16.35 PSet #1

Ryan Fish

February 17, 2015

Pre-deliverables

1 Real-time Systems and Software

- Learn the language of software requirements so I can communicate them to a design team, and interpret needs from conversations with customers.
- Be exposed to common errors made in this field so I may learn by example rather than experience. Life-critical systems aren't a good place for learning by experience.
- Practice building real-time software and systems to prepare for a hopeful career in embedded mechanical control systems.

2 Documentation

2.1 Java version on Athena

1. 1.7.0_65
2. ran `java -version` on an athena remote session
3. Googled `check java version linux` and logged in to athena remotely, maybe 45 seconds all told

2.2 Enable assertions

1. Assertions in the program are enabled by running java with the `-ea` flag
2. http://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html#enable-disable
3. 20 seconds via Google

2.3 Double to string

1. `String doub = Double.toString(num)` where `num` is the input double and `doub` is the output String
2. http://docs.oracle.com/javase/7/docs/api/java/lang/Double.html
3. found via google, 2 minutes to verify it produces the expected output

2.4 Create jar with files from dir `asst1`

1. `jar cf asst1.jar asst1`
2. http://docs.oracle.com/javase/tutorial/deployment/jar/build.html
3. 20 seconds via Google

4 Reqs and Unit Testing

4.1 Faults in Assignment requirements

1. No outline of variables or state that gets referenced in method description
2. Requirements are not numbered, harder to trace to tests
3. Requirements are not always written in a specifically testable manner, and use should poorly

4.2 Specific failings

1. The specification of the output of the program is a should statement, something like output format needs to be determined with a shall statement.
2. The simulator class clock is not well defined in terms of purpose or format.
3. No description is given for how to deal with failure to provide valid inputs e.g. to the GroundVehicle constructor.

4.3 GroundVehicle reqs

TODO

4.4 List of Unit Tests

4.4.1 Test needed

1. my util.clampDouble method
2. my util.clampInt method
3. my util.wrapAngle method
4. Control constructor
5. controlVehicle
6. getVelocity
7. setVelocity
8. updateState
9. getControl
10. setNumSides
11. main

4.4.2 No Tests needed

1. getSpeed - simple getter
2. getRotVel - simple getter
3. getPosition - simple getter
4. getCurrentSec - simple getter
5. getCurrentMSec - simple getter
6. setPosition - relies on clamp and wrap
7. GroundVehicle constructor - relies on setPosition and controlVehicle
8. run

4.4.3 Equivalence Classes

1. First arg greater than bounds, less than bounds, within bounds, again for second arg
2.

5 Output

6 Code control

6.1 Subversion Log

1. ```
   $ git commit -am "oops, now I have read section 6, and rearranged files properly"
   [master 6064bee] oops, now I have read section 6, and rearranged files properly
   11 files changed, 249 insertions(+), 14 deletions(-)
   rename asst1/{src => }/Control.java (100%)
   rename asst1/{src => }/GroundVehicle.java (100%)
   rename asst1/{src => }/Simulator.java (100%)
   rename asst1/{src => }/util.java (100%)
   ```
2. maybe 6 hours on the code and 30 seconds on the file copying and committing.
   ```

