Jackson Lawler and Nicholas Setliff
Test Plan
Capstone

Abstract

We will be utilizing a number of testing strategies in our design. We will be using the overall strategy of TDD and therefore the approach will be top-down. We will write test cases that must pass for implementation to be correct and then make each test pass one at a time. There will be test cases for each feature to ensure their correctness. Unit testing will be done first, followed by the usual integration and system testing phases. Each new unit will be integrated into the whole one piece at a time, therefore integration testing will be done throughout the course of the project rather than in one large session. We are using an Agile TDD approach, so there will be small, incremental working releases as each major milestone is made. The entire work schedule, as such, will not be entirely fleshed our beforehand because of our agile approach, but we do have a rough idea of what each sprint will likely contain:

Sprints

1. Sprint 1 – Working Graph:
   1. Make the graph
   2. Store the graph on a server
   3. Make an API to add nodes to the graph
   4. Make an API to request the graph
2. Sprint 2 – Use the Graph:
   1. Write navigation routines to navigate the graph
      1. Handicap
      2. non-handicap
      3. Fitness will be placed at a lower priority than the other two and maybe be pushed back if needed
3. Sprint 3 – GUI:
   1. Implement the non-AR navigation
   2. Implement the AR Navigation
   3. Real-life testing
4. Sprint 4 – Settings:
   1. Create the defaults and user settings view laid out in the SRS
   2. Make a user-friendly front-end for the add nodes API of the graph so that others may add nodes

Notes: each sprint is intentionally created in such a manner that each can be worked on independently of one another. Nothing in a previous sprint forbids working on another sprint and therefore sprints might be worked on concurrently one different teams. Example: the graph does not have to be created to make the navigation work. We can mock that up enough for a UI to be created that can access the graph after it is completed.

Specific testing strategies:

- BVA will be used where appropriate to ensure there are no edge-case problems
    - Any if/then/else or switch/case block will use BVA to test any and all such edge cases
- Special value testing may be used if appropriate in certain areas
- Extended decision tables will be created for certain parts of the program (e.g. a fitness program that requests more vertical distance traveled than total distance traveled is impossible)
- Unit testing will use built-in suites for the languages used as appropriate.