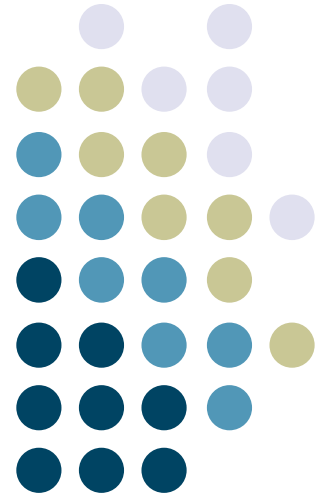
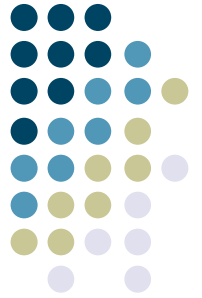


CS4470: Intro to UI Software  
CS6456: Principles of UI Software

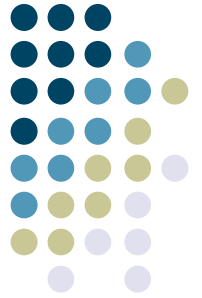


Keith Edwards



# Today's Agenda

- Introductions
  - Me
  - TAs
  - You
- Class Overview
  - Syllabus
  - Resources
  - Class Policies



# Introductions

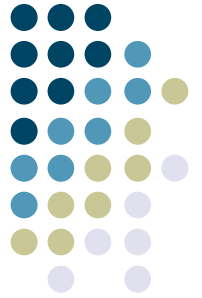
- Instructor
  - Keith Edwards
  - TSRB 213
  - good: keith@cc.gatech.edu
  - bad: 404-385-6783
- My research areas:
  - Technical side of HCI
  - UI infrastructures, new interaction techniques
  - Ubicomp
  - Making security, networking more usable

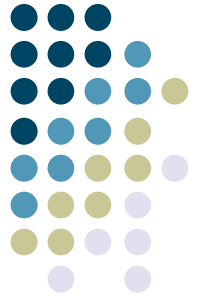


# Introductions

- TA

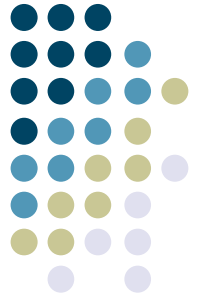
Georgia  
Tech





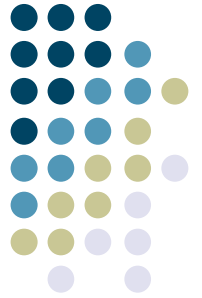
# Now, It's Your Turn

- Name (pronunciation if non-obvious)
- Major, Year
- Interests
- Why UI SW?



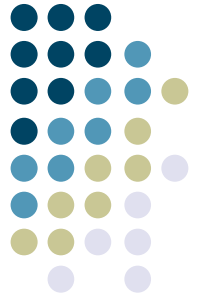
# What is this class about?

- Organizing principles of UI software
- Practice in UI implementation (lots)
  
- First half of the semester: Basics of traditional 2-dimensional GUIs
- Second half of the semester: Advanced topics (animation, gestures, recognizers, audio, etc.)



# What this class is NOT about

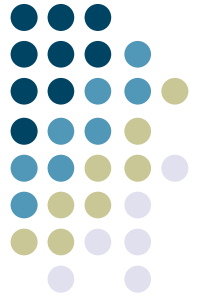
- User-centered design
  - That's what 3750/6750 are for!



# Basic Course Info

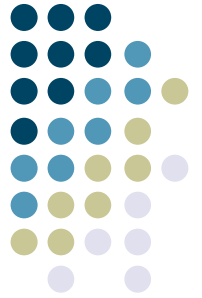
- “Prerequisite”: CS3750/6750
  - Remedial background texts:
    - “Human-Computer Interaction,” Dix, Finlay, Abowd, Beale
    - “The Design of Everyday Things,” Norman
- Web materials
  - Up now:
    - See CoC website for link
  - General info (books/readings, exams, homework)
  - Syllabus
    - Will be updated throughout the semester
    - Will contain links to lecture slides





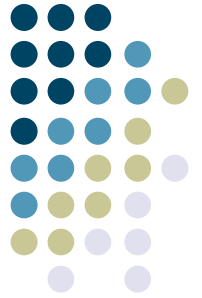
# Resources

- Recommended:
  - *Java Swing, Second Edition*
    - Loy, Eckstein, Wood, Elliott, Cole
    - O'Reilly Associates
    - Helpful for the Swing-based programming assignments
- Recommended and Free!
  - Java AWT Reference
    - Zukowski
    - O'Reilly Associates
    - Somewhat out-of-date, but downloadable!
    - <http://www.oreilly.com/catalog/javawt/book/index.html>
    - AWT is the layer “underneath” Swing



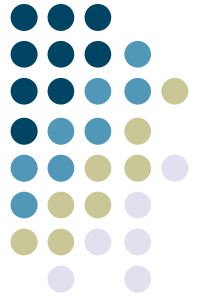
# Grading Criteria

- Different criteria for undergrad versus grad
- Undergrads:
  - 5 individual homework assignments (5 x 12%)
- Grads:
  - Do the **first two** individual homework assignments (2 x 10%)
  - Research project (40%) -- two person teams
  - Writing, implementation, presentation
- Everyone:
  - Midterm and final exam (20% each)



