GlassLine Front End FAQ

Brehon Humphrey (bhumphre@usc.edu) & Jonathan Sun (sunjonat@usc.edu)

1. How do I get started?
    a. First, open the project in Eclipse. This can be done through File → Import → General → Existing Files Into Workspace.
    b. You will see four packages: engine, gui, shared, and transducer.
        i. engine.agent is where all the agents should go (just import your classes over to the new project)
        ii. gui.components is where all the front end components are stored. If you wish, you can change the animations for components to suit your needs.
        iii. gui.panels.subcontrolpanels is where each of the panels in the control panel is stored. These will have to be edited to suit your needs.
    c. If you run the project, you will see a window with the factory. Currently, because no back end is attached, nothing will move. You can change the window title by changing the parameter in gui.driver.FactoryFrame.java.
2. How do I connect the front/back ends together?
    a. First, you need to make sure all your Agents are communicating with their GUI counterparts through the Transducer. This is achieved using the Transducer.register() and Transducer.fireEvent() methods.
    b. Some events require special parameters, such as the index of the workstation, to be passed in the args array. Consult the API and ensure that the events are passing these properly.
    c. In the gui.panels.FactoryPanel class, there is a method, initializeBackEnd(). This is where you should put any and all Agent initialization, including starting threads and registering with the transducer (which is contained in the FactoryPanel class).
    d. Implement the Control Panel as per the back end design. This involves implementing the eventFired() and setTransducer() methods in gui.panels.ControlPanel, as well as gui.panels.subcontrolpanels.GlassInfoPanel, GlassSelectPanel and NonNormPanel (for

V2). A sample TracePanel has been provided to store Agent prints, and you are free to add it and use it in the ControlPanel.

3. What is the Transducer for?
    a. The Transducer class provides an easy, modular way for the front/back ends to communicate without requiring that they be tightly coupled.
    b. Students who attended the Transducer lab earlier in the semester should know how to register with the Transducer and fire/receive events through it.
    c. As per your design, feel free to add/remove channels/events as you wish. The point of the Transducer is to be completely flexible to any back end design.
    d. You can run the program with various command line arguments to print certain Transducer events to console:
        i. tdebug1 will print a message to console every time the Transducer fires an event or registers/unregisters a TReceiver. Use this to confirm that the proper events are being fired.
        ii. tdebug2 will print everything from tdebug1, but will also print what the Transducer is internally doing, including what order events are queued up in. This will give you more information about the internal state of the Transducer, such as which channels have already been registered.
        iii. tdebug3 will print everything from tdebug2, but will also print the Transducer's internal scheduler state, including what events it will process next. This will give you the most information about how events are concurrently processed.
4. What if I have questions/find bugs in the front end?
    a. For individual graphical component/animations/imageicon questions, email Brehon Humphrey (bhumphre@usc.edu).
    b. For paneling/API/transducer questions, email Jonathan Sun (sunjonat@usc.edu).
    c. For requirements issues, contact Professor Wilczynski directly.