User Interfaces: Paper Prototyping & User Testing

the extreme basics.

User Interfaces are important(!?)

- Apple versus PC … what's the difference?

- Apple maps versus Google maps … what's the difference?

- Racket versus Java … what's the difference?

- Programming in an IDE versus programming in a simple text editor … what's the difference?

Human Capabilities (of interest to us)

- We are great at pattern recognition
- We automatically try to organise things to look for clues
- We find recognition easier than recall
    - (close your eyes and try to remember the order in which methods are implemented in your class. Or try to remember the order in which menus are laid out in Eclipse/IntelliJ)
- We are great at learning things *incrementally* rather than all at once
- We like doing tasks, not learning things abstractly.

Affordance Theory (Gibson)

Summary: Affordance theory states that the world is perceived not only in terms of object shapes and spatial relationships but also in terms of object possibilities for action (affordances) — perception drives action.

Originators: J. J. Gibson (1904-1979) http://www.learning-theories.com/affordance-theory-gibson.html

- Affordances are "action possibilities" latent in the environment, objectively measurable and independent of the individual's ability to recognize them, but always in relation to agents and therefore dependent on their capabilities.

http://en.wikipedia.org/wiki/Affordance

User Centred Design

http://en.wikipedia.org/wiki/The_Design_of_Everyday_Things

- Brought Affordances into the world of design
- Introduced *User Centred Design:*
    - simplifying the structure of tasks,
    - making things visible,
    - getting the mapping right,
    - exploiting the powers of constraint,
    - designing for error,
    - explaining affordances

Nielsen's Principles

- **Match the Real World where possible**
- **Consistency and Standards**
    - (don't introduce new user interface elements — unless the technology is totally new such as the introduction of swiping/pinching for the iPad)
- **Help and Documentation**
    - (should be searchable)
- **User Control and Freedom**
    - (always allow users a way out of a multi-stage process! Support undo/redo)
- **Visibility of System Status**
    - (don't have a blank screen with no indicator of something going on in the background)
- **Flexibility and Efficiency of Use**
    - (remember that expert users want to do things *fast* — Keyboard shortcuts can allow this!)
- **Error Prevention**
    - (give lots of warnings if someone is trying to do something that will be unrecoverable)
- **Recognition is better than Recall**
    - (don't ask users to memorise things about the UI, but give them recognisable pathways to find things)
- **Make error messages helpful**
    - (don't be cryptic — consider how annoying some JVM errors are!)
- **Aesthetic and Minimalist Design**
    - (this helps focuses a user's attention on what they should be doing)

Achieving Usability

- User Testing
- Field Studies
- **Prototyping**
    - Paper Prototyping (we will do this!!)
    - Code Prototyping

The basics (and way overgeneralising)

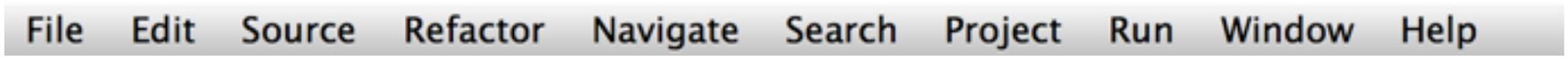
- **Functions**
- **Choices**
- **Data Entry**
- **Data Presentation**

Functions

- Use **buttons** for single, independent actions that are relevant to the current screen

