HCI: Figuring Stuff Out about Objects

Affordances, Signifiers and Mapping

Prof. Andrew D. Bagdanov

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

September 30, 2016

Outline

The complexity of modern devices

The nature of artificial things

- All artificial things are designed in some way, whether it is a chair, the layout of furniture in a room, or as complicated as the layout of electronic circuitry in handheld devices.
- Not all designed things involve physical structure: I designed this lecture, businesses have organizational structures that have been designed (perhaps informally).
- Though we have designed stuff since prehistoric times, the field of design is a relatively recent development.
- In the best of cases, the products should be delightful and enjoyable.
- This implies not only satisfying engineering requirements related to manufacturing and ergonomics.
- It also implies paying attention to the entire experience, including the aesthetics of form and the quality of interaction.

Three branches of design

The emerging field of design (at least the types of design relevant to this course) have branched into three major areas:

- Industrial design: creating and developing concepts and specifications that optimize the function, value, and appearance of products and systems for the mutual benefit of both user and manufacturer.
- Interaction design: focusing on how people interact with technology in order to enhance understanding of what can be done, what is happening, and what has just occurred. Interaction design draws upon principles of psychology, design, art, and emotion to ensure a positive, enjoyable experience.
- Experience design: designing products, processes, services, events, and environments with a focus placed on the quality and enjoyment of the total experience.

Note that none of these is a clearly defined field of study or practice. But the focus of the efforts does vary, with industrial designers emphasizing form and material, interactive designers emphasizing understandability and usability, and experience designers emphasizing the emotional impact.

Design and rigidity

- Design is, clearly, concerned with how things work.
- When done well, the result is pleasurable; when done badly, frustration and irritation is the only result.
- When done badly, the products might be unusable, or they might force users to *to behave the way the product wishes rather than as we wish.*
- Machines are conceived and designed by humans, and by our standards are relatively limited in their ability to interact.
- Instead, they follow very simple and rigid rules, and if we get the rules wrong even slightly, the machine does what it is told, no matter how insensible and illogical.
- Contrast this with interactions between persons: we are imaginative and have common sense; i.e. a lot of common, valuable knowledge at our disposal.
- Machines have no leeway or common sense, and often the rules followed by a machine are *known only by the machine and its designers.*

A useful analogy

- Imagine a complex human-human interaction, like selling metro tickets.
- Now imagine that you have to write an explicit list of rules for an untrained person to follow in order to sell metro tickets.
- These rules must specify exactly what to do in all cases: monthly passes, senior discounts, reimbursements, etc.
- Can you imagine a foolproof "script" you could write for such an interaction?
- We humans are able to apply our human reasoning and improvise and generalize from incomplete plans and instructions; machines cannot.
- This is exactly what happens often in bureaucratic situations where functionaries of the system do not (or cannot) apply reason to interpreting situations and instructions.

Stop thinking like an engineer

- Some reasons for the deficiencies in human-machine come from the limitations of today's technology.
- These can be self-imposed restrictions, often to keep down costs.
- But more often, these problems come from a lack of understanding of key design principles. Why?
- Because much design is done by engineers who are technology experts, but are limited in their understanding of people.
- Engineers can often make the mistake of thinking that logical explanation is sufficient: "If people read the instructions, everything would be all right."
- We are trained to think logically, and we come to believe that all people think this way – *and we design our machines accordingly.*
- When people have trouble, we become upset: "What is this guy doing?! Why is he doing that?!"
- The problem most is that they are too logical, while we must accept human behavior the way it is, not the way we would wish it to be.

Human-centered design

- People are frustrated with everyday things: everyday life sometimes seems like a never-ending fight against confusion, errors, frustration, and a continual cycle of maintaining our belongings.
- Perhaps a solution is to shift focus of design away from the objects we design and towards the humans who will use them.
- Human-centered design (HCD) is an approach that puts human needs, capabilities, and behavior first, then designs to accommodate them.
- Good design starts with an understanding of psychology and technology and requires good communication, especially from machine to person, indicating what actions are possible, what is happening, and what is about to happen.
- Communication is especially important when things go wrong.
- Good design becomes essential, however, when things start going wrong and the user must be informed of what and why (i.e. communication via feedback).

