Programming Languages, Cognitive Science, and Computational Thinking

Michael Hansen

Lab Lunch Talk, Fall 2011

INDIANA UNIVERSITY
PERVASIVE TECHNOLOGY INSTITUTE

A Quote and a Question

"Millions [spent on researching] compilers, but hardly a penny for understanding human programming language use." – Newell and Card, 1985

- What role should Cognitive Science play in…
  - Language (library) design and evolution?
  - Computer Science Education?

Language Design (1/2)

"Programming is the process of translating a mental plan into one that is compatible with the computer."

– Hoc & Nguyen-Xuan, 1990

- Language design guided by
  - Technical objectives (scalability, mathematical elegance)
  - Problems with previous languages
  - Feedback from technical community

Language Design (2/2)

- Who is the target audience?
  - 23:1 individual differences (Curtis, 1984)
  - Experts internalize “programming plans” and “rules of discourse” (Solloway, 1984)
  - Debugging models
    - Assume experts and novices use same process, but…
    - Novices introduce errors by correcting immediate behavior
- Most successful language ever?
  - VBA in Excel ☹

HCI Dimensions

- Visibility
  - Memory overload
- Closeness of Mapping
  - High-level operators match domain
  - Low-level primitives used otherwise
- Speak the User's Language
  - If unfamiliar, superficial knowledge transfer
  - Natural language, mathematics

Problem: Take a look at the following definition:

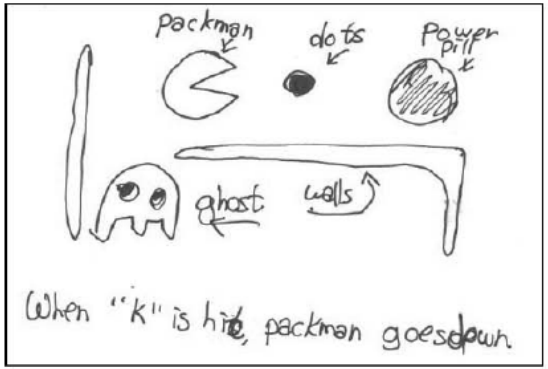
```
bool operator<(const T& x, const T& y)
{
  return true;
}
```

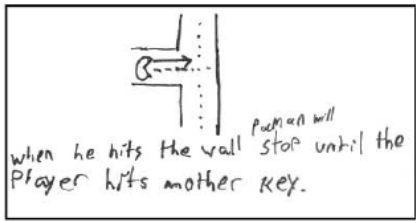
Explain why this is wrong for any class **T.**

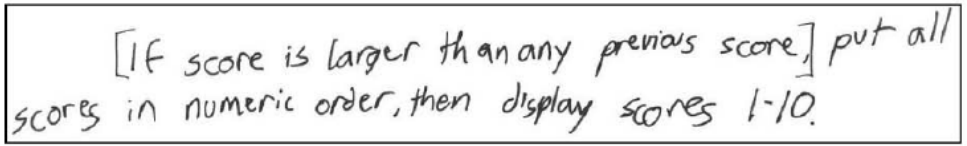
- Stepanov, 2007

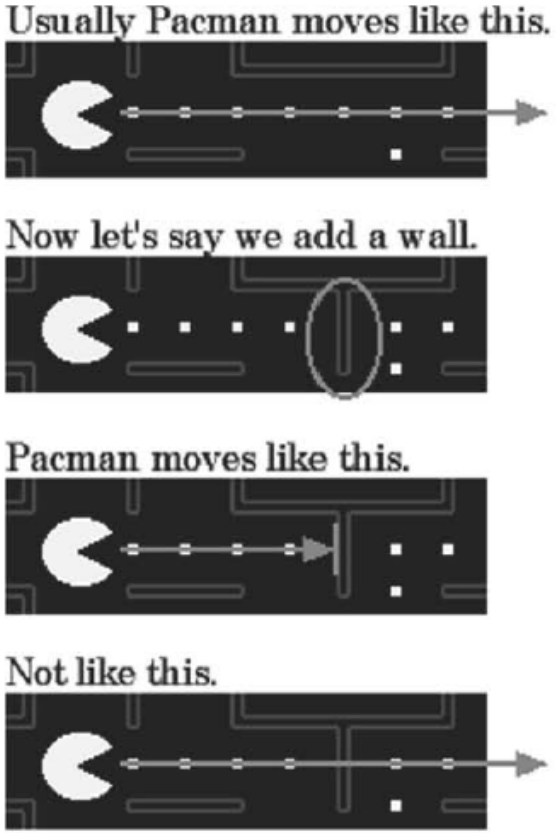
Case Study (1/3)

- PacMan game design (Pane, 2006)
  - Children and adults









Do this: Write a statement that summarizes how I (as the computer) should move Pacman in relation to the presence or absence of other things.

