**Test Plan**

**Team Honeybadger’s: Monte Carlo Localization Simulator**

**SE300**

**Version 2**

**Savanh Lu, Miralda Rodney, Julian Pryde, Stephen Kristin**

**Table of Contents**

**1. Identifier**

**2. Introduction**

**2.1. Purpose**

**2.2. Background**

**2.3. Scope**

**3. Test items**

**3.1. Program modules**

**3.2. Use cases/Requirements**

**4. Testable Features**

**5. Non-Testable Features**

**6. Approach**

**6.1. Unit / Integration Tests**

**6.2. System Tests**

**6.3 Suspension/Resumption criteria.**

**7. Test Deliverables**

**8. Testing Tasks**

**9. Environmental Needs**

**9.1. Hardware**

**9.2. Software**

**10. Responsibilities**

**10.1. System test group**

**10.2. Artificial Intelligence Club**

**10.3. Development project group**

**11. Schedule**

1. **Identifier**

MCL.1.1-A

**2. Introduction**

**2.1 Purpose**

The purpose of the test plan for the Monte Carlo Localization Simulator is to describe a set of basic software test documents. The documents will 1) Detail activities required to prepare for and conduct the system test. 2) Show communication between all responsible parties, the tasks that they are to perform and the schedule on which the tasks must be completed. 3) Define the sources of the information used to prepare the plan. 4) Define the test tools and environment needed to conduct the system test.

**2.2 Background**

The Artificial Intelligence club at Embry-Riddle Aeronautical University are conducting a 5 year plan to create an Ambassador Robot (AmBot). The purpose of the robot is to navigate around and through campus while interacting with humans. A small group of students separate from the Artificial Intelligence club are creating a virtual location simulator to test the location accuracy of a virtual robot and collecting data that may be used for reference by engineers working on the Ambassador Robot.

The Monte Carlo Localization Simulator is a two-dimensional simulation that will allow a user to define an environment for the operation of a robot and its sensor. The user should be able to define where the robot will start the simulation and where the robot will end the simulation. The program will randomly generate the point locations in the program. The user will specify the range and accuracy of the sensors, the amount of error in the sensors, the amount of error in the robot’s movement itself. The correctness of the robot’s operation will be documented and saved.

**2.3 Scope**

This test plan covers a full systems test of the Monte Carlo Localization Simulator.

**2.4 References**

The Monte Carlo Localization Simulation SRS Document version 1.0  
Julian Pryde, Stephen Kristin, Savanh Lu, Miralda Rodney

**3. Test items**

All files in the Monte Carlo Localization Simulator will be tested in this systems test.

The project requirements can be found in the Software Requirements Specification document. The Users guide, Operations guide, and Installation guide can all be found in the project README.md file.

**3.1. Program modules**

The files to be tested are:

Chart.java

GUI.java

IO.java\*

Main.java\*

Map.java\*

Robot.java\*

SensedMap.java\*

Sensor.java\*

The starred files above are the ones which contain testable features.

**3.2. Use cases/Requirements**

**Use Case:**

Display main GUI

Create Robot

Create map of reference points

Create Sensor

Move from specified start point to specified end point

Detect points with sensor

Add error to position

Add error to sensed points

Detect position of robot

Write final positions with and without error from run to output file

Write reference point positions with and without error from run to output file

Write calculated positions to output file

Read information from output file

Display positions and errors on graphs

**4. Testable Features**

The following features will be tested:

Find position in field of reference points by sensing reference points

Display GUI to take parameters for run

Write output of each run to a file

Display graphs of discrepancy between actual and calculated positions

Display graphs of discrepancy between actual and sensed points

Functionality of the GUI

**5. Non-Testable Features**

Program initialization and GUI load.

**6. Approach**

The test group for the project will be using the system test plan to enumerate the test cases, the procedures required to properly perform the tests and the testing criteria that will determine whether or not a test fails for all the features implemented in the program.

**6.1. Unit / Integration Tests**

The group will test integration of each unit as they affect the project as a whole. As each feature is implemented, it is added to the program and integrated into the entire system. The group will then test how the entire project responds to the addition of the feature by testing for the correct operations.

**6.2. System Tests**

**Class Name**: GUI

**Test Name**: GUI Input test

**Description**: When the GUI is shown after the program is run, there are text boxes that allow for user input. The user has a constraint on the kind of input that the program is expecting. To test the proper functionality of the program, the tester will input incorrect information such as letters or special characters as well as correct numerical information in the format prompted by the program. Once the “start simulation button” is pressed, the program will either run correctly or it will given an error.

**Class Name**: Sensor

**Test Name**: Detected Points File Created

**Description**: When the sensor detects a point, it documents the location of the points that it has detected along with the robot’s location at the time that it senses the point in a file that is formatted in a certain way.