A brief aside

- I am a lifelong and passionate Unix user (it's been more than 20 years since I used a Windows machine daily).
- There's a story (perhaps apocryphal) that Ken Thompson (Unix co-inventor) was talking with one of the core VMS developers about error recovery.
- Aside to the aside: VMS was a competing commercial OS of the day. The name of this VMS developer is lost to history, but we'll call him Mr. VMS.
- The conversation went something like this:
  - Mr. VMS: Hell, I estimate that more than 75% of the code in the VMS kernel is directly related to error recovery.
  - Ken Thompson: That seems like a lot.
  - Mr. VMS: Really, what do you guys do?
  - Ken Thompson: Well, we have this function called `panic()`...

HCD and user experience

- Great designers produce pleasurable experiences (a word engineers tend not to like because it is subjective).
- However, let's ask ourselves (engineers, all) about our favorite motorcycle or keyboard or user interface.
- Speaking for myself, I can discuss at length the experience of my favorite keyboard: the feel, and response, the sound of clicking keys.
- This is the wonderful experiences of good design.

Affordances

Gauging potentialities

- Our world is filled with objects, many natural, the rest artificial.
- Every day we encounter thousands of objects, many of them new to us, and though some them are similar to ones we already know, many are unique.
- How do we manage to figure out how to use these new, unique objects?
- The term affordance refers to the relationship between a physical object and a person.
- Important: an affordance is a *relationship* between the properties of an object and the capabilities of the user that determine how the object could be used.
- A chair *affords* support and, therefore, *affords* sitting.
- Most chairs can also be carried by a single person (they afford lifting), but some can only be lifted by a strong person or by a team of people.
- So, the existence of an affordance depends on both the object and the interacting user.

Affordances are jointly determined

- This bears repeating and reflection: the presence of an affordance is jointly determined by the qualities of the object and the abilities of the interacting user.
- This relational definition of affordance can be tricky because we are used to thinking that properties are associated with objects.
- But affordance is not a property, an affordance is a relationship.
- Whether an affordance exists depends upon the properties of both the object and the agent.
- An affordance can be seen as a perceived action possibility.
- As such, its existence clearly depends on both the object and the user perceiving it.

Perceivable affordances

- Glass affords transparency, and at the same time, its physical structure blocks the passage of most physical objects.
- So, glass affords both "seeing through" and "support", but not the passage of physical objects (this "prevention" of interaction can be called an anti-affordance).
- To be effective, affordances and anti-affordances must to be discoverable, which is difficult for glass.
- We like glass because of its relative invisibility, but this aspect also hides its anti-affordance of blocking passage.
- As a result, birds (and people) often try to pass through windows.
- Affordances exist even if they are not visible, but for designers, their visibility is critical: visible affordances provide strong clues to the operations of things.
- If an affordance or anti-affordance cannot be perceived, some means of signaling its presence is required: this we call a signifier, which we will discuss later.

A historical perspective

James Gibson's Affordances

- Gibson introduced affordances in 1977 to cognitive psychology, and defined them as all "action possibilities" latent in the environment.
- These should be measurable and independent of the ability to recognize them, but always in relation to agent's capabilities.

Don Norman's Drunken Appropriation

- Don Norman, in the 1980s, appropriated the term affordances to mean something slightly different.
- (Don Norman always says that he came to his definition after many beer-fueled sessions of arguing with Gibson himself)
- Anyway, Don Norman defined affordances to mean just those action possibilities readily perceivable by an actor.
- This is a definition much more amenable to our purposes (i.e. the design of HCI).

From affordances to signifiers

- But the reality is that affordances exist even if they are not visible.
- For designers, however, their visibility is crucial because visible affordances provide strong clues about what it possible, about how things can be manipulated.
- Perceived affordances help users figure out what actions are possible – without the need for labels or instructions.
- Affordances signal in some way the possibilities of interaction to the user.
- These factors we will call signifiers, and they may be explicit or implicit (or somewhere in between).