Case Study (2/3)

- Event or rule based
  - When PacMan hits a wall, make him stop
- Aggregate operators
  - Turn all ghosts blue, move all scores down
- Natural language arithmetic
  - Add 100 to score (not `score += 100`)
- State is remembered
  - Motion not modeled as continuous update
- List data structures
  - Complex structures arise from queries

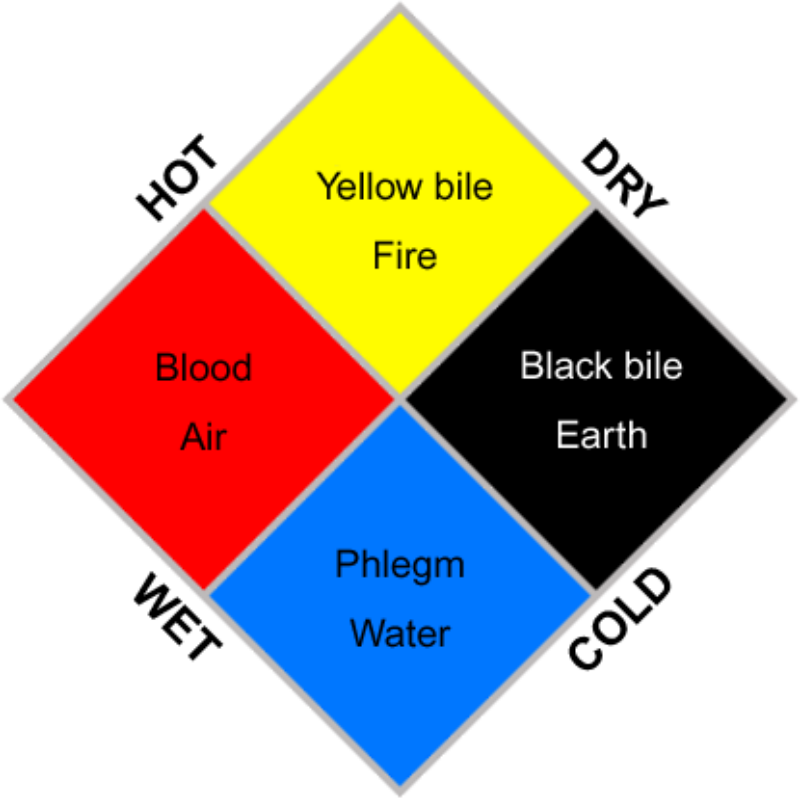
Case Study (3/3)

- Boolean expressions avoided
  - Mutually exclusive rules or general case + exceptions
  - Operator precedence context-dependent
    - `select the objects that match (not red) and square`
    - `select the objects that match (not triangle and green)`
- Previous findings (Miller, 1974; 1981)
  - Looping structures do not match intuition (aggregate vs. iteration)
  - AND, OR, NOT do not match natural lang. sematics
  - if-then confusion
    - We see `if X then Y; Z`
    - They read `if X Y then Z`

Sounds About Right?

