Micro-survey

Creating success for yourself in CS464:

- Recall you have to have at least 70% on the exams to get a C in this class
- **Come to class prepared!** Notes for the week are posted the weekend before, including in-class activities and readings/guides needed to do understand the material and activities.
- Self-evaluations for class participation points:
  - On almost every eval, I wrote: examples? Detailed examples? Ex?
  - I’m looking for concrete examples on these, not high level abstractions. For example, on the question about what you specifically did:
    - “I helped narrow the vision to something that made more sense.”
    - Give me an example of what didn’t make sense, why it didn’t make sense, and an example of a specific change you suggested.
    - “Our initial vision included fixing every breakdown we had in the consolidated models like ‘I <user> don’t trust <any tool> what you are doing’. The problem is that we’re only looking at providing ‘glue’ between systems, and we can’t change the tools to provide more info that might build trust for the user. I suggested that we take this out of our vision, and concentrate on the 3 breakdowns in the consolidated sequence model we identified as the users tried to interact with their various tools. Then I helped re-write the vision to make these changes.”
HTAs

• Why HTAs?
  – Systematic way to progress to work re-design.
  – Enough detail to re-design work.

• Creating an HTA from the consolidated sequence model:
  – What should it show? How detailed?
  – If you’re not sure about a step (e.g. maybe it is too abstract or too detailed), put it in initially.
  – Use the ideas in slides 11 and 12 of the TaskAnalysisGuide.pdf to refine the tasks.
  – When you move into re-design, be prepared to tweak the HTA again – adding more detail or abstracting as needed to make the re-design straightforward.

“Syntax” of HTAs

0. Decide reports to create
1. Sort downloaded data by date
   1.1 Open downloaded data in Excel
   1.2 Select all data and all columns
   1.3 Choose Filter -> Custom Sort -> sort on date column newest to oldest
2. View previous reports
   2.1 Get out latest hard-copy version of each report
   2.2 Open latest version of each report on computer
3. Compare report data and downloaded data dates
   3.1 Look at each report section and check date listed for data against date in spreadsheet
   3.2 Note every report that has a section where newer data is available
4. Decide which reports used older data and need to be replaced
   4.1 Record the name of each report that has newer data to remember what reports need to be regenerated

PLANS
   { Plan 0: do 1-4 in numerical order
   Plan 1: do 1.1 - 1.3 in numerical order
   Plan 2: do either 2.1 or 2.2
   Plan 3: do 3.1 and then 3.2
   Plan 4: do 4.1 }
HTAs to a UED

• The process:
  1. Create an HTA of the existing task
  2. Create an HTA of the re-designed task
  3. Repeat 1 & 2 for all the tasks that comprise the work you are re-designing
  4. Create a UED that has places (“rooms” or “focus areas”) and functionality for all the re-designed HTAs

• Alternatively, create the UED places for the first re-designed HTA, then repeat 1 & 2 for each additional task and augment the UED as you go along

• HTA Validation: Do your re-designed HTAs provide what you specified in your vision statement?

UED

• What does the UED show?
  – places where user can work
  – a set of functions available in that place
  – links to other, related places

• Elements:
  – Focus areas (the places)
  – Links between focus areas
  – In each focus area:
    • Name
    • Purpose
    • Functions provided
    • Objects that are manipulated
  – UEDs have logical places within the system
    • Screens, views where work is done with NO h/w or s/w constraints

• UED Validation:
  – Does your UED provide the re-designed HTAs that you specified in your vision statement? Does it also provide the things in your vision that you said you were going to preserve?
“Syntax” of UEDs

<table>
<thead>
<tr>
<th>NAME of the place</th>
<th>Purpose: Choose old reports to update</th>
</tr>
</thead>
<tbody>
<tr>
<td>PURPOSE</td>
<td>Functions:</td>
</tr>
<tr>
<td></td>
<td>• Creates “list” of report sections with old and new data dates</td>
</tr>
<tr>
<td></td>
<td>• Tags reports with sections based on older data as proposed to be updated</td>
</tr>
<tr>
<td></td>
<td>• Allows user to select reports or confirm proposed reports to be updated</td>
</tr>
<tr>
<td></td>
<td>• Creates “list” of reports to create</td>
</tr>
<tr>
<td>FUNCTIONS</td>
<td>Links:</td>
</tr>
<tr>
<td></td>
<td>&gt; 17. Location of new data</td>
</tr>
<tr>
<td></td>
<td>&gt; 20. Location of existing reports</td>
</tr>
<tr>
<td></td>
<td>&gt; 22. Validate data</td>
</tr>
<tr>
<td>LINKS</td>
<td>Objects:</td>
</tr>
<tr>
<td></td>
<td>New downloaded data</td>
</tr>
<tr>
<td></td>
<td>Latest versions of existing reports</td>
</tr>
</tbody>
</table>

Making Design Concrete

Learning objectives:
1. Understand how to define your project’s design space.
2. Interaction design
3. Gestalt (today or Wednesday)

