Lab 3: Paper Prototypes to Mockups

Objectives

- Present and evaluate paper prototypes
- Revise design based on feedback from classmates
- Create a partially-functioning mockup in Python of your revised prototypes

Introduction and background

This past week, you drew conclusions from your user tests and proposed changes to the Carlpedia interface on the basis of those user tests and the course readings. You should have a paper prototype and presentation to make to the other students in the course to get some additional feedback on your proposed design.

This week in lab, you will be making the paper prototype presentations. We'll then discuss any generalizations that we can draw from the user tests, meet in groups to consider making changes to our prototypes, and consider how to program the prototypes in Python. By lab next week, you should have a program with a mock-up of your prototype to show to the class.

Your tasks

Peer evaluation of prototypes

Each group will present the results of their user tests and their prototypes to the class. The presentation should be thesis-driven: it should have a main point (the hypothesis), present relevant evidence (results from your user tests), and logical conclusions. As part of the logical conclusions, you will do a walk-through of the paper prototype you developed. You will use the doc camera to demonstrate what happens when a typical user interacts with your revised interface. (You should call up a volunteer from the audience to “drive” your prototype for this part. This will give you a better sense of what a typical user will do with your interface and give you better insights into what works and doesn’t work in your design.)

As you are presenting your results to the class, your classmates will be evaluating your design using the rubric from last week’s homework. You will see your peers’ feedback at the conclusion.

Discussion

- What design choices did you find most effective in the prototypes presented?
- What design choices did you like? Why?
- What design choices did you dislike? Why?

- Would these design choices be equally effective for all user populations?
- Would these design choices be equally effective for all tasks?
- Were there any design choices that were not well supported by evidence?

Design revision

You now have feedback from your classmates---the feedback you received while presenting your paper prototypes to your classmates, as well as the feedback from the previous discussion---as well as from your user tests. Your task now is to take this feedback and make modifications to your design.

As you consider the feedback of your classmates, you may also want to consider the following questions:

- Were certain tasks or interactions unclear to your classmates?
- Did you have to explain how to interact with your prototype at any point?
- Did your classmates attempt to interact with your prototype in unexpected or inappropriate ways?

From paper to mockup: realizing your prototype in Python

The next step in this process is creating a mockup in Python of your paper prototype. The mockup will not have the full functionality of the Carlpedia page(s), but it will demonstrate the new and improved interface and workflow you are proposing.

In class this week, we've worked with a number of examples of object-oriented graphics programming. By now you should be fairly comfortable combining graphical objects (circles, rectangles, lines, etc) to form more complex graphics. The same principles apply to designing mockups.

If you look closely at a web page, you can decompose the elements on that page to graphics objects: text boxes, rectangles, etc. Further, you can classify them into higher-order objects: buttons, menus, etc. Creating a mockup of a web page, then, is no different from creating graphics from simple graphical objects. You identify the elements that make up the page, and then create objects corresponding to these elements and place them on the "page" (in this case, the graphics window) in the desired locations.

The graphics library we're using includes objects like clickable buttons, so you can actually program some rudimentary actions into your page as well.

Here are some activities for you to try, to get you used to making mockups in Python:

- Write a program that places 5 evenly-spaced buttons within a window of size (100, 300). BONUS: Use a for loop to create and draw the buttons.
- Write functions to create other user interface elements. The function should take in the center position (x, y) of the element as well as any other text/size information that controls the appearance of the element.
- Write a program that generates a simple mockup of a simple web page. The web page should contain a menu bar, a search bar, some text, and a button to take the user to the next page. Each element should be generated by functions that you provide.

Homework

Your assignment this week is to turn your paper prototype into a working mockup of the Carlpedia site.

Your program should do/contain the following:

- One or more graphical objects representing items on the wiki home page. These include boxes, text, columns, icons, etc. Your program should instantiate as many objects as necessary of the appropriate type, assigning descriptive variable names to each object and calling appropriate methods on them.
- One function per section of the home page. For instance, if your layout has a title bar, a menu down the left side, and three information boxes, then you should write a separate function to generate each of these. Your functions should all use appropriate input parameters and return appropriate values where applicable.
- A main method that instantiates the main wiki window, calls each of the functions to populate the window, and displays the wiki layout.
- (optional) Functions to handle the button clicks, menu selections, etc. from your paper prototype.
- At a minimum, comments for each function, including the main method, that adequately describe what the function does.

Your prototype is worth 28 points and will be evaluated on the following criteria:

- **Execution/Correctness: 5 points**
    - Program runs to completion with no errors
    - Interface represents the prototype faithfully and correctly
    - Interactions with the interface (button presses, menu selections, etc.) result in the correct behavior (changes to the interface, actions, etc)
- **Python elements: 10 points**
    - Graphical objects
        - Dot notation used appropriately to call actions on the objects
        - Objects are created correctly (constructor call) with the appropriate initial state information
        - Appropriate information is passed to the objects when executing actions
        - Return values from object actions are stored and handled appropriately
        - The number and type of objects is appropriate
        - Objects are combined in appropriate ways to represent the design of the page
        - Objects interact in appropriate ways to represent the functionality of the page
    - Functions
        - The input parameters are appropriately chosen, in terms of number and “type”
        - Return values are used when appropriate
        - One task per function
        - No prompting for input w/in function or printing out result w/in function (maintain the “wall” between the function and the main program)
    - Other (loops, conditionals, calculations, etc): used appropriately and correctly

- **Style: 5 points**
  - Variable and function names clearly specify the data stored and/or task executed
  - Comments clearly indicate what the program is doing
  - Comments use the correct format/style
  - Whitespace logically separates the different tasks of the program, to aid readability
  - Similar statements are logically grouped together
  - Functions are clearly separate from “main” code
  - Overall, the program is readable and easy to follow
- **User-centric design: 8 points**
  - Interface functionality is clearly implied by the configuration of graphical objects
  - All expected elements of the interface are clearly represented in the display
  - Colors, fonts, etc are appropriate for the interface and imply hierarchies/ functionality
  - Interface functionality clearly follows prescribed metaphors, etc.

Turn in your Carlpedia mockup on Moodle. Only one person in each group needs to turn in the code. If you have multiple files (for example, your mockup uses image files), please zip them up into a single file named mockup.zip.

Writing

When you get your paper back, you should work on revising the paper based on comments from Mija, peer feedback on your prototype presentation, and your own group's re-reading of the paper. After you revise the paper and before you resubmit it to Mija, each group needs to meet with Nick about the choices they made in their revision. The final draft is due Wed, Oct. 12.