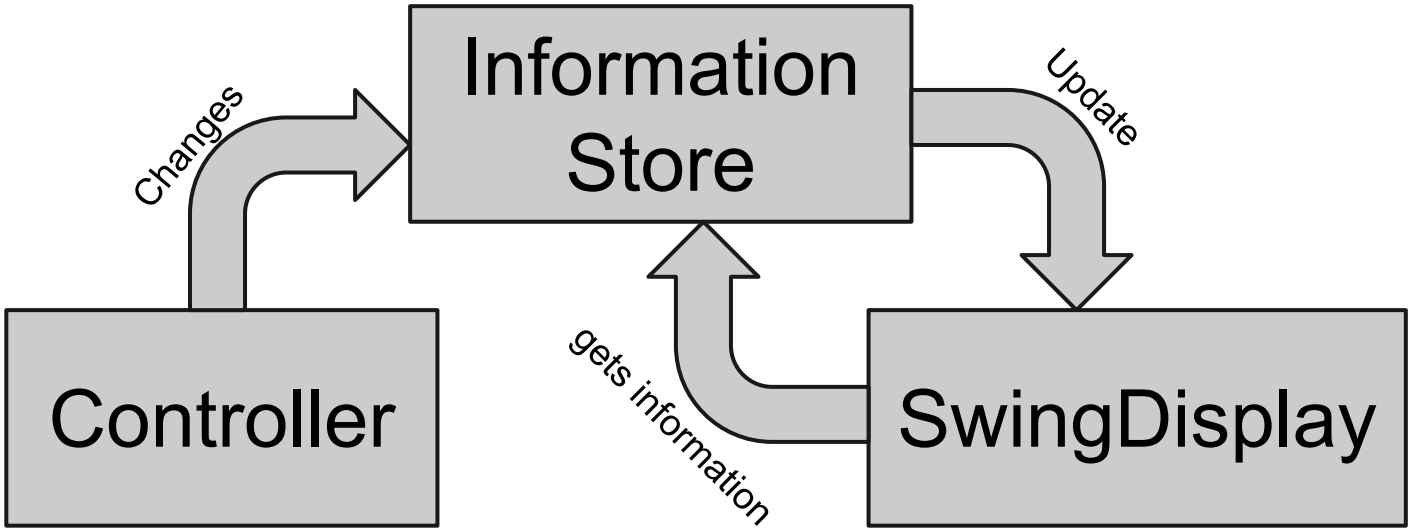
Finite State Machine

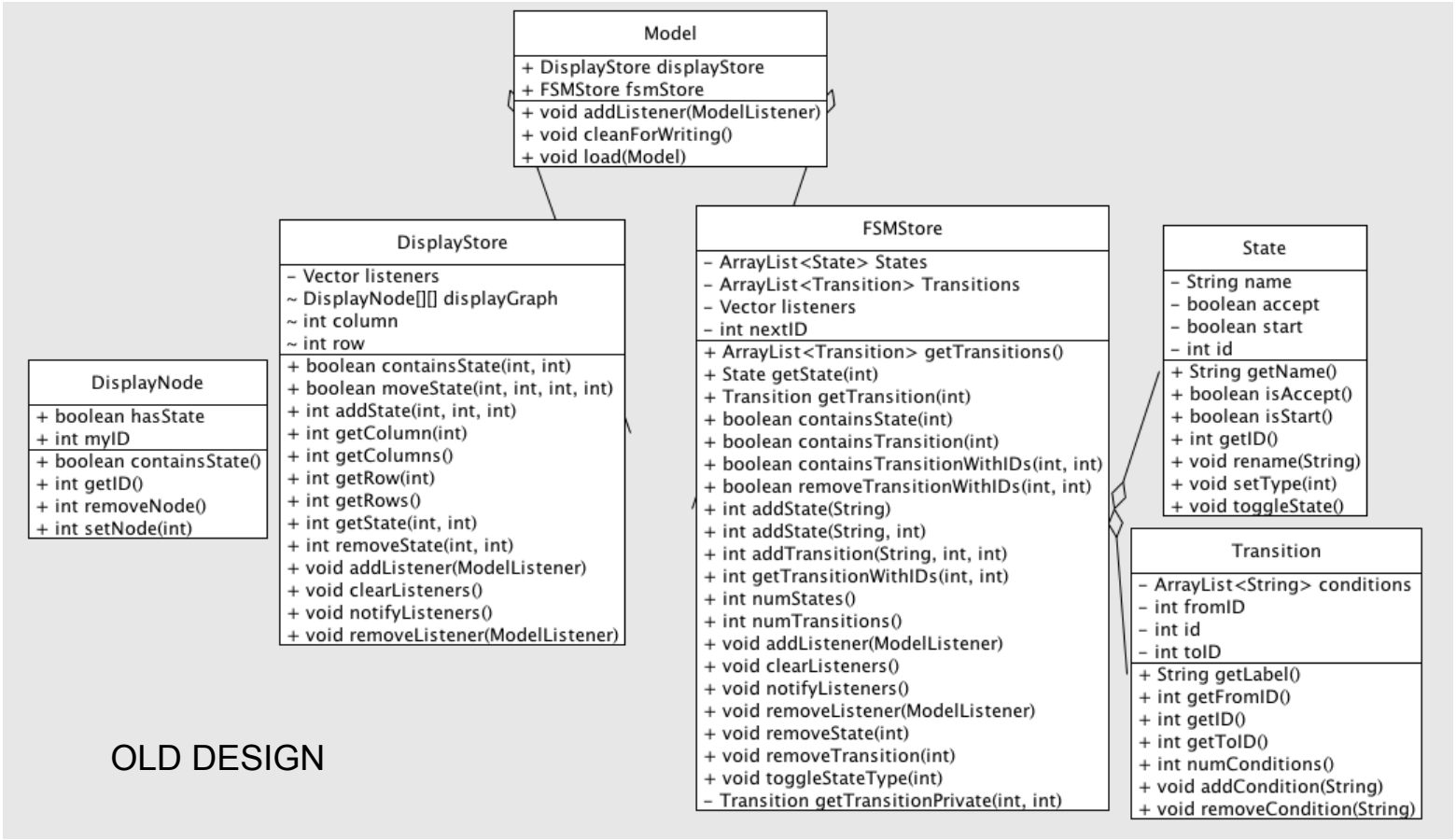
Frank, Rudy & Nate

Updates:

What have we done?