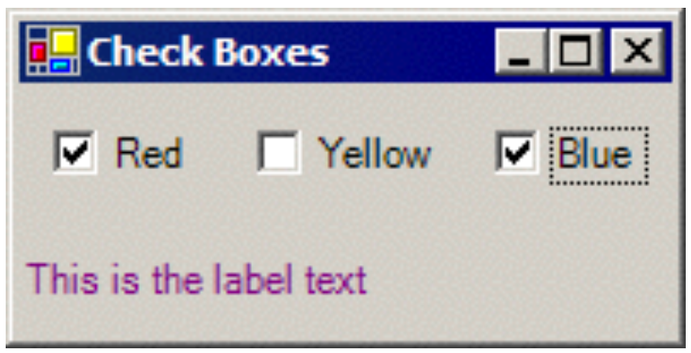
- Use **toolbars** for frequent actions

- Use **menus** for infrequent actions



Choices

- Use **check boxes** for independent choices



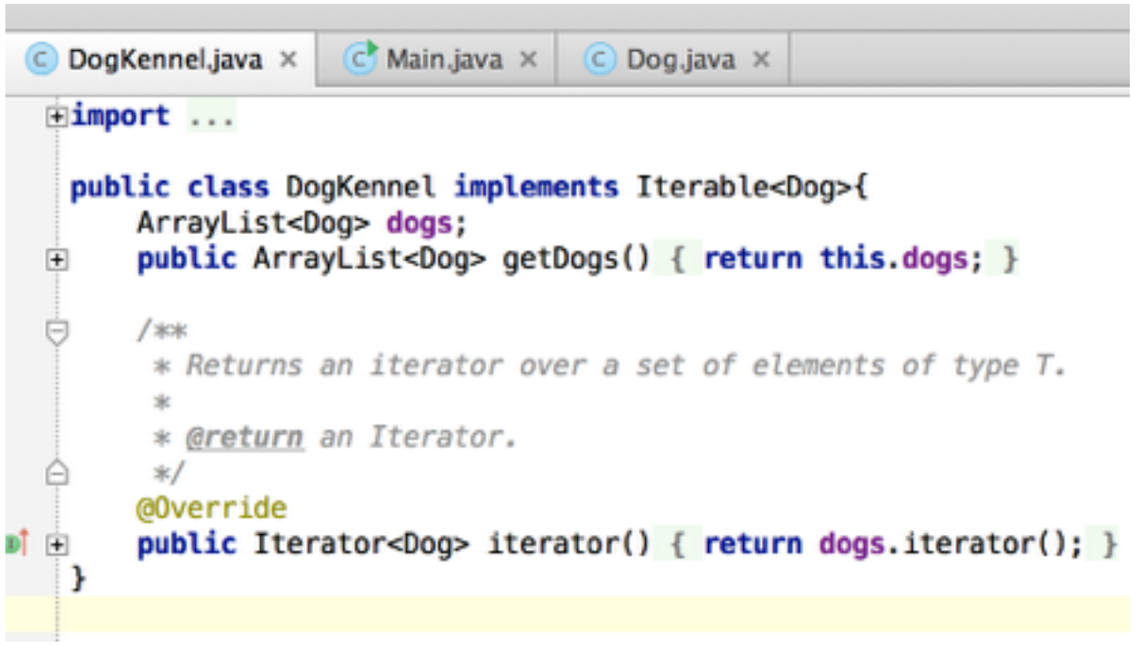
- Use **radio buttons** for dependent options

Data Entry

- use **text fields** (usually with a label) when the user may type in anything they want
- use **lists** when there are many fixed choices (too many for radio buttons to be practical) and you want all choices visible on screen at once
- use **combo boxes** when there are many fixed choices, but you don't want to take up screen real estate by showing them all at once
- use a **slider or spinner** for a numeric value

Data Presentation

- use a **tabbed pane** when there are many screens that the user may want to switch between at any moment

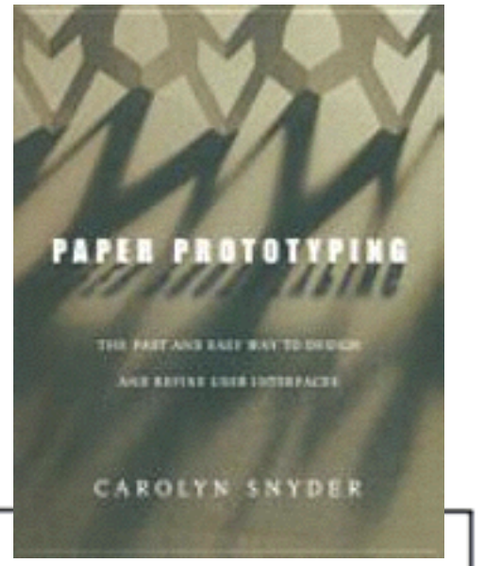


- use **dialog boxes** or **option panes** to present temporary screens or options

