Software Requirements Specification

for

Final Projectile

Version: 1.1

Prepared by Caitlin Wheatley
Syler Wagner

16.35 Real-Time Systems and Software

Approved By:

<date approved>

Signature:

Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This system shall fulfill the final project requirement for 16.35 by providing an interactive, networked system.

1.2 Document overview

This document shall outline the system requirements of Final Projectile, an interactive, multi-player game, including individual classes and interface requirements.

1.3 Intended Audience

This system shall be intended for the use of the Spring 2015 16.35 class and Professor Julie Shah.

1.4 Project Scope

This document applies to all software necessary to run the FinalProjectile game, which encompasses the classes outlined in section 3.

1.5 References

This SRS contains no references to websites or other documents.

2. Overall Description

2.1 Product Features

This system includes all classes necessary for the FinalProjectile game. This includes, but is not limited to, vehicle classes, controllers, and projectiles. Refer to section 3 for specific classes, features, and characteristics.

2.2 User Classes and Characteristics

Refer to section 3 for specific classes, features, and characteristics.

2.3 Operating Environment

The system shall be designed to operate on a personal computer. Players shall have the capability of logging into the game from a remote server. Refer to section 4 for details on interfaces.

2.4 Design and Implementation

Refer to section 3 for specific classes, features, and characteristics.

2.5 User Documentation

All requirements and documentation shall be contained in this document.

2.6 Assumptions and Dependencies

The design of this system shall assume a pre-existing Display and networking capabilities. The running of this system shall be assumed to occur on a platform with the appropriate version of Java installed and running.

3. System Features and Requirements

3.1 Ground Vehicle

???

3.2 Control

???

3.3 Simulator

3.3.1 Variables

3.3.1.1 The Simulator shall contain an internal representation of the system time. This representation shall consist of an integer containing the time in seconds and an integer containing the remainder in milliseconds.

3.3.1.2 The Simulator shall contain a list _vehicleList of the GroundVehicles with which it is associated.

3.3.1.3 The Simulator shall contain a list _followerList of the FollowingControllers with which it is associated.

3.3.1.4 The Simulator shall contain a list _leadingList of the LeadingControllers with which it is associated.

3.3.1.5 The Simulator shall contain internal representations of the x and y dimensions of the simulation size, SIM_X and SIM_Y

3.3.1.6 The values of SIM_X and SIM_Y shall be constant

3.3.1.7 The values of SIM_X and SIM_Y shall be initialized according to the display dimensions set in the DisplayServer

3.3.1.7.1 The x dimension of the simulation shall be initialized to SIM_X = DisplayServer.DISPLAY_X / 5

3.3.1.7.2 The y dimension of the simulation shall be initialized to SIM_Y = DisplayServer.DISPLAY_Y / 5

3.3.1.8 The Simulator shall contain a reference to the UserController _uc object with which it is associated

3.3.1.9 The Simulator shall contain a list _projectileList of Projectiles with which it is associated.

3.3.1.10 The Simulator shall contain a reference to the DisplayClient _dc with which it is associated.

3.3.2 Constructor(Simulator)

3.3.2.1 The internal representation of the DisplayClient shall be initialized according to constructor arguments.

3.3.2.2 The constructor shall take arguments of DisplayClient _dc

3.3.2.2.1 In case of invalid arguments, the constructor shall throw and IllegalArgumentException.

3.3.3 Methods

3.3.3.1 getVehicle()

3.3.3.1.1 This method shall return the internal representation of GroundVehicle _v.

3.3.3.2 getDisplayClient()

3.3.3.2.1 This method shall return the internal representation of the DisplayClient _dc.

3.3.3.3 addVehicle(GroundVehicle v)

3.3.3.3.1 This method shall add the GroundVehicle v to the internal list of GroundVehicles _vehicleList.

3.3.3.3.1.1 In case of invalid arguments, this method shall throw an IllegalArgumentException.

3.3.3.4 addUserController(UserController uc)

3.3.3.4.1 This method shall initialize the internal representation of UserController _uc to uc.

3.3.3.4.1.1 In case of invalid arguments, this method shall throw an IllegalArgumentException.

3.3.3.5 generateProjectile()

3.3.3.5.1 This method shall create a new Projectile object and add it to the internal _projectileList.

3.3.3.6 getCurrentMSec()

3.3.3.6.1 This method shall return the internal representation of time _currentMSec.

3.3.3.7 run()

3.3.3.7.1 This method shall run while the internal representation of system time is less than 100s.

3.3.3.7.2 This method shall update the DisplayClient _dc and the internal representation of time in secs and msecs every 100ms of system time.