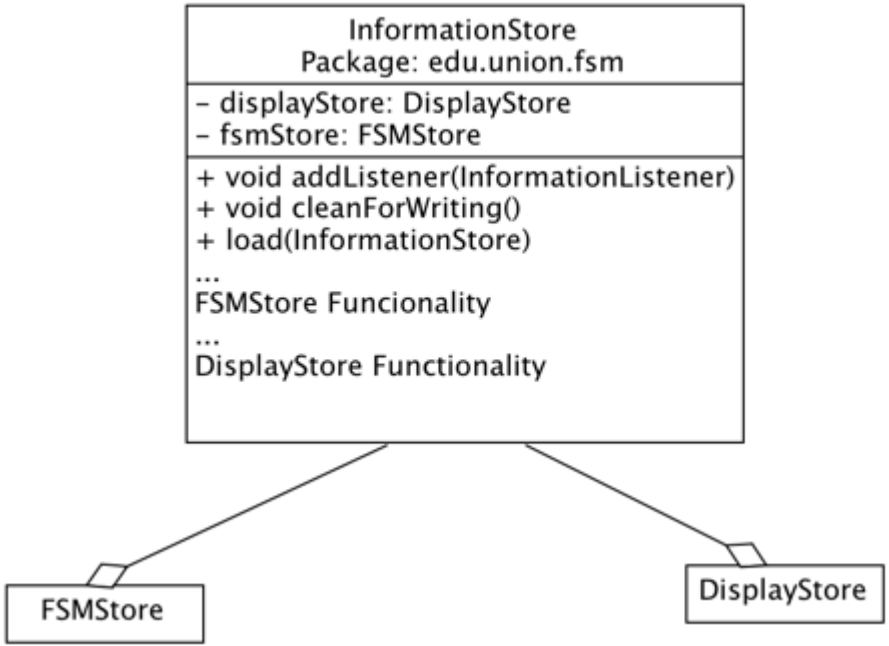
Model-View-Control Layout





Storage Changes: Modularizing

(pulling the functionality into the model itself)



