2.2. The constructor shall take three arguments

4.2.2.1. The constructor shall take s as an argument

4.2.2.2. The constructor shall take current_runner as an argument

4.2.2.3. The constructor shall take next_runner as an argument

4.3. Methods

4.3.1. getControl() method shall generate a control according to the time the simulation has been running for

4.3.1.1. The first runner on both teams shall merge in to the middle of the track and then travel in a straight line along the track

4.3.1.2. The first runner shall continue to run until it meets up with the second runner, at which point it shall stop moving, hasBaton shall be set to false and justRan shall be set to true

5. RunnerController class

5.1. RunnerController extends FirstRunnerController

5.2. Variables

5.2.1. prev_runner

5.2.1.1. The RunnerController class contains a representation of the previous runner in the relay

5.2.2. current_runner

5.2.2.1. The RunnerController class contains a representation of the current runner in the relay

5.2.3. next_runner

5.2.3.1. The RunnerController class contains a representation of the next runner in the relay

5.2.4. comp_runner

5.2.4.1. The RunnerController class contains a representation of the competing runner, who is running the same relay leg on the opposite team

5.2.5. s

5.2.5.1. The RunnerController class contains a Simulator object

5.3. Constructor

5.3.1. The internal representation of s, prev_runner, current_runner, next_runner, and comp_runner shall be initialized according to the constructor arguments

5.3.2. The constructor shall take five arguments

5.3.2.1. The constructor shall take s as an argument

5.3.2.2. The constructor shall take prev_runner as an argument

5.3.2.3. The constructor shall take current_runner as an argument

5.3.2.4. The constructor shall take next_runner as an argument

5.3.2.5. The constructor shall take comp_runner as an argument

5.4. Methods

5.4.1. getControl() method shall generate a control according to the time the simulation has been running for

5.4.1.1. When the first runner meets up with the second runner, the second runner shall merge into the track at a point in time that will cause it to coincide with the first runner, at which point hasBaton shall be set to true and justRan shall be set to true

5.4.1.2. The third runner shall repeat step 5.4.1.1 with the second runner

6. LastRunnerController class

6.1. Variables

6.1.1. prev_runner

6.1.1.1. The RunnerController class contains a representation of the previous runner in the relay

6.1.2. current_runner

6.1.2.1. The RunnerController class contains a representation of the current runner in the relay

6.1.3. sec_prev_runner

6.1.3.1. The RunnerController class contains a representation of the runner before the previous runner

6.1.4. comp_runner

6.1.4.1. The RunnerController class contains a representation of the competing runner, who is running the same relay leg on the opposite team

6.1.5. s

6.1.5.1. The RunnerController class contains a Simulator object

6.2. Constructor

6.2.1. The internal representation of s, prev_runner, current_runner, sec_prev_runner, and comp_runner shall be initialized according to the constructor arguments

6.2.2. The constructor shall take five arguments

6.2.2.1. The constructor shall take s as an argument

6.2.2.2. The constructor shall take prev_runner as an argument

6.2.2.3. The constructor shall take current_runner as an argument

6.2.2.4. The constructor shall take sec_prev_runner as an argument

6.2.2.5. The constructor shall take comp_runner as an argument

6.2.3. The RunnerController class shall extend FirstRunnerController

6.3. Methods

6.3.1. getControl() method shall generate a control according to the time the simulation has been running for

6.3.1.1. When the third runner stops running, the fourth runner shall continue until crossing the finish line

2.3 Internal data requirements

We will be running this program on Jamaica VM, a real time Java server.

2.4 Environment requirements

This program will run in the Eclipse integrated development environment

3. Qualification provisions

1. Simulator
    1. Test that the time is properly returned as 0 before the run() is called

    2. Test that the total speed of one speed inputted is not greater than 2
2. Control
    1. Test that speed is within the range [0,1]
    2. Test that theta is 45 deg, 0 deg, or -45 deg
3. FirstRunnerController
    1. Test that the constructor properly takes in s, current_runner, and next_runner
4. RunnerController
    1. Test that the constructor properly takes in s, prev_runner, current_runner, next_runner, and comp_runner
5. LastRunnerController
    1. Test that the constructor properly takes in s, current_runner, prev_runner, sec_prev_runner, and comp_runner
6. Runner
    1. Test that the constructor properly takes in pose, inputSpeed, hasBaton, start_x, justRan, teamID, legID, s, justPassed, and hasWon as arguments
    2. Test that hasBaton is not true for more than one Runner at the same time on one team
    3. Test that a teamID value is either 0 or 1
    4. Test that a teamID value that is not 0 or 1 throws an IllegalArgumentException
    5. Test that two teams with equivalent IDs throws an IllegalArgumentException
    6. Test that a legID value is an integer value in the range [1,4]
    7. Test that a LegID value outside of the range [1,4] throws an error
    8. Test that a legID value that is a non-integer value in the range [1,4] throws an IllegalArgumentException
    9. Test that multiple runners with equivalent legID values throws an IllegalArgumentException
    10. Test that hasWon is set to true when a team wins
    11. Test that input_speed is within the range [0,1]