Responsiveness and Performance
Responsiveness

- See also: Jeff Johnson, *GUI Bloopers 2.0*
- Key usability problem of interactive systems
  - Bad responsiveness opens Gulf of Evaluation
- Examples for bad responsiveness:
  - A screen pointer that doesn't keep up
  - Delayed response to button-clicks
  - Sliders and scrollbars that lag
  - Applications that go “dead” during disk operations
  - Multiple screen repaints
Reasons for Poor Responsiveness

- Importance not widely known
  - UI designers think of other things first
  - UI designers rarely specify responsiveness
  - Programmers tend to equate it with performance
- This kind of tuning is always difficult
  - “We’ll get it in the next release,” and so on
- Developers treat human input like machine input
- Simple, naïve implementations
- GUI tools and platforms are inadequate
  - Limitations of online apps (which everybody knows about)
Example: Scrollbar

• Does text move as you scroll (good) or after you let go (bad)?

• If designer doesn’t specify, developer will make a decision

• That will usually be the technically simplest
  • Since developers are not trained in user interface theory and concepts
  • Just as UI designers are generally not trained in implementing large software products in C++
Some Eternal Facts

• Responsiveness ≠ performance!

• Processing resources will always be limited
  • We still look at hourglass as much as 15 years ago
  • UIs are real-time systems with deadlines based on human cognition
  • Software does not need to do everything instantly, or in a given order, or even at all
Three Human Deadlines

• 0.1 seconds
  • Perception of cause and effect (recall CMN model)
  • E.g., delay between moving mouse and pointer following, or between mouse click and inverting button

• 1 second
  • Turn-taking in conversation, minimum reaction time for unexpected events
  • E.g., you have 1s max to show progress indicator, open window, or finish system-initiated operations (like auto-save)

• 10 seconds
  • Typical human attention span
  • Max. time for one step of a task
    • E.g., entering a check into a banking program, or completing one step of a wizard
  • Max. time to finish giving input for a task
    • E.g., from selecting “Print” menu entry to sending off the print job
Design Techniques for Responsiveness

• Meet human-time deadlines
  • Rely on the three deadlines and recognize the differences
  • Acknowledge user input immediately, and display busy and progress indicators
  • Use them as frequently as you can, you never know when it will take longer

• Example “Progress bar”:
  • Make it real, show total items remaining, overall progress, and estimated total time remaining
  • Only useful if it advances roughly linearly! (no hanging at 99% please)
  • Estimated time should always go down, never up
  • “Less than a minute” is better than “47 seconds” (why?)
Design Techniques for Responsiveness

- Display important information first
  - Example: How to draw a clock
- Work in parallel
  - Delegate work that isn’t time-critical to background processes
  - Work ahead by preparing likely requests
- Optimize Queueing
  - Create a logical order by looking at all pieces first, then prioritize
Design Techniques for Responsiveness

- Manage time dynamically
- Adjust the strategy if not keeping up
- Decrease quality or quantity to keep up

Example: WordStar
- Ran on a 1 MHz computer, killed by IBM PC
- Written by an amateur, but he accommodated by making the system responsive
- WordStar never dropped characters typed
- Characters typed were always on screen instantly
- Instead stopped updating other areas of the screen
Design Techniques for Responsiveness

- Test under different conditions
- Test under heavy loads
- Test on slower systems, like your customers have
- Test over slower net connections
The Top 5 Performance Hits
Hard Disk Access
Large Memory Footprint
Interlocking Threads
Unsuitable Data and Control Structures
Reinventing the Wheel
Reinventing the Wheel
Interaction Design Notations

- Alan Dix et al.: Human-Computer Interaction, 3rd ed. (2003), Chapter 16
Print "Please enter a number"

INPUT n

Print "The square of",n,"is",n*n

• What are the problems with using such a notation to specify a dialog?
Why UI Specification Languages

- In normal programming languages, UI and algorithms are mixed up
- System and user decisions are hard to distinguish
- Error checking on inputs dominates and complicates code
- First step: bundling I/O in classes/procedures
- Second step: Use a more efficient, readable language to specify the dialog
  - A priori to design the dialog
  - As part of the implementation (executable spec.)
Specifying User Interfaces

