

Building User Interfaces

Advanced HCI

IAT351

Week 1 Lecture 1
9.01.2008

Lyn Bartram
lyn@sfu.ca

These slides are largely adapted from Keith Edwards, Georgia Tech

Welcome to IAT 351

Today's agenda

- Introductions
 - Me
 - Ta
 - You
- Class overview
 - Syllabus
 - Resources
 - Grading
 - Class Policies

Introductions

- Instructor
 - Lyn Bartram
 - SUR 14-745 ...or ...SUR 3760 (the HVI Lab)
 - Good: lyn@sfu.ca
 - OK: [MSN:lyn@cs.sfu.ca](msn://lyn@cs.sfu.ca)
 - BAD: 778 782 7439
 - Office hours: Wednesday after lecture
- HCI
 - Visualization
 - How humans work in complex systems
 - Interaction and interface techniques
 - (some) UI architecture

Introductions

- TA
 - Shahin Sheidaei
 - Office: ?
 - Good: ssheidaei@sfu.ca

Now, it's your turn

- Name (pronunciation if not obvious)
- Year, Main concentration
- Interests
- Why are you here?
- Programming background and experience

What are we doing here?

- This class is about ...
- Organizing principles of UI software
- UI technologies
- Practice in UI implementation

BUILDING THINGS!

- Part 1: Basics of traditional 2-dimensional GUIs
- Part 2: Advanced topics (ubicomp, haptics, groupware, etc).

Course info

- Prerequisites:
 - IAT 201
 - Some programming course
- Class structure
 - 90 minute lecture: Wednesday
 - 90 minute workshop: Friday
- Programming exercises and instruction take place in the workshop
- Don't ask for help if you don't attend

Course info

- No textbook, but collections from three books
 1. The Human-Computer Interaction Handbook: Fundamentals, Evolving technologies and Emerging Applications., Jacko and Sears, eds.
 - Electronic texbook available from the library
 2. Human Input to Computer Systems: Theories, Techniques and Technology, Bill Buxton. A manuscript-in-progress on input technologies.
 - electronic
 3. The Collected Readings for the course, available from the bookstore.

Course info

- Web materials
 - Website: www.sfu.ca/iat351
 - (will be up later today)
 - General info (readings, exams, homework)
 - Syllabus
 - Will be updated throughout the semester
 - Will have links to slides and demos used
 - Wiki: <https://wiki.sfu.ca/spring08/iat351d100/>
 - Used as a scratchpad and collected class resource
 - Used for team work on final project
 - Electronic submission (form to be determined)

Some more resources

- Remedial/reference background texts:
 - Norman, “The Design of Everyday Things”
 - Benyon et al., “Designing Interactive Systems”
 - Preece et al., “Interaction Design”
 - Schneiderman text

Resources

- Recommended:
 - *Java Swing, Second Edition*. Loy, Eckstein, Wood, Elliott, Cole
 - Helpful for the Swing-based programming assignments
 - Thinking in Java. Bruce Eckel
 - www.bruceeckel.com
 - Excellent reference and earlier versions available free
- Recommended and Free!
 - Java AWT Reference. Zukowski
 - Somewhat out-of-date, but downloadable!
 - <http://www.oreilly.com/catalog/javawt/book/index.html>
 - AWT is the layer “underneath” Swing
 - Sun’s java site

Evaluation: Two options

- Five programming assignments
 - Each worth 12 %
 - **Individual**
 - Spaced pretty evenly throughout the term
 - Final course project
 - Worth 40%
 - Team of 2 (maybe 1 of 3)
 - Last 3-4 weeks of term
 - Some overlap
- Four programming assignments
 - Each worth 10 %
 - **Individual**
 - Mostly in first 1/2 of term
 - Midterm
 - 20%
 - Final course project
 - Worth 40%
 - Team of 2 (maybe 1 of 3)
 - Last 3-4 weeks of term
 - Writing, design, implementation

Timing complication

- Typically final project is due week 14
 - Report
 - Presentation
 - Demo
- Lyn is away until April 12
- Two options:
- Final project due week 13
- Set a time in week 15 in exam time for class to meet (3 hours)

Programming

- Homework assignments are in Java
 - Java use is required
 - Turn-in and late policy:
 - Due by the workshop on the announced due date
 - Late turn-ins will be marked down 25% for each date they are late
- Project work is more comprehensive
 - Exact details TBD, but likely:
 - Written paper, implementation task, presentation and demo
- What you turn in must compile and run!
- Please pay attention to platform issues (hard-coded filenames, e.g.)

Important

- There will be some Java training in class and practice in the workshops
- If you are not comfortable with Java programming:
 1. Be prepared to learn
 2. Come to the workshops
 3. Use the resources and look for examples!
- While examples and programming assignments are in Swing, focus of the lectures is on broader UI software concepts
 - You'll have to understand how these concepts are applied in Swing
 - We will help with a lot of this, but Swing is huge and you may encounter Swing features/bugs that I am unaware of
 - Be prepared to do independent problem solving if necessary

Motivation

- Moore's law has done its job
- No longer: “ can it be built” ?
- Now: “can they use it?” or, more particularly, “how many ways can it be used?”
- Follow-on: “will they use it”? --> “Can I sell it”?
- Shift towards usability and experience as a key product denominator
 - Good interaction design
 - Good visual design
 - Good industrial/physical design
 - iPod/iPhone

Why study UI software?