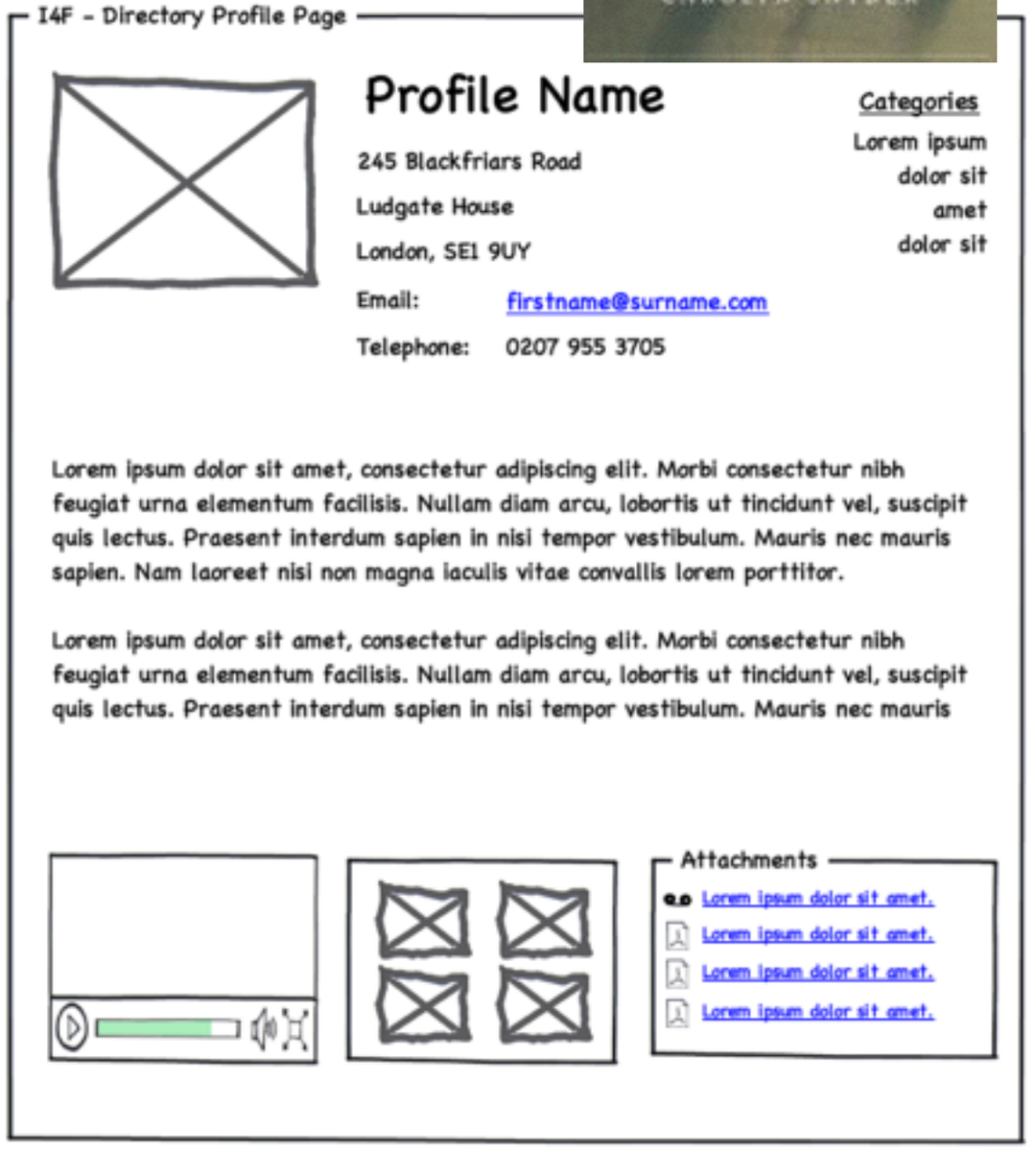
Prototyping

- Creating a scaled-down or incomplete version of a system to demonstrate or test aspects of it
- Reasons to do prototyping:
  - aids UI design
  - provides basis for testing
  - team-building
  - allows interaction with user to ensure satisfaction

Super Lightweight Prototyping Wireframes/Mockups



- much faster to create than code
- easier to change than code
- encourages feedback, since it feels less permanent or final
- focuses on big things vs. small (like the font)
- implementation neutral
- can be done by non-technical people
- PP shows us “what” is in the UI, but also “how” the user can achieve their goals in the UI; helps uncover requirements