- Problem: Describe the proposed design of a user interface
- Approach: natural/semi-formal/formal languages
- Many standard computer science techniques apply
- The more modern the UI, the harder to describe textually, depending on modality and UI style
Grammars

• Mostly BNF-like

\[
\text{expr ::= empty } | \text{ atom expr } | \text{ '(' expr ')' expr}
\]

• E.g., Shneiderman's multiparty grammar

\[
\begin{align*}
\langle \text{Session} \rangle & \ ::= \langle \text{U: Opening} \rangle \langle \text{C: Responding} \rangle \\
\langle \text{U: Opening} \rangle & \ ::= \text{LOGIN} \langle \text{U: Name} \rangle \\
\langle \text{U: Name} \rangle & \ ::= \langle \text{U: string} \rangle \\
\langle \text{C: Responding} \rangle & \ ::= \text{HELLO }[\langle \text{U: Name} \rangle]
\end{align*}
\]

• Great for command-line UIs, e.g., banking ATMs, Unix commands

• Less suitable for GUIs
Grammars

• Regular expressions
  
  • select-line click click* double-click

• E.g., Unix “copy” command synopsis:

  \[
  \begin{align*}
  \text{cp} & \ [-R \ [-H \mid -L \mid -P]\] \ [-f \mid -i \mid -n] \ [-pv] \ \text{source}\_\text{file} \ \text{target}\_\text{file} \\
  \text{cp} & \ [-R \ [-H \mid -L \mid -P]\] \ [-f \mid -i \mid -n] \ [-pv] \ \text{source}\_\text{file} \ldots \ \text{target}\_\text{dir}
  \end{align*}
  \]

  recursion policies  \hspace{1cm}  overwrite policies

• Short and precise, but hard to read, requires additional information about semantics
Production Rules

• Unordered list of rules:  
  \[ \text{if condition then action} \]

• Condition based on state or pending events

• Every rule always potentially active

• Good for concurrency

• Bad for sequence
Event-based Production Rules

\[
\text{select-line} \rightarrow \text{first} \\
\text{click \hspace{0.1cm} first} \rightarrow \text{rest} \\
\text{click \hspace{0.1cm} rest} \rightarrow \text{rest} \\
\text{double-click \hspace{0.1cm} rest} \rightarrow < \text{draw line} >
\]

• Note:
  • Events added to list of pending events
  • ‘first’ and ‘rest’ are internally generated events
  • Bad at state!
Graph Notations: STNs

- State Transition Networks (STNs)
  - Most common tool to specify dialogs
  - Established format (since 1960s)
- Consisting of:
  - States (usually the system waiting for some user action)
  - Transitions (which have a user action and a system response associated with them)
- Describes sequences of user actions and system responses
Example: STN for Personal Orchestra Dialog
baton button clicked / determine selection

- yes / switch to info page
- no / switch to orchestra waiting

requested info page?

baton button clicked / switch to piece selection

complaint ends / display current piece selection

baton button clicked / switch to piece selection

conducted first beat / –

conducting has started / start audio and video

Legend:

input / output

Description of what users do

STATE NAME

input / output

conducting too fast or too slow / jump to complaint sequence

completed successfully / display congratulations page
Please stand on the footprints
Bitte auf die Fußabdrücke stellen
请大家站好位置
• Unconventional notation (agreed upon in the team)
Silhouettes: Client STN

_init_

- run POST
- start Fl.
- connect to Fl.
- look for server
- look for web server

_WAITINGFORSERVER_

- query state from server
- display error on client

_ERROR_

- display error on client

ANY STATE

- server sends state to Fl.
- display shadow

ANY STATE

- display shadow

состояние М

- server sends state to Fl.
- display shadow

ANY STATE OTHER THAN MAINTENANCE STATES

- server sends state to Fl.
- display shadow

ANY STATE

- server sends state to Fl.
- display shadow

ANY STATE

- server sends state to Fl.
- display shadow

ANY STATE

- server sends state to Fl.
- display shadow

ANY STATE

- server sends state to Fl.
- display shadow

ANY STATE

- server sends state to Fl.
- display shadow

ANY STATE

- server sends state to Fl.
- display shadow

ANY STATE

- server sends state to Fl.
- display shadow

ANY STATE

- server sends state to Fl.
- display shadow

ANY STATE

- server sends state to Fl.
- display shadow

ANY STATE
Waiting people will be asked to move up to the front and take their positions on the indicated spots.