Materials originally created by Prof. Jamie Ruiz
Impact

• Good visual design can significantly reduce processing time by user

• Tullis (1981) redesigned screen for telephone line testing
  – 40% reduction in time to interpret display
  – 79 person years saved for every year of use

• Tullis (1984)
  – 5.5 sec vs. 3.2 sec average search times for lodging info screens

Design Space

• Design space is both conceptual:
  – full range of possibilities for addressing identified problem; infinite in scope

• And real:
  – mapping of one or two dimensions of the design
  – e.g. tangible representations

• Mapping helps organize and suggest possibilities
Making Design Concrete

- Design process requires lots of cognitive resources
- Make *tangible representations* of every design idea to cope with this complexity
  - Write everything down!
  - Sketch!
  - Create example prototypes!

Benefits of Tangible Representations

- Offload cognition into environment
- Reflect on and evaluate the design
- Gather feedback from stakeholders
- Discover unintended side effects (positive/negative) of the design idea
Design Tools

• Low-fidelity prototypes (paper-based)
• High-fidelity prototypes (functional mock-ups)
• Scenarios & Storyboards
• Sketches – Read Chapter 1 of the sketching book and begin practicing the 10 plus 10 method: *Sketching User Experiences* book available on Canvas in the Files section.

Why prototype?

• Faster, cheaper than building functional system
  – One bug in implementation can wreck evaluation
  – Broader design palette than computer-based tools
• Evaluation
  – Roughness of system encourages “right” level of evaluation
  – Can iterate on-the-fly
  – Palm Pilot, Laser Printer, Dynabook evaluated this way
• Communication
  – More clearly communicate intents
Types of Prototypes

Horizontal vs. Vertical:
- **Horizontal**
  - Broad overview of system at cost of great detail
  - Built from UEDs (more on this Wed)
  - Example: Website’s overall site design
- **Vertical**
  - Detailed rendition of one small part of system
  - Example: Detailed page design for one web page

Low Fidelity vs High Fidelity
- **Low fidelity**
  - Extremely coarse approximation of system using common physical materials
  - Intent is to get primary ideas across
  - Details follow
- **High fidelity**
  - Systems nearly identical to proposed production version

Source: DENMM: An Informal Web Site Design Tool Inspired by CS489 / CS698
Observations of Practice (Newman et al, 2003)
Low Fidelity Prototyping

Paper Prototyping

• Start with blank sheets of paper
• As design starts to solidify, sketch unvarying system states on piece of paper / cardboard
• Use Post-Its to represent dynamic content
  – Drop-down menus
  – Views (e.g., graphs) that change
  – Drag-n-drop items
• Can also use tracing paper, transparencies for dynamic content
Wizard-of-Oz

• Low-fidelity prototypes can be brought to life
• Person fills in for computation
• Have different system states already built
  – Or build on-the-fly based on user demands
Designing the Interaction

Goals:
• To engineer the interaction with the system from the user’s perspective
  – Input/Output
  – Guide how the user will use the system
• Tools
  – Scenarios and storyboards
  – Mental models of interaction

Scenarios
• Description of interaction with software
  – Goals, Expectations, Actions, and Relations
• Helps user understand how software will be used
• Need appropriate level of detail for design stage
  – During early stage evaluation:
    • Are goals and action/reactions reasonable (grounded in reality)
  – During late stage evaluation:
    • A set of tasks that users can perform with software
• Create alternatives to explore the design space, errors, worst-case scenarios, etc.
Scenario example – text only

Tom presses the on button on his smartphone and is presented with a screen where he can select his username and input his password. After logging in, he is presented with an alphabetically sorted list of application icons. He clicks on “Inventory Management.” Pointing the smartphone camera at the box on the shelf, he presses the “scan” button on the screen to scan the QR-code on the side of the box into the system.

Scenario example - storyboards

- Makes scenario real
- Shows people, words, screenshots, whatever is appropriate
Designing interactive systems

- What Constitutes a Good Design?
- What Constitutes a Bad Design?

In reality, it is very difficult to describe why designs are good, but very easy to say what makes a bad design.

Conceptual Models and Design: Mismatches cause “gulfs of execution and evaluation” for the user.

Gulfs of Execution and Evaluation

Conceptual Models:
Thermostat? Car Heater? Refrigerator control?
Moving Towards Good Design: UI Design Principles

- **Affordances**
  - Perceived and actual properties of a thing that determine how it can be used (e.g. turn knobs, push buttons)
- **Mapping** (take advantage of physical analogies/cultural standards)
- **Constraints**
  - Physical (e.g. only one way to connect, disabled buttons/menus)
  - Semantic Constraints (rely on knowledge of the situation and world)
  - Logical (e.g. My Documents, My Pictures, etc.)
  - Cultural (e.g. Next ➔, Prev ←)
- **Visibility/Feedback**
  - Provide some mechanism to show:
    - Input is received
    - When system is doing something
    - The outcome of an action
  - Minimizes Gulf of Evaluation
- **Consistency** (allows users to leverage control from familiar onto new)
- **Metaphors**
Gestalt Principles

“In anything at all, perfection is finally attained not when there is no longer anything to add, but when there is no longer anything to take away”

Antoine de Saint Exupery

Simplicity

• Simple designs present the minimum amount of information to achieve maximum effect

• Simplicity leads to quickly recognized and understood functionality
  — Less information == less time to process

• Simplicity can also aid recall
  — Less to remember
Achieving Simplicity

- Reduce, reduce, reduce
- Reduce some more
- Reduce until it hurts
- Regularize
Reduce!