Affordances versus Signifiers

- Many people find the concept of affordances difficult to grasp because they are relationships and not properties.
- As designers, we want to deal with fixed properties: we want to "put this affordance there, and that other affordance there."
- But, as a relationship we can't really do this: affordances are not inherent in a single object or interface, but rather are complex relationships between that object or interface and the user.
- After Don Norman first introduced the term affordance to the design world, he found designers re-appropriating the term to mean something wholly inherent in the designed object.
- If you are communicating what is possible or how to do something, you are signifying the existence of an affordance – but this is not the affordance itself.
- Affordances determine what actions are possible, while signifiers communicate where or how the action should take place.

A tangible example of affordances

Moodle and Github login examples

Signifiers

Signposts

- People need some way of understanding the objects they wish to use, some sign of what it is for, what is happening, and what the alternative actions are.
- People search for clues, it is anything that might signify meaningful information that is important.
- Designers need to provide these clues: what people need, and what designers must provide, are signifiers.
- Good design requires good communication of the purpose, structure, and operation of the device to the people who use it. That is the role of the signifier.
- Signifiers can be deliberate and intentional, such as the sign PUSH on a door.
- But they may also be accidental and unintentional like the presence or absence of people waiting at a train station to determine whether we have missed the train.

Signifiers are needed

- No, signifiers are desperately needed:

Perceived affordances as signifiers

- Affordances represent the possibilities in the world for how an agent (a person, animal, or machine) can interact with something.
- Sometimes they are perceivable and these we call perceived affordances.
- Signifiers, however, are signals: some are signs, labels, and drawings placed in the world.
- Signs like "push," "pull," or "exit" on doors, or arrows and diagrams indicating what is to be acted upon or in which direction to gesture, or other instructions, are all signifiers.
- Some signifiers are simply the perceived affordances, such as the handle of a door or the physical structure of a switch.
- Some perceived affordances may not be real: they may look like doors or places to push, or an impediment to entry, when in fact they are not.
- Note the difference between an actual affordance and a perceived one.

A perceived affordance

- The handle tells us to push or pull.
- Perception of affordances can have strong cultural constraints

To summarize

- Affordances are the possible interactions between people and the environment – some are perceivable, others are not.
- Perceived affordances often act as signifiers, but they can be ambiguous.
- Signifiers signal things, in particular what actions are possible and how they should be done.
- Signifiers must be perceivable, else they fail to function.

Mapping

Physical and virtual space

- Mapping is a technical term borrowed from mathematics which means the relationship between the elements of two sets of things.
- Suppose there are many lights in the ceiling of a classroom or auditorium and a row of light switches on the wall at the front of the room.
- The mapping of switches to lights specifies which switch controls which light.
- Or even better: the switch to control the projector screen.
- Mappings like these create implicit or explicit correspondences between controls (virtual or physical) and the physical world.

Mapping in user interfaces

- Mapping is an important concept in the design and layout of controls and displays.
- When mapping uses spatial correspondence between the layout of the controls and the devices being controlled, it is easy to determine how to use them:
  - Example: to steer a car, we rotate the steering wheel clockwise to turn right – the top of the wheel moves in the same direction as the car, this is the mapping. Other choices could have been made: In early cars, steering was controlled by a variety of devices, including tillers, handlebars, and reins.
  - Example: bulldozers and military tanks have two tracks instead of wheels and use separate controls for the speed and direction of each. To turn right, the left track is increased in speed, while the right track is slowed or even reversed. This mapping is more complex, but still discoverable.

Natural mapping

- Natural mapping takes advantage of spatial analogies and leads to immediate understanding.
- In such cases we say that the functioning of the device/interface is discoverable through use.
- For example, to move an object up, move the control up; to make it easy to determine which control works which light in a large room or auditorium, arrange the controls in the same pattern as the lights.
- Some natural mappings are cultural or biological, as in the universal standard that moving the hand up signifies more, moving it down signifies less.
- Other natural mappings follow from the principles of perception and allow for the natural grouping of controls and feedback.
- Groupings and proximity are important principles from Gestalt psychology that can be used to map controls to function: related controls should be grouped together, and controls should be close to the item being controlled.

Natural mapping and proxies

- A device is easy to use when the set of possible actions is visible (i.e. when affordances are perceived and real), and when the controls and displays exploit natural mappings.