Playing people will be presented a welcome screen.

People from the outer waiting queue will take their seats in the inside waiting booths.
Iconic illustrations will remind the player how to create buildings by their shadows.

People will see their shadows while posing in front of the screens.

A countdown will indicate that shadows will be "frozen".

People will see the representation of their shadows as buildings.
ElectricityPosing
• collect features
• broadcast score

ElectricityCountdown
• collect features
• broadcast score

ElectricityResult
• send "illuminate building" → Fl.

A countdown will indicate that shadows will be "frozen".
People will see illuminated buildings in 4 different versions?

Iconic illustrations will remind the player how to pose in order to have their shadows overlap and thus create a power line.
People will see their shadows while posing in front of the screens.
Iconic illustrations will remind the player how to create trees by their shadows.

People will see their shadows while posing in front of the screens.

A countdown will indicate that shadows will be “frozen”.

People will see the representation of their shadows as trees.
Iconic illustrations will remind the player how to pose in order to have their shadows overlap and thus create a water line.

People will see their shadows while posing in front of the screens.

A countdown will indicate that shadows will be "frozen".

People will see the trees that they created with different green colours/trees with leaves?
People will see a animation of what will happen with their skyline.

People will see their city block with the block number and the URL of the DuC Website.

MyCity

• send state → Fl.

MyCity

• send state → Fl.

FullResult

• send state → Fl.
• display mnemonics

“GotoStateMyCity”

“GotoStateFullResult”

• get 3D image ← web server
• display 3D image
People will be asked to move out.

Beyond the playing area, visitors can deepen their understanding of city concepts at individual interactive stations before leaving the pavilion.

PleaseLeave
• assign Player IDs to Game ID
• send this to server

srv. msg.
"GotoStatePleaseLeave"

PleaseLeave
• send state → Fl.
Hierarchical STNs

- **Start** and **Finish** states serve to glue an STN for a sub dialog (e.g., a certain menu selection) into a larger dialog (e.g., operating the application in general).

- Same expressive power as STNs, just more convenient.

- The dialog structure of an entire system can be specified this way.
Using STNs in Prototyping

- Create a simple STN for the dialogs envisioned
- Create one UI snapshot (sketch if paper prototype) per state (label it with the state name)
- Include offscreen area for annotations and to include extra buttons simulating user actions that do not correspond to simple clicks on the current screen
- When walking the user through your paper prototype, consult the STN to find out how to respond to each user action
Using STNs in Prototyping

- Alternative: Let the computer “execute” the STN to run the prototype
- Use tools such as Keynote, PowerPoint
- Demo: Drawing tool prototype in HyperCard
Warum Informatik an der RWTH?
Girls' Day 2011
Schülerinformationstag 2011
Helle Köpfe 2011 für Grundschüler
5 vor 12: Die Wissenschaftsnacht 2011
Fit für Informatik? Mach’ den Test!
Vorkurs Informatik
Bachelor Informatik
Master Informatik
Master of Science in Software Systems Engineering
Master of Science in Media Informatics
Promotionsstudium Informatik
Diplomstudiengang Informatik
Lehramtsstudiengang Informatik - Gymnasium und Gesamtschule
Schwerpunkt Informatik im Studiengang Technik-Kommunikation
Fachschaft Mathematik/Physik/Informatik

Studienberatung Bachelor Informatik
Studienberatung Master Informatik
Studienberatung Master Software Systems Engineering
Studienberatung Master Media Informatics
Studienberatung Lehramt Informatik
Studienberatung Technik-Kommunikation
Auslandsstudienberatung

Prüfungsausschüsse...
Checking STN Properties: States

• Completeness
  • Can you get anywhere from anywhere?
  • How easily?

• Reversibility
  • Can you get to the previous state?
  • But NOT undo

• Dangerous states
  • Some states you don’t want to get to
Checking Transition Properties: Completeness

- Missing arcs indicate unspecified user input
- What happens when the user double-clicks in the Circle states?