3.4 Projectile

??

3.5 Vehicle Controller

3.5.1 Variables

3.5.1.1 The VehicleController shall contain a reference to the Simulator object _s with which it is associated.

3.5.1.2 The VehicleController shall contain a reference to the GroundVehicle object _v with which it is associated.

3.5.1.3 The VehicleController shall contain an internal representation of the time when the last control was issued.

3.5.1.3.1 The internal representation of the last control time shall be comprise of an integer representing the component of time in seconds and an integer representing the component of time in milliseconds.

3.5.2 Constructor(VehicleController)

3.5.2.1 The internal representations of the Simulator and the GroundVehicle shall be initialized according to constructor arguments.

3.5.2.2 The constructor shall take a Simulator s and a GroundVehicle v as arguments.

3.5.2.2.1 In case of invalid arguments, the constructor shall throw an IllegalArumentException

3.5.3 Methods

3.5.3.1 run()

3.5.3.1.1 This method shall call getControl when the system clock notes that 100 ms have passed.

3.5.3.1.2 This control shall be applied to the GroundVehicle _v.

3.5.3.2 getGroundVehicle()

3.5.3.2.1 This method shall return the GroundVehicle _v.

3.5.3.3 getSimulator()

3.5.3.3.1 This method shall return the Simulator _s.

3.5.3.4 getID()

3.5.3.4.1 This method shall return the VehicleController's ID _ID.

3.6 LeadingController

3.6.1 The LeadingController shall extend VehicleController

3.6.2 Variables

3.6.2.1 The LeadingController shall contain a list of references to the GroundVehicles _followerIndexList to evade.

3.6.3 Constructor

3.6.3.1 The representations of Simulator and GroundVehicle shall be initialized according to constructor arguments.

3.6.3.2 The constructor shall take a Simulator and GroundVehicle as arguments.

3.6.3.2.1 In case of invalid arguments, the constructor shall throw an IllegalArgumentException.

3.6.4 Methods

3.6.4.1 addFollower(GroundVehicle v)

3.6.4.1.1 This method shall add v to the list of GroundVehicles contained in LeadingController.

3.6.4.2 getControl(int sec, int msec)

3.6.4.2.1 This method shall generate a Control to be applied to LeadingController.

3.6.4.2.2 The Control shall be calculated such that the LeadingController moves away from the nearest GroundVehicle found with getClosestVehicle() as in 3.6.4.3.

3.6.4.3 getClosestFollower()

3.6.4.3.1 This method shall return the GroundVehicle whose spatial location is nearest that of the GroundVehicle associated with the Leading Controller.

3.6.4.4 tooCloseToWalls(double[] vehiclePosition)

3.6.4.4.1 This method shall return true if the spatial distance between the wall and the vehiclePosition is 20.

3.7 FollowingController

3.7.1 The FollowingController shall extend VehicleController.

3.7.2 Variables

3.7.2.1 The FollowingController shall contain a reference to the target GroundVehicle object _prey.

3.7.3 Constructor

3.7.3.1 The internal representation of _s, _v, and _prey shall initialize according to constructor arguments.

3.7.3.2 The constructor shall take a Simulator s, GroundVehicle v, and GroundVehicle prey as arguments.

3.7.3.2.1 In case of invalid arguments, the constructor shall throw an IllegalArgumentException.

3.7.4 Methods

3.7.4.1 getControl(int sec, int msec)

3.7.4.1.1 This method shall return a Control to be applied to FollowingController.

3.7.4.1.2 The Control shall be such that the FollowingController's GroundVehicle moves towards the targetVerhicle.

4. External Interface Requirements

4.1 User Interfaces

The main user interface shall be encompassed by the visual DisplayServer and the user-controlled UserController.

4.2 Hardware Interfaces

There shall be no hardware interfaces beyond the standard keyboard on the user's personal computer.

4.3 Software Interfaces

All software interfaces shall occur between the classes specified in section 3 and within the Java framework.

4.4 Communication Interfaces

Remote access shall be possible to allow multiple users to compete on the same game.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

This system shall perform to the specifications laid forth in this document.

5.2 Safety Requirements

There are no safety requirements associated with this system.

5.3 Security Requirements

There are no security requirements associated with this system.

5.4 Software Quality Attributes

This system shall have methods that have been rigorously tested using a unit test matrix.