Human Computer Interaction

Calculator Lab I

Prof. Andrew D. Bagdanov

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Firenze
`andrew.bagdanov AT unifi.it`

November 2, 2016

Outline

Purpose

Purpose of this laboratory

- In this lab we will first continue the exercises we began last week.
- Afterwards (or for those who have already completed these exercises over the weekend), we will begin working on a mini-project.
- This mini-project will be used to illustrate the complexity of real GUI applications.
- We will continue to refine our implementations of this mini-project through to the end of the class.

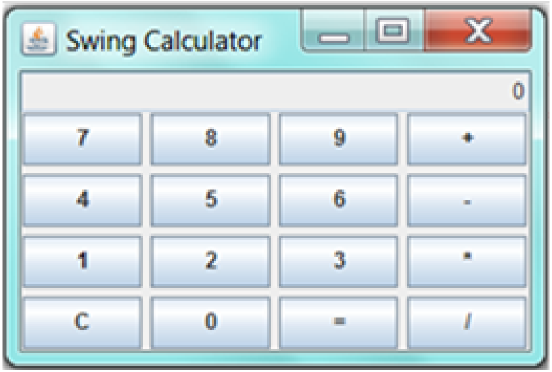
Mini-project (a calculator app)

Mini-project (introduction)

- In this mini-project you will implement a fully functional calculator application.
- We will begin by implementing only the button layouts.
- Then we will spend some time thinking about functionality, generality, and extensibility/maintainability.
- This project will naturally lead us to a discussion of styling views, abstracting data manipulation into a model, and regularizing control of the application.
- In other words, to the Model-View-Controller paradigm.

Mini-project step 1: button layouts

- Write a Kivy application that implements (at least) the following button layout:



- Please start your implementation in a new, dedicated project directory.
- We will (eventually) be splitting everything into modules as much as possible.

Mini-project step 1: button layouts

- To duplicate this type of layout, what should the root widget be?
- What should the children widgets be?
- Start writing. . .

Mini-project step 1.5:

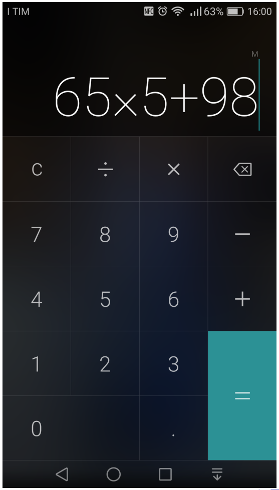
- OK, now you should have at least a basic calculator layout working.
- The application doesn't do anything, but at least the main view is there.
- This exercise is at least partially designed to illustrate how laborious it is to program the visual aspects of a GUI.
- Surely, there is a better way. . .
- Because, side effects of this style are verbosity and inflexibility.

Mini-project step 2: what events, what ops?

- Now let's think about how to make the application do something.
- Are there differences between the buttons, other than the obvious cosmetic ones?
- Take some time to think about what events need to be caught and reacted to.
- What about the underlying data representation for the calculator?
- How do calculators work?

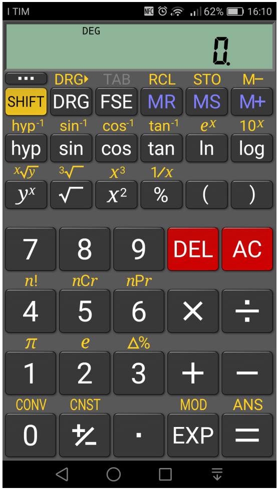
Mini-project step 3: extensibility?

- This is a pretty boring calculator application.
- However, note that the standard Android calculator isn't much more exciting:



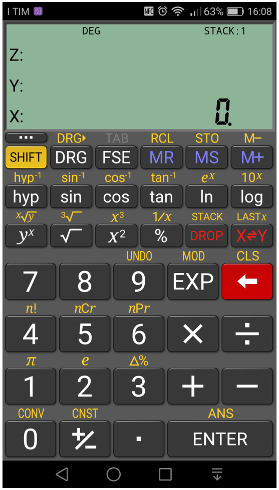
Mini-project step 3: extensibility?

- Here's a more featureful calculator app:



Mini-project step 3: extensibility?

- Here's a more featureful calculator app:



Mini-project step 3: extensibility?

- What if we want our calculator application to support multiple modes of operation (e.g. algebraic and RPN)?
- What if we want to skin our application to support multiple styles?
- Clearly we need better (and more flexible) abstractions for the view and for the model.
- We will see how to (at least partially) address these issues tomorrow when I will present the KV design language.
- This will let us factor out the static elements of the application view into a compact, declarative form.

Homework

Homework

Homework 10.1: Buttery smooth Emacs

This article on implementing double buffering in Emacs is entertaining and a good, funny rant about the differences between Terminal User Interfaces (TUIs) and Graphical User Interfaces (GUIs).

Homework 10.2: Your calculator application

Continue working on your implementation of the calculator mini-project. If you haven't already, implement event handlers to at least allow for numeric input and cancellation of current input.