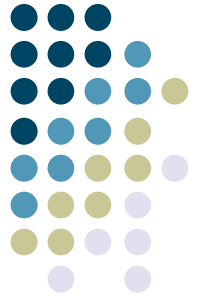
# Other Policies

- **Homework** assignments are in Java
  - Java use is required
  - Turnin and late policy:
    - Due 11:55PM on the announced due date
    - Late turnins will be marked down 10% for each date they are late, maximum of three days
- Grad **Project** work is more flexible
  - You will choose programming environment
  - Multiple milestones involving 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.)



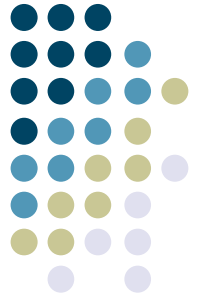
# Important Note

- There will be minimal Java training in class
- If you are not comfortable with Java programming:
  1. Learn
  2. Drop course
- 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
  - I can 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



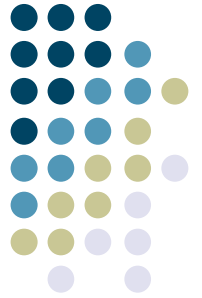
# GT Honor Code

- All homeworks are expected to be completed by individuals and not in collaboration with others. While you can ask others about Swing coding problems, no code sharing is allowed.
- Grad projects are done in teams of two; be clear about what code you wrote and what code you are reusing (if you're using external libraries, etc.)
- Obviously, all exams are to be completed by the student alone.
- Students are expected to follow the GT Honor Code. I am required to forward all suspected cases of academic misconduct to the Dean of Students, where they will be pursued to resolution.
- This is a very unpleasant experience for all involved, so please do not put us in this situation.



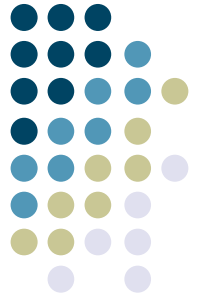
# Motivation

- Moore's Law has done its job...
- No longer: "Can you build it?"
- Now: "Can they use it?"
- Follow-on: "Will they use it?" → "Can I sell it?"
- Shift toward "usability" (broadly defined) as a key product differentiator
  - Good user experience design
  - Good visual design
  - Good physical/industrial design
  - Think: iPod, iPhone



# Why a class on UI software?

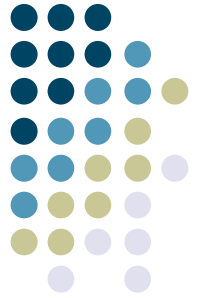
- Most systems are built for a user
- Good user interfaces are critical for software survival and economics
- Designing for users is *important* and *difficult*
  - Lots of code devoted to UI
  - Hard to get right the first time (iteration necessary)



# What's the User Interface?

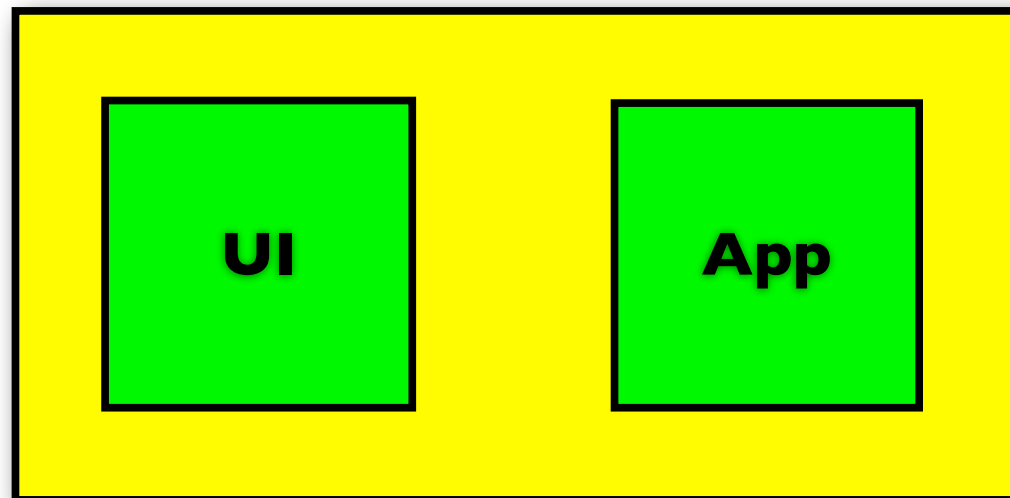
- Since mid-40's
  - Display (paper terminal, CRT, LCD, ...)
  - Keyboard
- Since late '60's
  - Pointing device
  - WIMP/GUI style of interaction
- Since early '90's
  - An extension of our physical environment
  - Sensing, inferencing

