CMPT 381 Assignment 2
Screen Design, MVC, Heuristic Evaluation

Due: Friday, February 14, 11:59pm (handin closes Sunday, February 16, 11:59pm)

Overview

In this assignment you will design a screen for a specific task, evaluate your design using a simplified heuristic evaluation, and build the UI in JavaFX using the MVC design pattern. Part 1 covers the screen design, Part 2 covers the heuristic evaluation, and Part 3 covers the development of the interface.

Part 1: screen design for a milkshake construction screen

Result for Part 1: a photo of your (hand-drawn) screen design, in PNG format.

Design a milkshake construction screen for the vending machine that allows a user to interactively build a milkshake and see multiple views of what they have built (both textual and graphical). The screen must support the following task elements (note that you do not have to consider any personas or other task descriptions for this assignment):

- The user builds a specification for a milkshake based on ice cream and various toppings; when they are satisfied with what they have built, they can have the milkshake created by the vending machine.
- The user can add and remove scoops of ice cream to the milkshake (up to eight scoops in total), in any of six flavours: chocolate, vanilla, strawberry, lemon, coffee, and mint. Any combination of flavours is allowed.
- The user can add and remove toppings to the milkshake (up to eight units of toppings in total). There are six types of toppings available: sprinkles, cherries, chocolate chips, whipped cream, coconut, and marshmallows. Any combination of toppings is allowed (including multiple units of any topping).
- Scoops of ice cream cost $1.00 each; toppings cost 50 cents per unit.
- At every interaction with the user, the system shows the current specification in several ways:
    - In the panels where the ice cream and the toppings are selected, the system shows how many scoops of each ice cream flavour have been chosen, and how many units of each kind of topping.
    - In a summary panel, the system shows a text summary of the specification as well as the total cost of the milkshake.
    - In a picture panel, the system shows a graphical representation of the milkshake specification, with graphical elements for each scoop of ice cream and each unit of toppings. The graphical elements must clearly indicate the specification (e.g., for users who cannot read the text summary).
- You do not have to handle payment or any other back-end processes of the vending machine.

Design a screen that allows users to carry out the task of milkshake construction as described above, and that is understandable, effective, efficient, and satisfying to use (i.e., has excellent usability). You may use any widgets or other graphical elements in your screen (limited, however, to what you will be able to build with JavaFX), and you may organize the task flow however you like, as long as the resulting UI is highly usable. Use any of the design methods that you have learned so far to come up with your screen design (however, you will only hand in the end product of your design process). Make use of the lecture material on design concepts and design heuristics as you think about different ways of supporting the construction task.

Part 2: Evaluate your design using simplified heuristic evaluation

Results for Part 2: a two-page written summary (in PDF format) of problems and issues found in your evaluation, organized by heuristic; and a photo of your (hand-drawn) revised screen design, in PNG format.

Carry out a heuristic-based inspection of your screen design before you build it in JavaFX. Following the methods described in class (also see the resources linked to the moodle), do a simplified heuristic evaluation using Nielsen's ten usability heuristics. You do **not** need to use multiple evaluators. For each heuristic, assess your design and write down problems and issues that you see in the UI. Note that we expect you to find problems with your design! It is much better to be clear and honest about deficiencies than it is to pretend that no problems exist – similarly, don't fix problems that you find and then re-start the heuristic evaluation so that you can pretend that there were no problems.

After completing the evaluation, use your results to redesign any problematic elements of the UI, and create a revised screen that solves as many of the problems as possible.

Part 3: build the milkshake construction system in JavaFX, using MVC

Result for Part 3: a zip archive of your IDEA project folder for your working JavaFX project

Build a JavaFX program that implements the screen you designed in Part 1. The system must be built using the Model-View-Controller pattern, with the following specifications:

- Model (class MilkshakeModel):
  - The model keeps track of the current milkshake specification (i.e., scoops of ice cream and units of toppings)
- Views (classes IceCreamView, ToppingsView, PictureView, and SummaryView):
  - The views show the current milkshake specification (in different ways, depending on the view)
  - The IceCreamView contains the controls for adding and removing scoops of ice cream from the milkshake, and shows how many scoops of each flavour have been chosen.
  - The ToppingsView contains the controls for adding and removing units of toppings from the milkshake, and shows how many units of each type of topping have been chosen.
  - The PictureView shows a graphical representation of the milkshake, with graphical objects for each scoop and flavour of ice cream, and for each unit and type of toppings. (Note that the graphical representation does not need to show what the resulting milkshake will actually look like)
  - The SummaryView shows a text summary of the milkshake specification and the total price for the milkshake.
  - All views should use colours, fonts, images, and graphics that are appropriate for the usage context (i.e., a touchscreen interface on a vending machine)
- Controller (class MilkshakeController):
  - The controller is the handler for all input events, and interprets those events to call appropriate methods in the model class to change the model.
  - Although the usage context is a touchscreen interface, your system should use mouse events rather than touch events (although the UI design should be appropriate for touch).
  - Any events that do not result in changes to the system should result in a message written to the console, stating what the event was.
- MVC:
  - Your application class should create the model, views, and controller, and should set up the components so that they have appropriate visibility to each other. Components should only have access to the other components if absolutely necessary (follow the guidelines given in lectures).
  - Note that you should **not** use the Controller.java class provided by the IDE when you start a JavaFX project; delete this file before you write your own controller class.

  - Communication between the model and the various views **must** use a publish-subscribe approach, as described in lectures. Your code should include a Java interface MilkshakeModelListener as part of the publish-subscribe mechanism.
- Layout:
  - Choose appropriate layout managers to translate your design to JavaFX. Your interface should follow the principles of screen design introduced in class, and should be visually clear, consistent, and understandable (note that your UI does not need to be fine art, but you should ensure that your colours, typography, images, and layout lead to a pleasant and successful experience for the user).
  - The initial size of the window should be 1200x800, and your screen should handle window resizing in an appropriate fashion: if the window is enlarged, the added space should be used appropriately for the task; if the window is reduced in size, the different views should do their best to adjust.
  - Yes, we have previously stated that the milkshake machine has a full-screen interface. We will bend this rule for this assignment to allow you to show off how well you can handle resizing :)

What to hand in

This assignment is to be done individually; each student will hand in an assignment.

- Part 1: a PNG file of your screen design (named something like abc123-cmpt381-assn2-screen-design.png).
- Part 2: a zip file with a PDF document (your summary of the heuristic evaluation) and a PNG file of your redesigned screen.
- Part 3: a zip file of your IDEA project folder and a readme.txt file that indicates exactly what the marker needs to do to run your code. (Systems for 381 should never require the marker to install external libraries, other than JavaFX).

Where to hand in

Hand in your three files (one PNG file and two zip files) to the link on the course moodle.

Evaluation

Marks will be given for: demonstrating that you can use the screen design principles to design an effective and usable design for the given task; for correctly carrying out the heuristic evaluation to identify problems in the design and allow a redesign of the screen; and for producing a working GUI that corresponds to your final design and that correctly implements the design using MVC.

Weighting of the parts in the overall grade: Part 1=25%, Part 2=25%, Part 3=50%.

Note that no late assignments will be allowed, and no extensions will be given, without medical reasons.