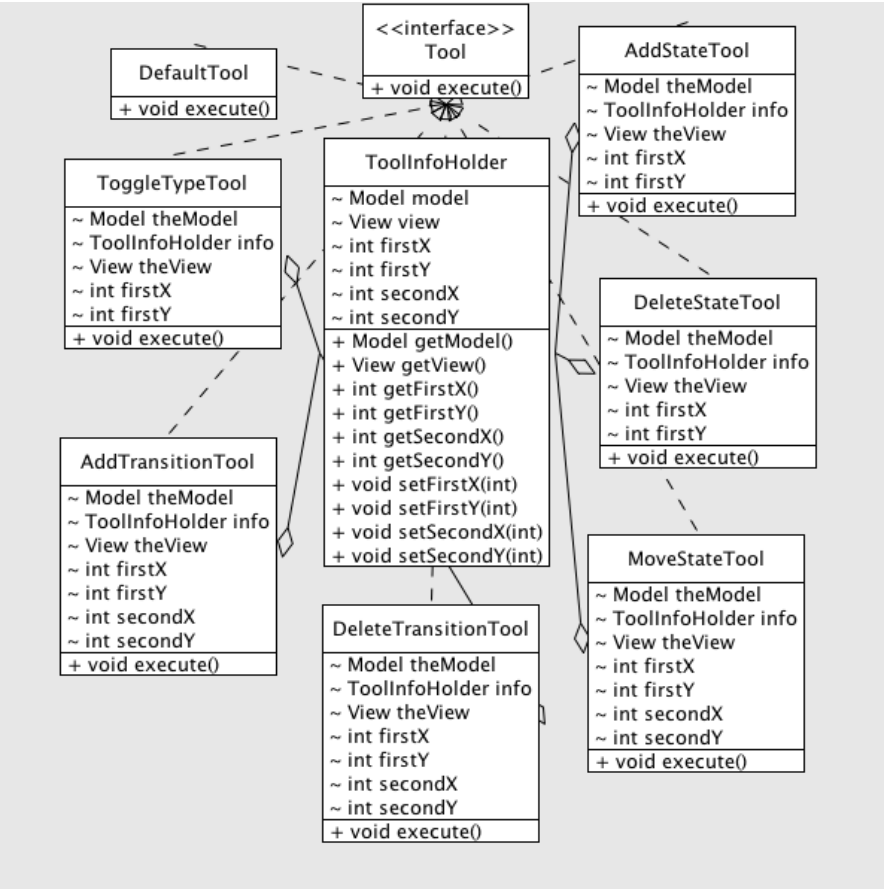
NEW DESIGN (MOCKUP)

Storage Changes: Modularizing

(pulling the functionality into the model itself)

Controller Changes

- Pulling out Tools to their own package
- storing data in custom Tool Info wrapper
- streamlining data calls to the InformationStore