• For every visual element (or interactive element)...

• ...Ask yourself if you can remove the element and still have a functional, understandable system/design

• Good candidates for removal
  – Lines used to segment areas
  – Bounding boxes
  – Gratuitous graphics

• Consider how many people will need the visual information over the long-term

Regularize

• Regularization and repetition can produce simplicity
• We see patterns and chunk them into a single unit
• Makes things more predictable
• Increases ability to scan

Regularize across:
  – Size
  – Color
  – Line weight
  – Alignment
  – Shape
  – Texture
  – Orientation
  – Spacing

Example: Keyboard
Irregularity

• With regularity, irregularity becomes meaningful

• **Irregularities draw attention!!!**

• People will ascribe meaning to irregularity, even if you did not intend there to be any
  – Avoid accidental irregularities!

Organization and Structure

• Structure doesn’t happen naturally
  – Explicitly planned, designed

• People naturally try to find order and structure, even if none was intentionally designed

• Use Gestalt principles to create structure...
Achieving Organization and Structure

• Grouping
  – Apply Gestalt principles to create groups of similar or related items instead of explicit structure

• Hierarchy
  – guide viewer, to allow scanning of information
  – according to intended reading/viewing sequence
  – adjust properties such as size, position, spacing, white space...

• Relationship
  – Establish relationships between elements by using position, size, value (color, shape, etc.)
  – Alignment is very effective tool at creating relationships
  – Similarity of form also effective

• Balance
  – Want to create visually stable composition, similar to physical balance
  – Stability achieved by manipulating properties such as position, size, hue, form
  – Symmetric layouts achieve balance naturally

Common Errors

• Haphazard layout

• Proximity not taken into account when laying out components in interface

• Unclear hierarchy

• Bounding boxes creating visual clutter, competing for attention
  – Use negative space (white space) instead

Testing it out...
  – Use the squint test...
  – Mimics early portion of visual recognition system
Wireframes

- Wireframes allow you to explore basic layout and visual composition of interface at high level

- Focus on functional areas and user’s flow through the visual interface
  - What functionality, where? What tools?
  - What information will user need?

- Draw and label boxes indicating different portions of interface
  - Don’t need to provide actual contents of boxes
  - Can use “squiggly” lines for text if want to add weight

Balsamiq

- From the webpage: “Balsamiq Mockups is a rapid wireframing tool that helps you Work Faster & Smarter. It reproduces the experience of sketching on a whiteboard, but using a computer. Making mockups is fast. You’ll generate more ideas, so you can throw out the bad ones and discover the best solutions.”

- We have a class license to use Balsamiq for wireframing this semester. This is a fully-functioning license with a trial period extended for the duration of the class.

- Tutorial: https://support.balsamiq.com/tutorials/firstmockup/

- Install Balsamiq Mockups 3 for Desktop, for a PC or Mac, from https://balsamiq.com/download/

- The Class License Key can be used by students as long as it is not shared publicly. The key can be used on multiple machines.

See BalsamiqInfo.txt in Files on Canvas for the key
Gestalt Grouping Principles

- Proximity (nearby elements associated)
- Similarity (elements with visual characteristics associated)
- Continuity (visual system prefers continuous, unbroken contours)
- Closure (visual system will create a complete picture)
- Area (smaller of 2 overlapping elements becomes object of interest)
- Symmetry (the greater the symmetry, the more we ascribe meaning)

Proximity

- Individual elements associated more strongly with nearby elements than with those further away
Similarity

- Elements associated more strongly when they share basic visual characteristics
  - Shape
  - Size
  - Color
  - Texture
  - Orientation

Continuity

- Visual system prefers continuous, unbroken contours
- Will seek out simplest possible explanation for abstract drawings
  - Even if several, plausible combinations exist
Closure

• Visual system will “fill in holes” to create a complete picture

• Will close figures when information absent

Principle of Area

• *Figure* is element that is interpreted as object of interest
• *Ground* is area on which figure rests
• Principle of *area* suggests that smaller of two overlapping elements seen as the *figure* while larger element is seen as ground
  – Also, darker objects appear more often as figure with lighter areas seen as ground
Symmetry

• We prefer symmetry

• The greater the symmetry, the more we ascribe meaning, relationships in the composition

• Symmetrical, unconnected elements are integrated into one coherent object
Micro-survey

Today's learning objectives were:
1. Understand how to define your project's design space.
2. Interaction design
3. Gestalt