http://www.balsamiq.com

User Testing a prototype

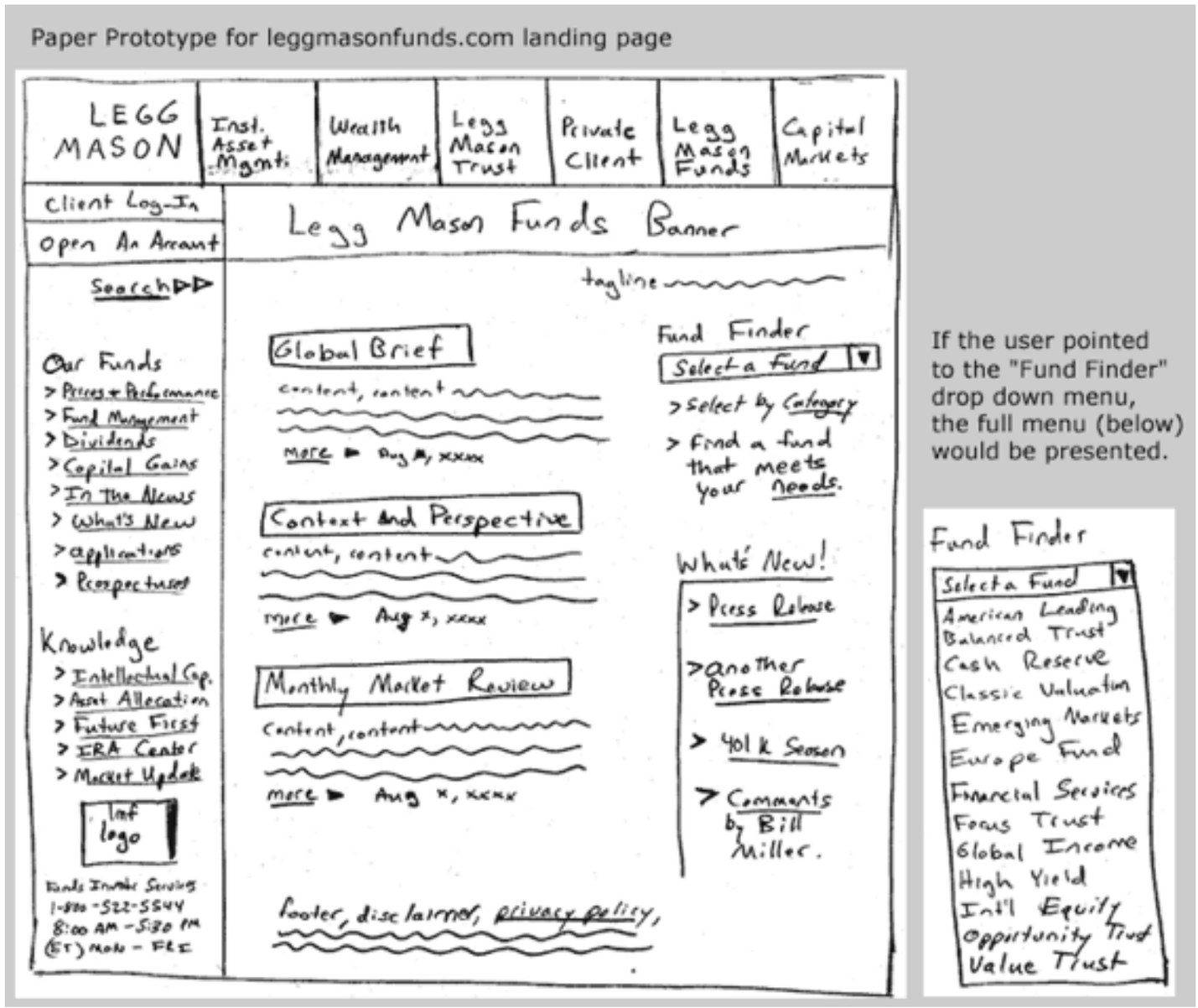
- Give the user an instruction that does not confound your test. This means:
    - Introduce the user to the main goal of the application.
    - Ask your user to complete a task that accomplishes a task — probably one that encompasses a user story.
    - Do not include the solution to *how* to do the task
    - Do not ask leading questions like “do you like this?”
    - Do not correct the user while they are attempting to follow the instruction — record what they do but do not critique or comment.
    - Generally, do not ask the same user to do the same instruction with a new version of the tool — find a new user to test.

Do say: “Please use the app to find a way from location 1 to location 2”

Don’t say: “Press location 1, then press location 2, then hit the *route* button”

How to make a paper prototype

- Draw each window on paper
- Show the user the window that would result from their action



LightWeight Prototyping Example: Vampire Avoidance System

- **Develop UI from user story:**
    - As a user, I want to be able to tap a destination on a map (that shows the locations of vampires) and see a route that gets me safely home, as well as a safety rating for the route.
    - (I will demo how to do the UI development)
- **1. Give USER INSTRUCTION:**
    - **You say:** *Hi, User! You are someone who lives in a town filled with vampires. You want a safe route home. Here is the app for that! Please interact with our application to try to find a safe route home. Please think aloud while you interact with the application.*
- **2. Take Notes:** Note what the user does, and especially note where the user becomes confused (when they ask a question to themselves, like "Hmm … where to find…") and plan to fix that in your next UI iteration.
- **3. If the user is completely lost,** send the user away, fix the UI, recruit another user, and then test again.

Summary

- Test your user interface *even before you build it*
- Follow good design principles
- A bad user interface will hurt your product's success!
- Now you try to make a lightweight prototype and do some validation!

Practice Activity…

- **PHASE 1: DESIGN THE UI**
  - Look at your system
  - Choose ONE user story
  - Create mock-ups for the windows in the user story using sticky notes and pieces of paper
  - Write a user instruction, and walk through what you expect the user to do given that instruction, using your paper UI.
- **PHASE 2: ROLE PLAYING!**
  - Send one person from your group to another group (to be a user)
- **PHASE 3: USER TESTING**
  1. Give your visitor the user instruction.
  2. The user MUST ONLY use the paper UI to carry out the instruction, though they should “think aloud” about what they are doing.
  3. If the user expresses confusion, make a note.
  4. If the user can’t figure out how to use the UI to carry out the instruction, ask them what they thought they should have done and make a note.
  5. ask the user to go back to their original group, and send you a different user (you don’t want them to simply have learned the interface so you can’t just re-ask the same person)
  6. Go back to step 1 - repeat until the user can work through the whole user story without any confusion.