A user user manual (how to build software that works the way your users think)

Learn how to build software that works the way users' brains work. Based on psychological principles and the results of hundreds of hours of user testing, this class shows you why users behave the way they do, and what you can do to make sure your software supports their mental processes.

People are strange. The way they interact with our software is often unbelievable to us. Why can't they just click the damn button? In reality, much of users' "random" behavior is actually very predictable when we understand how people think. Once you are familiar with these thought processes, you can build software that appeals better to your users.

What you'll learn:

- How to structure your software so that it works the way people think
- The reason why some design elements "just work", and how to apply them consciously to your product
- How to write better UI text, dialogs and error messages
- How to get and retain users' attention
- The ingredients that make people love the software they use

Course outline:

- Cognitive concepts – how our brains work
  - Forgetfulness – why good software always remembers the milk
  - People can't do sums (some consequences)
  - Story telling and its place in software design
  - Why we always distract users at the worst time
  - Consistency, inconsistency and superstition
- Perception concepts – how we see the world
  - How some German psychologists told us all we need to know about UI design in the year 1900
  - Scanning versus reading: novelists need not apply
  - Using color as a signal (and colorblindness)
  - Visibility principles
- Physical concepts - Emulating the real world on a computer screen
  - Mapping controls to the task, to the real world, to expectations
  - Impatience and computer response times (tricks to make your software seem faster)
  - Keyboard versus mouse or touchscreen use
  - International differences
- Workflow concepts – rules for designing better applications
  - Smart defaults
  - Progressive disclosure
  - Control versus being led
  - Inductive interfaces
- Communicating through the UI
  - Dialog boxes and assistance text (help doesn't help)

    - Making people feel confident so they don’t need help
    - Error messages aren’t dialogs (but they should be)
    - An argument against “special” UI for novices
    - Communicating using graphics
- Designing delightful software interfaces
    - Attractive things work better
    - The phases of delight - tracking users’ experience over time
    - Fixing the little things
    - Creating big differences
    - Delight: unexpected out-of-context surprises