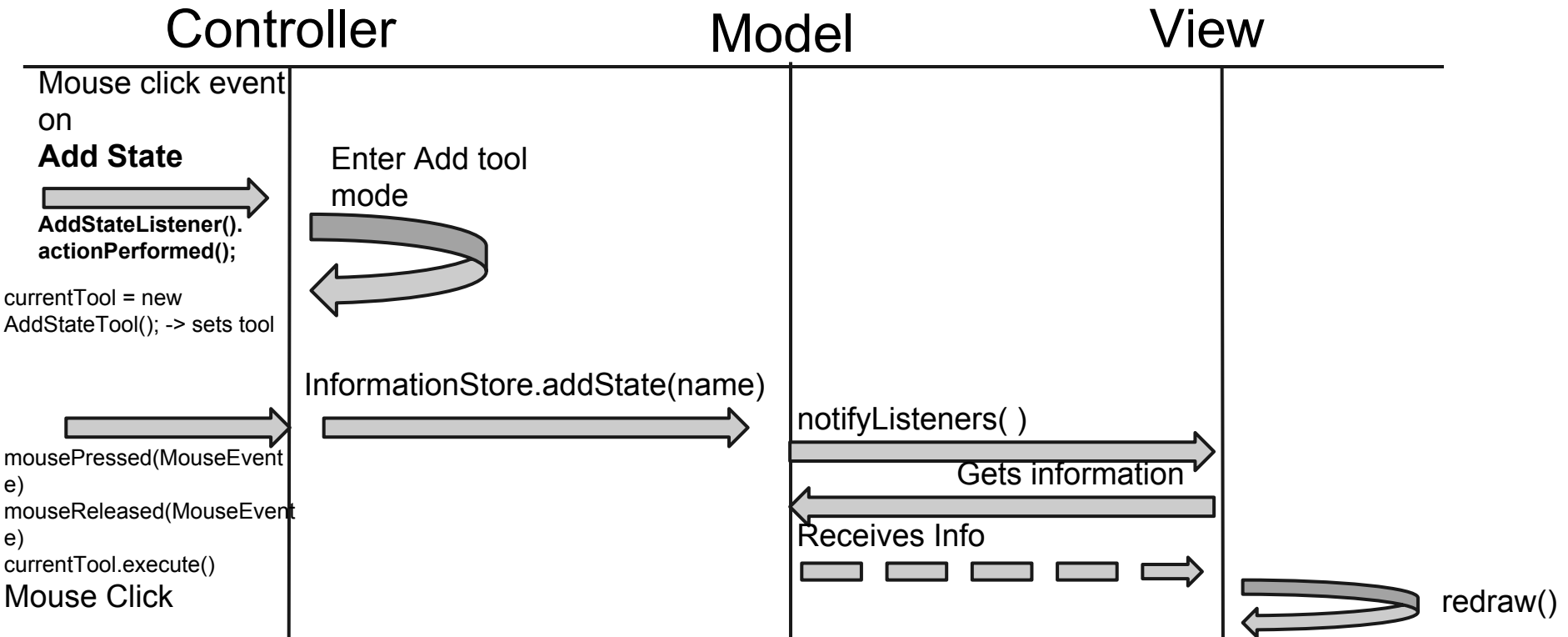


Tool Changes: New Package (edu.union.fsm.tool)

Controller Changes to do:

- pull out the listeners into their own package as well
- possibly pull out mouse listener functionality

Click on Empty Location with create State Tool - Sequence Diagram



Saving and Loading

- Serializable: for saving and loading, forces users to save to .bin files.
- Image: saves png representation of the FSM
- Independent of FSM Program (reusable)
- allows for saving and loading anywhere using a file chooser.

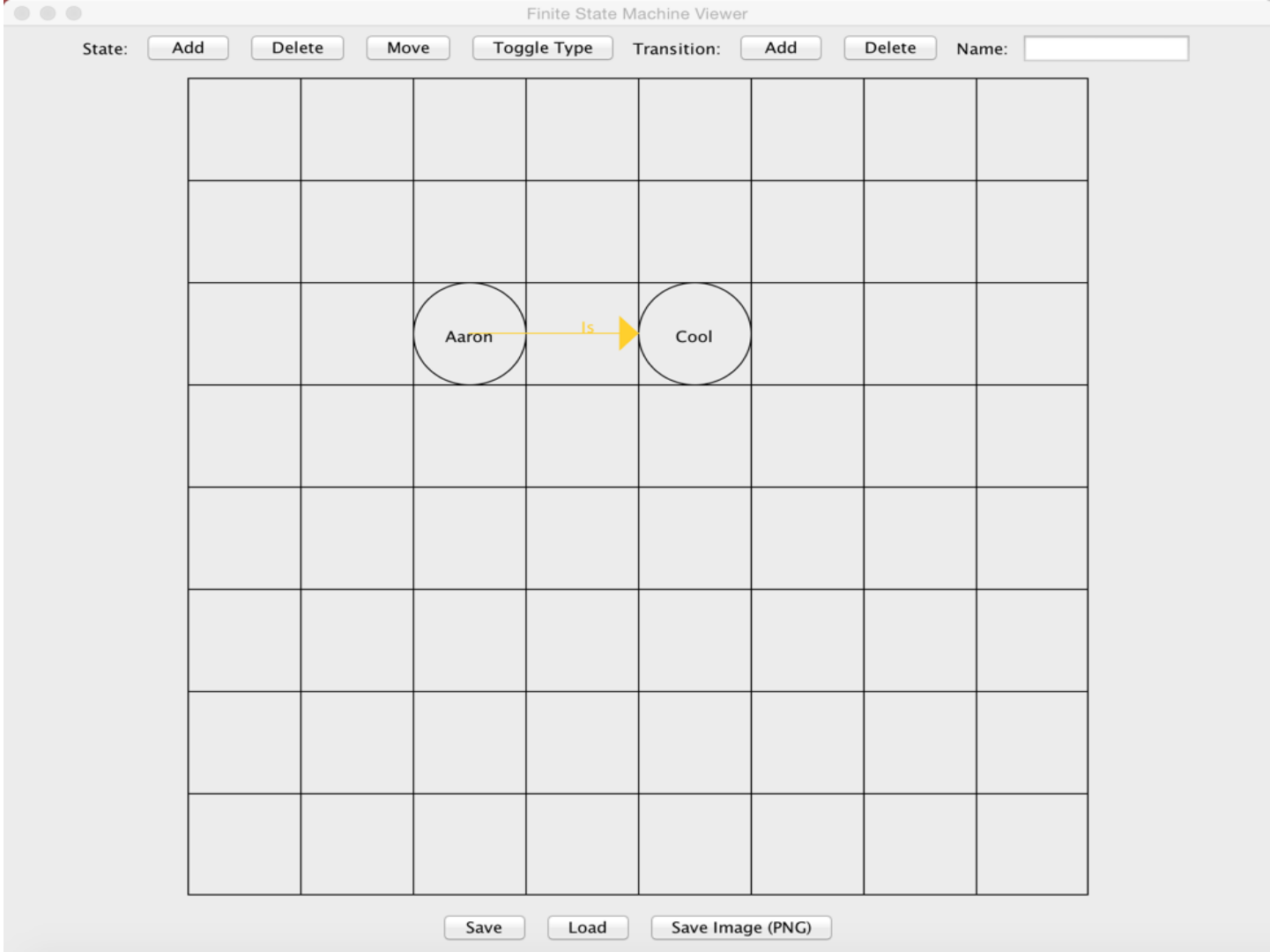
Saving and Loading (cont)