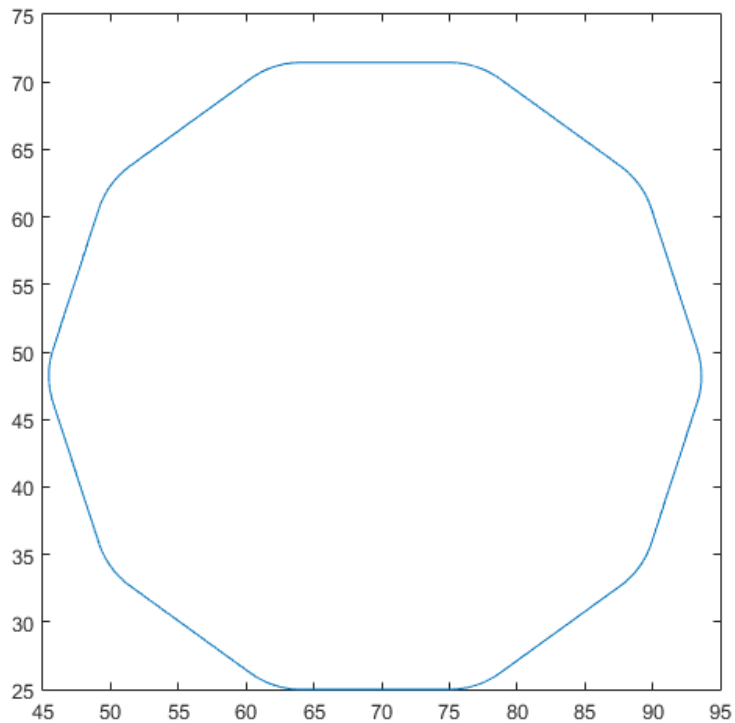

Figure 1: Decagon

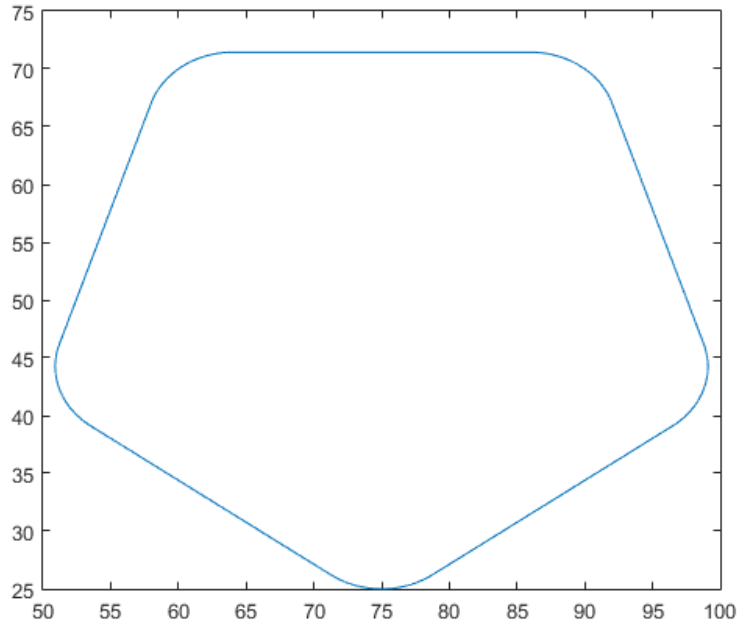

Figure 2: Pentagon

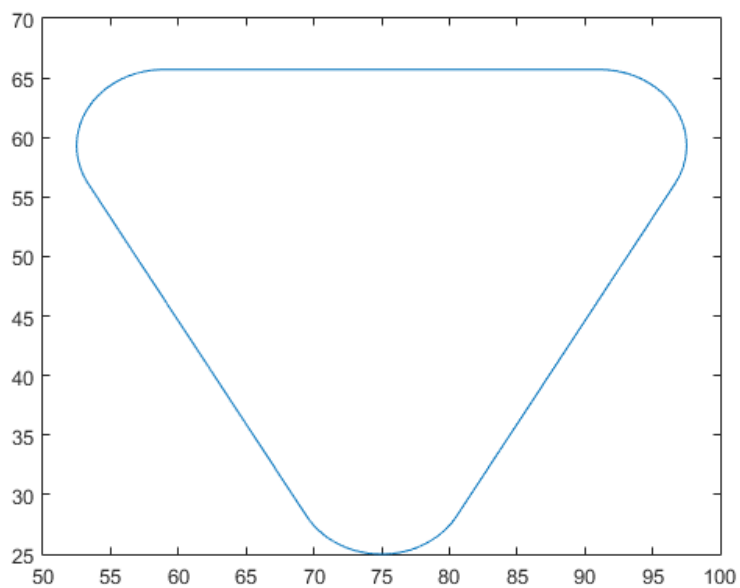

Figure 3: Triangle