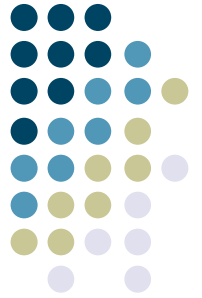




# Programmer's Perspective

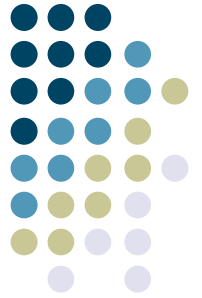
- The “UI” is typically viewed as one component of the overall system
  - The part that “deals with the user”
  - Separate from the “functional core” (AKA the “application”)





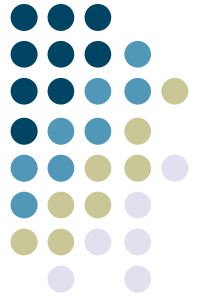
# Software Engineering and the UI

- 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

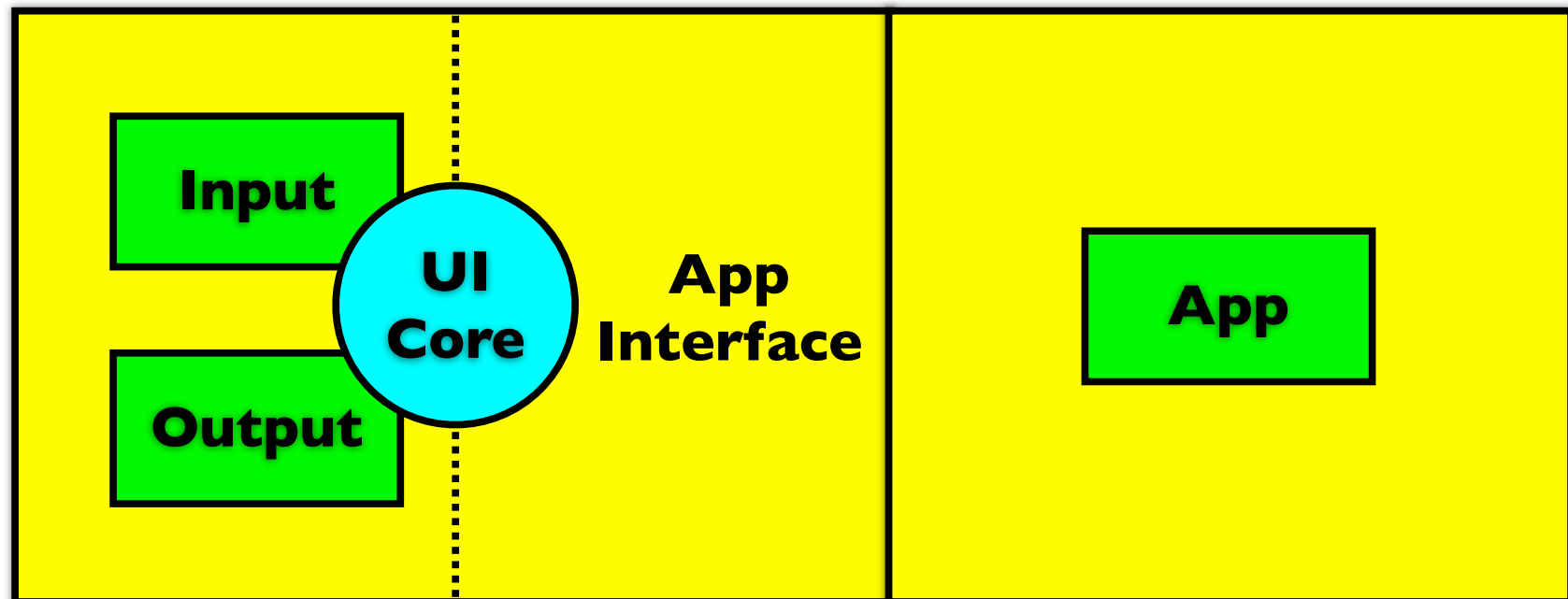


## In practice, very hard to do...

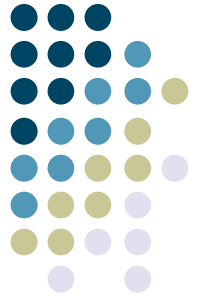
- More and more interactive programs are tightly coupled to the UI
  - Programs structured around UI concepts/flow
  - UI structure “sneaks into” application
- Not always bad...
  - Tight coupling can offer better feedback/performance



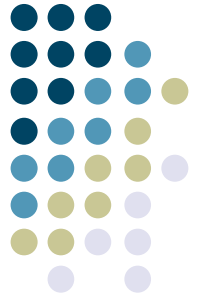
# Conceptual Overview of the UI



# Part I: Understanding Traditional GUIs



- UI software architecture and organization
- Input and output
  - Devices, software abstractions
- Interaction techniques and how to implement them
- Toolkits and programming environments



## Part II: Advanced Topics

- Multiscale input and output
  - Large surfaces, handheld or wearable devices
- Multitouch (two-handed) input
- Zoomable interfaces
- Animation
- Natural interaction types
  - Ink, audio, video
- Sensing-based interfaces
  - Recognition, context awareness
- Paper-based interfaces
  
- Requests?