16.35 PSet #2

Ryan Fish

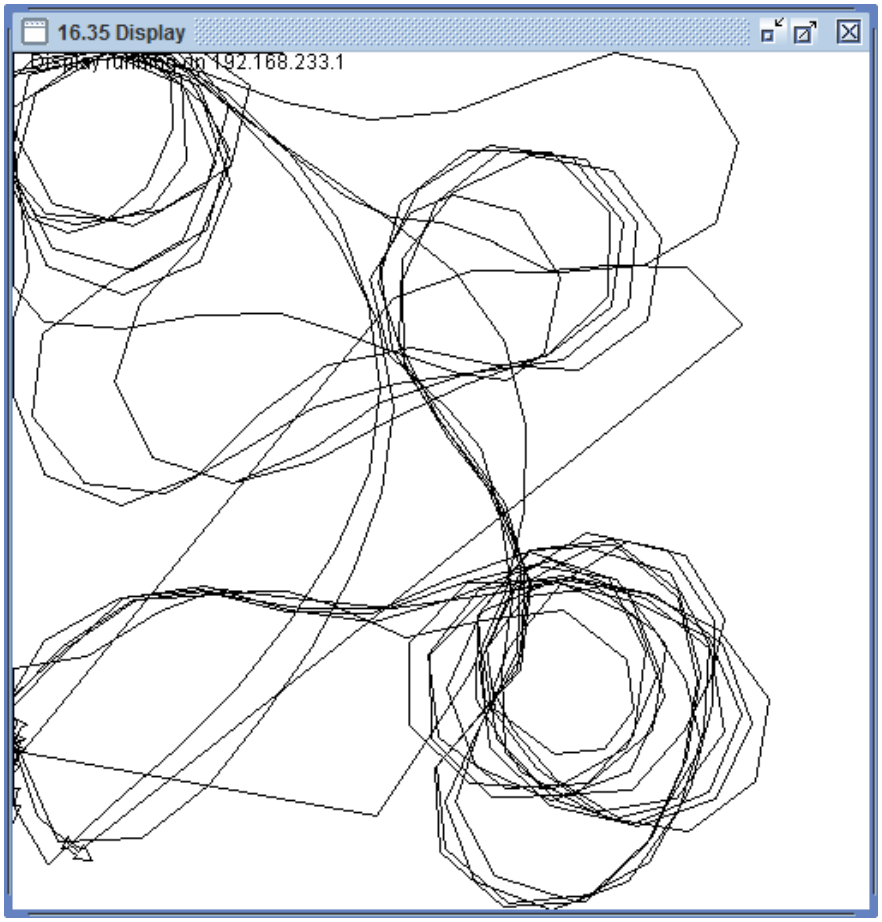
March 20, 2015

1 Predeliverables

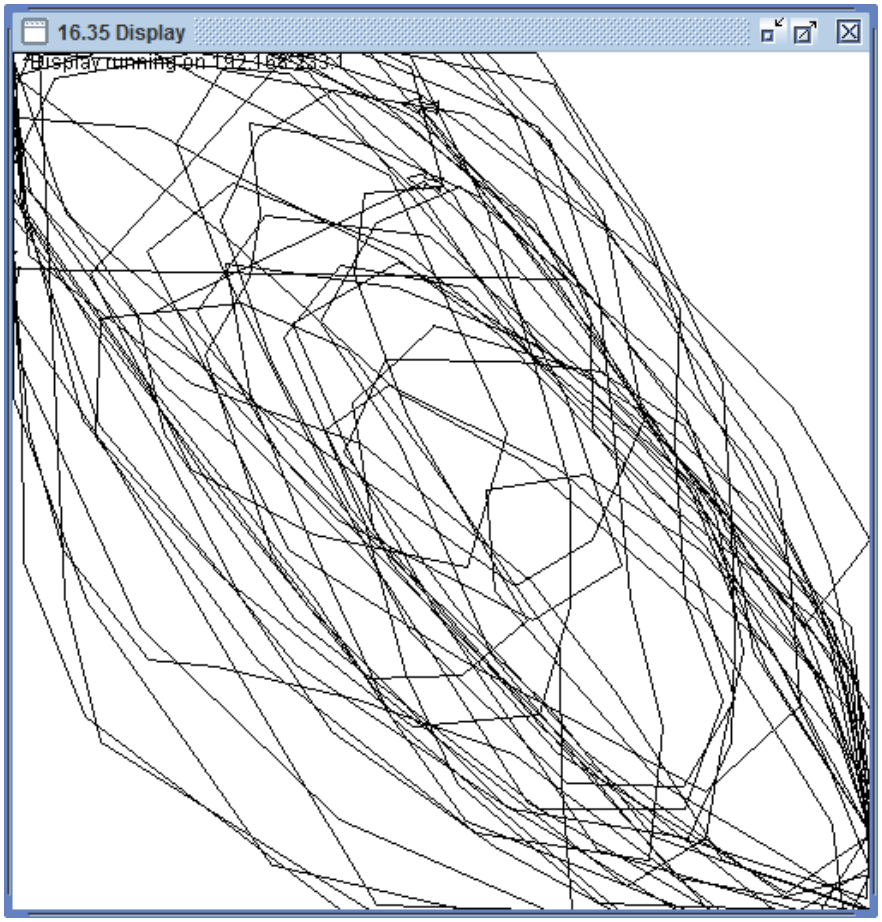
1.1 Test Case Rationale

TODO

1.2 No Noise, 5 Vehicles



1.3 Noise, 5 Vehicles



1.4 4 Conditions of Deadlock and You

MutEx We need to prevent reads occurring concurrently with writes to vehicle state.

Hold and Wait In particular, LeadingController may grab its own state and wait for a lock on the state of a vehicle it is leading.

No Preemption Without modification, the default synchronization system of Java does not allow for preemption in acquisition of resources.

Circular Wait A Leading Controller may wait on acquisition of the state of a Following controller, who is waiting to acquire the state of the same LeadingController.

1.5 Proposal for Avoiding Deadlock

The best way to attack deadlock in this scenario would be to attack circular wait. I will build an object to manage monitor acquisition that will assign a monotonically increasing index to object classes able to be locked, and prevent acquisition of an index lower than the highest one held. It will trigger a release of any objects held with higher indexes.