**Class Name**:Robot

**Test Name**: Moves one unit

**Description**: When the program runs, through the chain of operations, in the robot class, the robot should move one unit toward the endpoint. The next location should be the current location (x and y) multiplied by the constant sqrt(2)/2. The proper operation can be verified using JUNIT testing and assertEquals to make sure consecutive points are multiples of the constant.

**Class Name**: Sensor

**Test Name**: Robot’s Estimated Location

**Description**: While the sensor is sensing points, it is also estimating the robot's location from the points around it. The detected points file should be complete with a six rows. These rows are the estimated location of the reference points, the actual location of the robot, and the robot's estimated location.

**Class Name**: Chart

**Test Name**: Display graph of discrepancies

**Description**: Once all of the calculations have been performed for the aggregation of error, a graph should pop up displaying the correct information for the run.

**Class Name**: IO

**Test Name**: Writing to File

**Description**: After the start simulation button has been pressed and the program has run, there should be a file that has been written by the IO class that includes all the calculations for the charts.

**Class Name**: Map

**Test Name**: Random Reference Points

**Description**: The program should generate random reference points and the correct number of reference points. JUNIT tests can be used to make sure that user input equals the number generated.

**6.3 Suspension and Resumption Criteria**

**6.3.1 Suspension Criteria**

When a defect is introduced that cannot allow any further testing.  
Critical path deadline is missed so that the client will not accept delivery even if all testing is completed.  
A specific holiday shuts down both development and testing.

**6.3.2 Resumption Criteria**

When a fix is successfully implemented and the Testing Team is notified to continue testing.  
The contract is renegotiated with the client to extend delivery.  
The holiday period ends.

**7. Test Deliverables**

Test Plan

Test Specifications

Test Reports

**8. Testing Tasks**

See Attachment A.

**9. Environmental Needs**

Testing does not need any specific environment due to its computer-based nature. Must meet the minimum system requirements below.

**9.1 Hardware**

The user must have a computer at their disposal with memory that allows for an installation of an application that runs java-based files.

**9.2 Software**

Software with java-based platforms is necessary to test the Monte Carlo Localization Simulator.

**10. Responsibilities**

**10.1. System test group**

This group provides the overall management of the testing and the technical testing expertise. The members of this group are: Julian Pryde, Miralda Rodney, Stephen Kristin, and Jadeira S Lu.

**10.2. Artificial Intelligence Club**

This group is the client requested by Professor Keith Garfield and will provide assistance to the test group in the following activities:

Consultation on direction and data

Checking output screens and reports

System testing

**10.3. Development project group**

This group transmits the system to be tested and responds to the System Test Incident Reports. This group does any program debugging that is required. The members of this group are: Julian Pryde, Miralda Rodney, Stephen Kristen, and Jadeira S Lu.

**11. Schedule**

See Attachment A.

**Attachment A. Task List and Schedule**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **Task** | **Predecessor Tasks** | **Special Skills** | **Responsibility** | **Effort** | **Finish Date** |
| 1 | Prepare Test Plan | Complete MCL design and development plan | - | System Test Group | 7h | 4/8/2016 |
| 2 | Prepare Test Design Specification | Task 1 | Knowledge of MCL Procedures | System Test Group | 1h | 4/8/2016 |
| 3 | Prepare Test Case Specification | Complete test designs  Task 2 | - | System Test Group | 4h | 4/8/2016 |
| 4 | Prepare Test Procedure Specification | Complete test cases  Task 3 | - | System Test Group | 4h | 4/11/2016 |
| 5 | Test the system feature: GUI button functionality | Task 4 | - | System Test Group | 2h | 4/12/2016 |
| 6 | Test the system feature: GUI input functionality | Task 4 | - | System Test Group | 2h | 4/13/2016 |
| 7 | Test the system feature: Robot movement, without error | Task 4 | - | System Test Group | 3h | 4/14/2016 |
| 8 | Test the system feature: Robot movement with error | Task 4 | - | System Test Group | 3h | 4/15/2016 |
| 9 | Test the system feature: Reference points | Task 4 | - | System Test Group | 2h | 4/16/2016 |
| 10 | Test the system feature: Reference points with error | Task 4 | - | System Test Group | 2h | 4/17/2016 |
| 11 | Test the system feature: Robot acquiring location | Task 4 | - | System Test Group | 4h | 4/19/2016 |
| 12 | Check the tests results | Task 5 - 11 | - | System Test Group | 3h | 4/21/2016 |
| 13 | Resolve test incident reports. | Task 12 | - | System Test Group | - | 4/21/2016 |
| 14 | Repeat Tasks 5 - 11 until all tests have passed. | Task 13 | - | System Test Group | - | 4/17/2016 |
| 15 | Compose the system test summary report | Task 14 | - | System Test Group | 4h | 4/22/2016 |