- Most systems involve some user somewhere
- Good user interfaces are critical for software survival and economics
- Designing for users is **important** and **difficult**
 - Lots of code devoted to the UI
 - Hard to get right the first time (iteration necessary)

Why are user interfaces difficult to build?

- They are reactive and are programmed from the "inside-out"
 - Event based programming
 - More difficult to modularize
- They generally require multi-processing and concurrency
 - To deal with user typing; aborts
 - Window refresh
 - Window system as a different process
 - Multiple input devices
 - Performance balancing
 - Real-world events

Why are user interfaces difficult to build?

- There are real-time requirements for handling input events
 - Output 60 times a second
 - Keep up with mouse tracking
 - Video, sound, multi-media
- Need for robustness
 - No crashing, on any input
 - Helpful error messages and recover gracefully
 - Aborts
 - Undo

Some good news

- Large set of tools and middleware layers that abstract the lowest level details away from application programmer
 - .NET
 - Java , Swing
- Don't write a device driver any more, write an extension of some device object to meet new criteria
- UI Builders : we will NOT be relying on these in this course
 - Useful for prototyping
 - Often generate bad software engineering :)

What's the user interface?

- Since mid-40's
 - Display (paper terminal, CRT, LCD, ...)
 - Keyboard
 - “command line”, Unix interaction
- Since late '60's
 - Pointing device
 - WIMP/GUI style of interaction
 - “direct manipulation”
 - The desktop
- Since early '90's
 - An extension of our physical environment
 - Sensing, inferencing
 - Different modalities : sound, touch

Programmer's perspective

- The “UI” is typically viewed as one component of the overall system
 - The part that “deals with the user”
 - Separate from the “functional core” (the application)



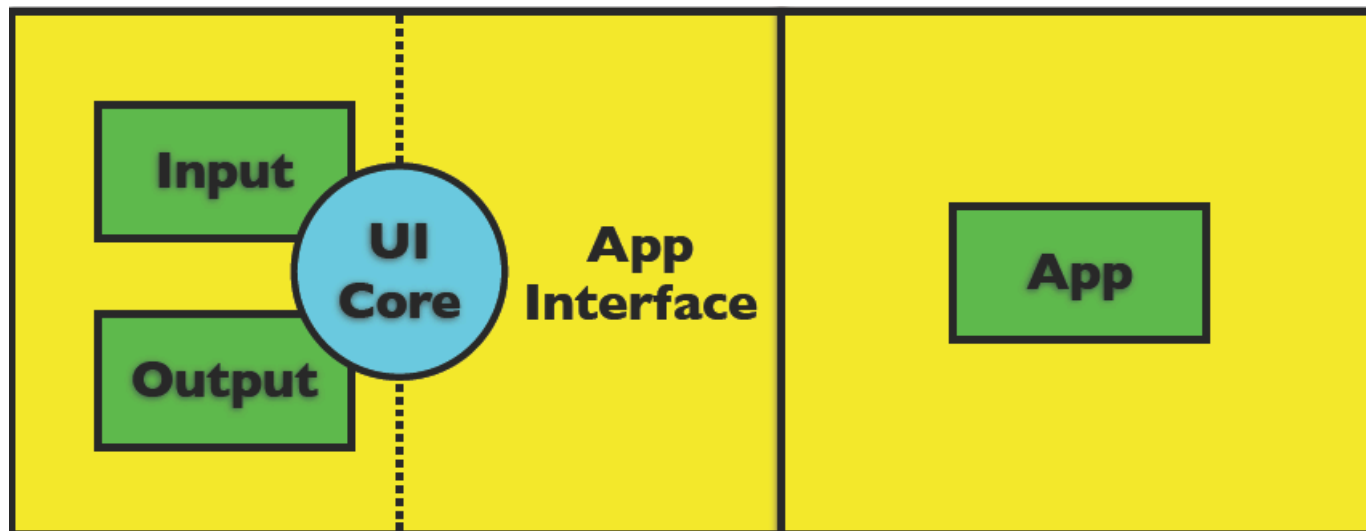
The Software engineering approach

- Advantages of “separation of concerns”
 - Keep UI code separate from app code
 - Isolate changes
 - More modular implementation
 - Different expertise needed
 - Don’t want to iterate the whole thing
 - Design in parallel

Hmm, in practice

- This is very hard to do in real-world application environments
- More and more interactive programs are “tightly coupled” to the UI
 - Programs are structured around the UI concepts and the flow of behaviour
 - UI structure ends up “sneaking into” application
 - Lower level support needs to be present to enable higher-level behaviour
- Not always bad
 - Tight coupling can lead to better performance and feedback

Conceptual overview of the UI



Part 1: Understanding traditional GUIs

- UI software architecture and organisation
- Output and input
 - Devices
 - Affordances and capabilities
 - Software abstractions
- Interaction techniques
 - Models and abstractions
 - How to implement them
- Toolkits and programming environments (a little)

Part 2: Advanced Topics (a whirlwind tour)

- Multiscale input and output
 - Large surfaces, wearable and handheld devices
- Sensing-based interfaces
 - Ubiquitous computing and physical computing
 - Haptics and tangibles
- Pen and ink based interfaces
- 3D user interfaces
 - Displays, environments and interaction techniques
- Sound and speech
- CSCW and groupware
- (perhaps) vision and camera-based interaction

Next meeting: workshop

- Tour of SIAT research work
- Two objectives:
 1. Get an idea of what the scope of interaction research covers, both explicit (main goal) and implicit (applied)
 2. Tickle your brain about future project opportunities for the final project
- Tour will be led by Gordon Pritchard and myself

And we're done for today

- Questions?