On our way

- We are (slowly) moving away from abstract conceptualizations of interaction with everyday objects towards a concrete philosophy of design.
- We can already start trying to formalize how we should think about designing human computer interactions:
  1. Identify the affordances that should be present in the device or object or system.
  2. Ensure that affordances are perceived correctly, and ensure that all perceived affordances are not false affordances.
  3. When affordances are not evident, add signifiers to signal their presence.
  4. Use natural mapping between controls and outcomes to ensure discoverability of function.
  5. Repeat.
- We are still missing a few things: feedback for the user to indicate what has gone wrong (or right), and conceptual models in which to interpret affordances and mappings.

A preview of next week

Why Python?

- Next week we will take a break from design and get our hands dirty with real Graphical User Interface (GUI) implementation.
- As I mentioned in the first lecture, we will be using a python-based toolkit for experimenting with and learning about graphical user interface implementation.
- On might (and reasonably should) ask why I chose Python for this purpose.
- That is why not Java or (ugh) C++?
- Well, first of all I didn't choose python because it is some sort of industry standard for building user interfaces.
- I also didn't choose it because it is super efficient in terms of execution times.
- Well, then, why?

A comparison

Java GUI *Hello World*

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class HelloWorldGUI2 {

   private static class HelloWorldDisplay extends JPanel {
      public void paintComponent(Graphics g) {
         super.paintComponent(g);
         g.drawString( "Hello World!", 20, 30 );
      }
   }

   public static void main(String[] args) {
      HelloWorldDisplay displayPanel = new HelloWorldDisplay();
      JButton okButton = new JButton("OK");

      JPanel content = new JPanel();
      content.setLayout(new BorderLayout());
      content.add(displayPanel, BorderLayout.CENTER);
      content.add(okButton, BorderLayout.SOUTH);

      JFrame window = new JFrame("GUI Test");
      window.setContentPane(content);
      window.setSize(250,100);
      window.setLocation(100,100);
      window.setVisible(true);
   }
}
```

A comparison

Python GUI *Hello World*

```
import qt, sys

a = qt.QApplication(sys.argv)
w = qt.QPushButton("Hello World",None)
a.setMainWidget(w)
w.show()
a.exec_loop()
```

Another python GUI *Hello World*

```
import Tkinter as tki

root = tki.Tk()
w = tki.Label(root, text="Hello, world!")
w.pack()
root.mainloop()
```

And another one. . .

```
import kivy

class MyApp(kivy.app.App):
    def build(self):
        return kivy.uix.label.Label(text='Hello world')

MyApp().run()
```

Sleek and essential

- I chose python because it is a concise, agile, and dynamic programming language.
- Its concision allows us to strip away most of the boilerplate code required to implement GUIs other programming languages (I'm looking at *you* Java).
- This, in turn, will allow us to expose the essential elements of GUI programming and provide clearer insight into the paradigms unique to this world.
- Its agility and dynamism will serve us well when prototyping user interfaces because we can experiment with multiple, sophisticated functionalities without arduous refactoring of code.
- Finally, python is a rich, mature, and popular programming language with a vibrant ecosystem – all of which will serve us well in the course (and you, in the long term, as professionals).

A personal perspective

- Is this going to be a course on python programming?
- No, and that wouldn't really be appropriate or necessary.
- If you already know one (or better, two) high-level programming languages, learning the basics of another should be an exercise for a weekend.
- After next week you will have mastered the basics, and by the end of the course you will be, if not expert, at least intermediate-level python programmers.
- But in any case, even if you never write another python program in your lives it will have served its purpose: to allow us to illuminate the salient aspects of GUI programming without interference from the language itself.

If you want to get a head start

- We will be using the Anaconda python distribution:

  https://www.continuum.io/downloads

- It is a modern and complete python distribution with an excellent and comprehensive package manager.
- It also comes with a streamlined Integrated Development Environment (IDE) called Spyder (if you're into that sort of thing).
- Versions are available for OSX, Windows, and Linux.
- Some online resources if you want to get started:
  - Overview:
    http://thepythonguru.com/getting-started-with-python/
  - Interactive tutorial:
    http://www.learnpython.org/
  - Comprehensive:
    https://docs.python.org/2/tutorial/index.html

Homework

Homework

Exercise 2.1: Observing and not taking for granted

Think about and observe the interfaces to common websites you visit or of common applications you use. Identify the affordances offered by them, and determine whether they are perceived or not. Can you find examples of false affordances? What about signifiers and mapping?