SaveBin
- Object toSave - JFrame toPrompt
+ void saveFile

loadBin
- Object toLoad - JFrame toPrompt
+ Object loadFile

SavePNG
- JComponent toSave - JFrame toPrompt
+ void saveFile

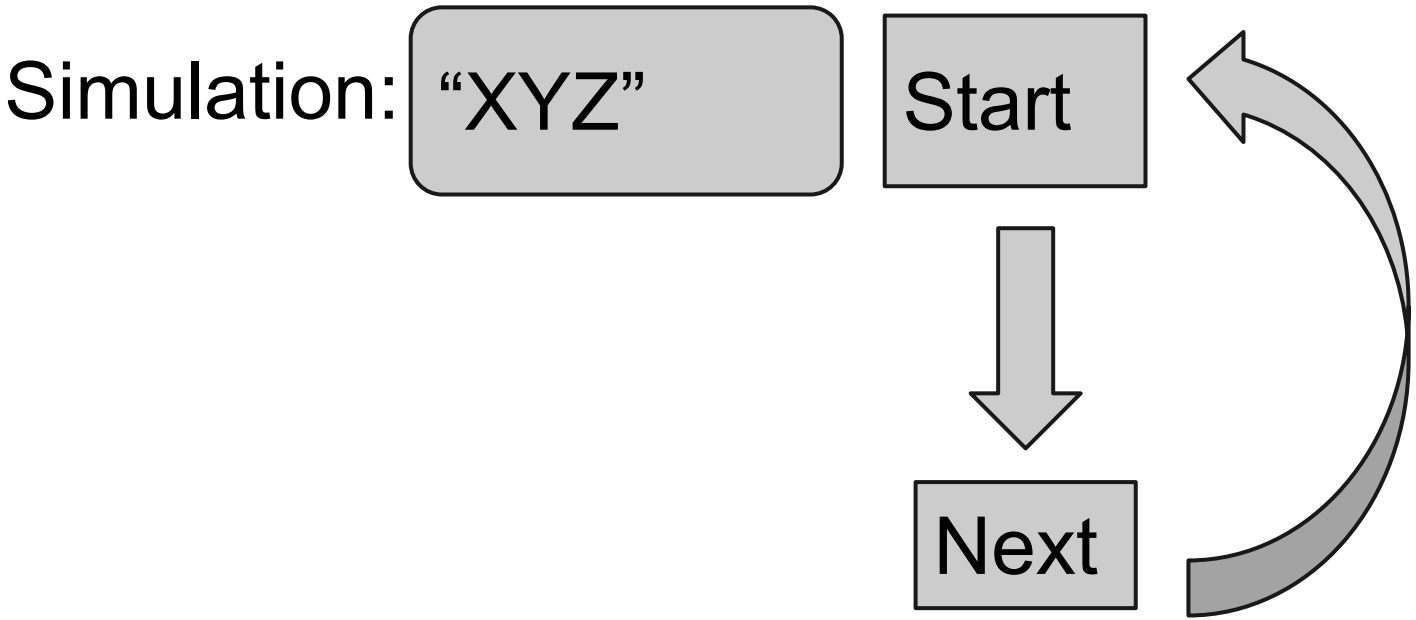
User Interface Diagram



Future Excitement

SIMULATIONS

UI for simulation



InformationStore Extended API

InformationStore

highlightState(int x, int y)
  boolean
highlightState(int id)
  boolean
unhighlightState(int x, int y)
  boolean
unhighlightState(int ID)
  boolean
clearHighlight()
  boolean
getStart()
  int id (null if no state)
getNextStates(String name, int fromID)
  ArrayList[toID]

Simulate Tool Functionality

SimulateTool
Start: Parse simulation text box into a Queue of transition labels. getStartState() highlightStartState() Change to Next button & Next mode if Next Mode: pop Queue check if valid transition highlight new states accordingly if Queue is empty: clear text reset tool to start mode

Demo

YAY!

Any Questions?