Migrating Behavior Search’s User Interface from Swing to JavaFX

An Nguyen Dang
Augustana College, Rock Island Illinois

Follow this and additional works at: https://digitalcommons.augustana.edu/celebrationoflearning

Part of the Business Administration, Management, and Operations Commons, and the Computer Sciences Commons

Augustana Digital Commons Citation
https://digitalcommons.augustana.edu/celebrationoflearning/2017/posters/10

This Poster Presentation is brought to you for free and open access by Augustana Digital Commons. It has been accepted for inclusion in Celebration of Learning by an authorized administrator of Augustana Digital Commons. For more information, please contact digitalcommons@augustana.edu.
I. Introduction

Agent-Based Models (ABMs) and NetLogo

- Agent-based modeling is a computer modeling technique that focuses on modeling the rules of individuals ("agents") and simulating the interactions between these individuals.
- ABMs are widely used to simulate behavior in many fields including archaeology, biology, economics, and social science.
- NetLogo is an agent-based modeling language and integrated modeling environment. It is a popular platform for building and running ABMs. (See: http://ccl.northwestern.edu/netlogo/)

BehaviorSearch

- BehaviorSearch is a tool to help automate the exploration and analysis of ABMs. (See: http://www.behaviorsearch.org/)
- This software interfaces with the NetLogo platform and allows the client to search for combinations of model parameter settings that will result in a specified target behavior.

A screenshot of the BehaviorSearch software before this project.

II. Motivation

Java Swing Graphical User Interface (GUI)

- Earlier versions of BehaviorSearch used the Swing GUI library
- With Swing, all of the graphical components and controlling methods get embedded in the same code, which makes the code long and hard to debug
- The Swing GUI style looked and felt dated (e.g. like Windows 2000)
- Application didn’t scale well to modern high-resolution screens
- Did not support displaying charts inherently
- BehaviorSearch used the third-party JFreeChart library for this
- All of the input forms were packed into one cluttered window

JavaFX Graphical User Interface (GUI)

- JavaFX fixes many of the problems with Swing
- It separates the graphical component details into a separate .fxml file that is easy to read and change
- It also provides SceneBuilder, which allows developers to easily modify .fxml with real-time preview using drag and drop
- Programs can be styled with CSS3, allowing easy changes of style
- Supports chart components (no third party library required)
- Organizes input forms into tabs for a less cluttered display

JavaFX supports charts inherently. This makes it easier to get access to, since they are created automatically by the FXMLLoader. This makes it hard to pass data from one scene to another scene.

Solution: Get the Controller object from the FXMLLoader, then pass parameters into the controller using an init method.

Example:
```
RunOptionDialogController runController;
runcController = loader.getController();
runcController.ini(runOptions, this);
```

Table Functionality in JavaFX

- The default table in JavaFX requires users to hit ENTER to commit each value they entered. This was not a natural behavior for the table in this software and could confuse users.

Solution: Design a custom subclass of the TableView class that is named AcceptOnExitTableCell (class outline shown at right), which overrides the default table behavior such that user’s entries are stored regardless of whether they hit ENTER.

III. Challenges

Multithreading in JavaFX

- When dealing with time-consuming computational tasks, like what BehaviorSearch does to analyze models, it is important to do those tasks in a parallel worker thread, so that the GUI stays responsive.
- Java Swing had a specialized solution for this, a SwingWorker class does all the computation in the background
- JavaFX doesn’t have similar class to SwingWorker

Solution: Create the new thread manually, create a task worker that implements the Runnable interface, and then use Platform.runLater() to update the GUI.

Utilizing Multiple Scenes

- Scenes in JavaFX are not easy to get access to, since they are created automatically by the FXMLLoader. This makes it hard to pass data from one scene to another scene.

Solution: Get the Controller object from the FXMLLoader, then pass parameters into the controller using an init method.

Example:
```
RunOptionDialogController runController;
runcController = loader.getController();
runcController.ini(runOptions, this);
```

Table Functionality in JavaFX

- The default table in JavaFX requires users to hit ENTER to commit each value they entered. This was not a natural behavior for the table in this software and could confuse users.

Solution: Design a custom subclass of the TableView class that is named AcceptOnExitTableCell (class outline shown at right), which overrides the default table behavior such that user’s entries are stored regardless of whether they hit ENTER.

IV. Results (Improved UI with JavaFX)

This software development project was completed over the course of 12 months. The final GUI was composed of 440 lines of XML code, 1345 lines of Java code, and 163 lines of Java comments, spread across 9 source files.