Diagram:
- Start → Menu
  - select 'circle'
  - click on centre
  - rubber circle
  - draw circle

- Menu → Circle1
  - select 'circle'
  - click on circumference
  - rubber circle
  - draw circle

- Circle1 → Circle2
  - click on centre
  - rubber circle
  - draw circle

- Circle2 → Finish
  - click on circumference
  - rubber circle
  - draw circle

- Start → Line1
  - select 'line'
  - click on first point
  - rubber band

- Line1 → Line2
  - click on point
  - rubber band
  - draw line

- Line2 → Finish
  - double click
  - draw line
  - draw last line
Checking Transition Properties: Reversibility

- E.g., reversing select ‘line’ requires
  Click - double click - select ‘graphics’
  (3 actions)
- Note: Reverse means just getting back to a state, **not** to “undo” its effect
**Dangerous States Example**

- Word processor: two modes and exit
  - F1 - changes mode
  - F2 - exit (and save)
  - Esc - no mode change
- But ... Esc resets autosave
Dangerous States Example

- Exit with/without save ⇒ dangerous states
- Duplicate states - semantic distinction
- F1-F2 - exit with save
- F1-Esc-F2 - exit with no save
Dangerous States Example: Layout Matters

old keyboard - OK

Diagram:
- Edit
- Menu
- Exit
- F1
- F2
- Esc
- Any update
- Tab
- 1
- F3
- F4

Updated diagram highlights:
- Old keyboard layout is visually represented with proper labels and keyboard function keys.
- The updated layout's state transitions are clarified with different colors and shapes, emphasizing the importance of layout design in interactive systems.
Dangerous States Example: Layout Matters

new keyboard layout

Intend F1-F2 (save)
Finger catches Esc
F1-Esc-F2 - disaster!
Checking STN Properties: Other Transition Properties

- Determinism
  - Several arcs for one action
    - Deliberate: application decides
    - Accidental: production rules
  - Nested escapes

- Consistency
  - Same action, same effect?
  - Modes and visibility
In-Class Exercise: STN

- Simple dialog to select bold, italics, and/or underline

- Draw the state diagram for:
  - Only Bold checkbox
  - Bold and italics checkboxes
  - All three checkboxes
Bold Checkbox

regular style

click on ‘bold’

bold only

Preview

- Bold
- Italic
- Underline
Bold & Italic Combined

- Regular style
  - Click on 'bold'
  - Click on 'italic'
- Bold only
  - Click on 'italic'
- Italic only
  - Click on 'bold'
- Bold italic
  - Click on 'bold'
  - Click on 'italic'
All Three Options

- **regular style**
  - 'italic'
  - 'underline'

- **bold only**
  - 'bold'
  - 'underline'

- **u'line only**
  - 'italic'
  - 'bold'

- **italic only**
  - 'italic'
  - 'underline'

- **bold italic**
  - 'italic'
  - 'underline'

- **italic u’line**
  - 'italic'
  - 'underline'

- **bold italic u’lin**
  - 'bold'
  - 'underline'

**Preview**

- Text Style
  - Bold
  - Italic
  - Underline
Adding Another Option . . .

Normal case

CAPITALIZED
STNs: State Explosion

- Problem: Combining two concurrent STNs with $N$ and $M$ states leads to new STN with $N \times M$ states
- STN hides clear structure of the dialog
- Especially problematic with modern GUIs
- Similar problems with “Escape” and “Help” options
  - ESC can be modeled as special second “Finish” exit active throughout subdialog
  - Help can be modeled as little subdialog hanging off every single state in the STN
- Gets messy
Example: ESC & Help in STNs

- **Main Menu**
  - Select 'graphics'
  - Select 'text'
  - Select 'paint'

- **Graphics Submenu**
- **Text Submenu**
- **Paint Submenu**

- **Circle 1**
  - From Menu
  - Click on centre
  - Rubber band
  - Press HELP button
  - Help Subsystem

- **Circle 2**
  - Click on circumference
  - Draw circle
  - Press HELP button
  - Help Subsystem

- **Finish**
  - ESC
  - Normal
  - Finish

Help Subsystem:
- Draw circle
- Press HELP button
- Click on centre
- Rubber band

Press HELP button to access the Help Subsystem.