HCI Dimensions

- Visibility
- Closeness of Mapping
- Speak the User's Language

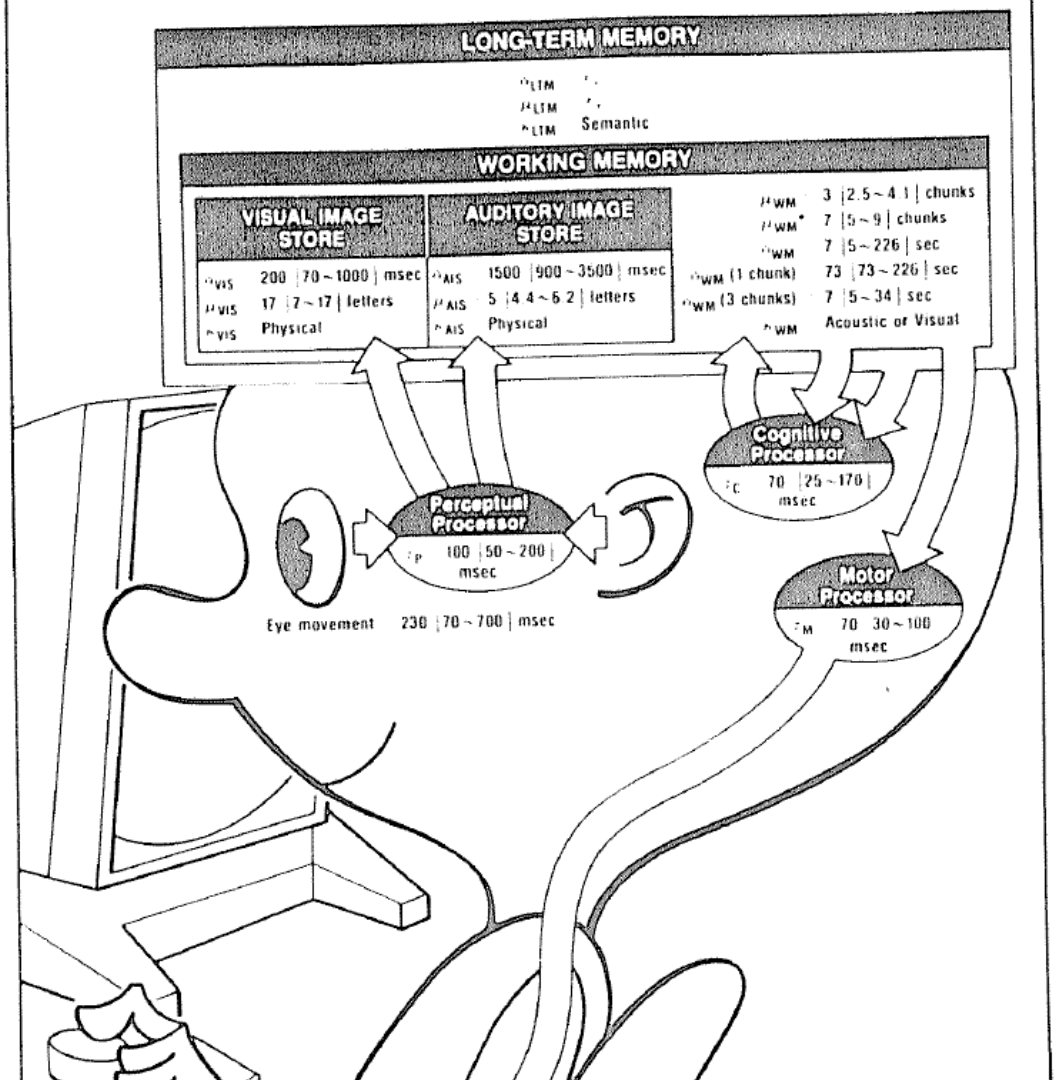


Theory and Practice (1/2)

- Usability studies, little appeal to Cognitive Science
  - Lacking engineering-style theory of HCI
- *"Hard sciences drive out soft"* – Newell and Card
  - Computer Science drives out HCI
  - Don't need to know much about user
- Need to start somewhere
  - Approximate, quantitative models of user interaction
  - Model Human Processor, GOMS, Keystroke-Level

Time Scales

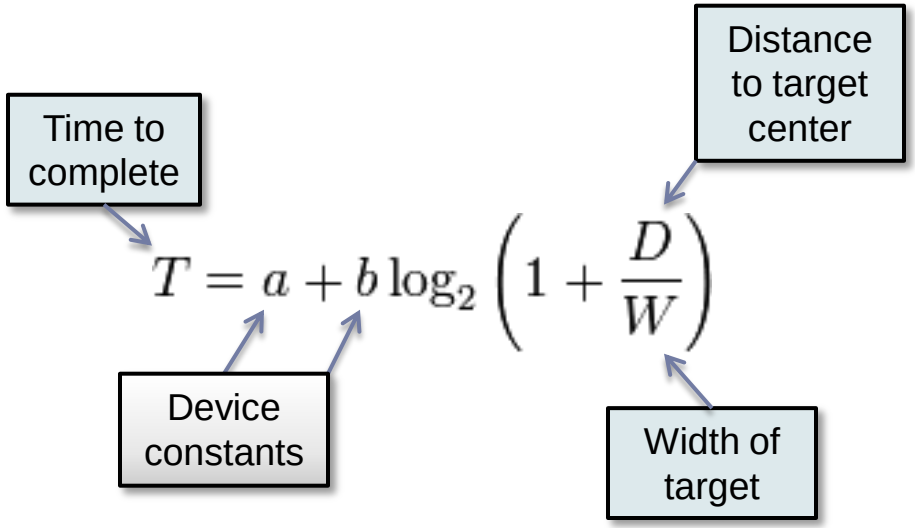
- Natural Law
  - < 30 ms
- Psychology
  - 30 ms to 30 sec
- Bounded Rationality
  - > 1 min



Theory and Practice (2/2)

- *"Tradeoffs, such as between the effort to learn a complex interface and the power of having it, could be understood enough to affect the types of interfaces explored."*

- Fitt's Law
  - Speed/accuracy trade-off
  - Not a cognitive model…



Criticisms

- Too low level
  - Time scale below Psychology (perceptual)
- Too limited in scope
  - Good cognitive models can be generalized, but…
- Too late
  - Research becomes obsolete
- Too difficult to apply
  - Keep trying, build on approximate models

Perceptual to Conceptual

- Programming is difficult for beginners
  - *"…[students] find it difficult to identify what is important in a problem and produce convoluted solutions that replicate the problem complexities."* (Kramer, 2007)
- Abstraction is key
  - Leaving out irrelevant aspects of problem
- Generalization
  - Extraction of common features from specific examples

Computational Thinking (CT)

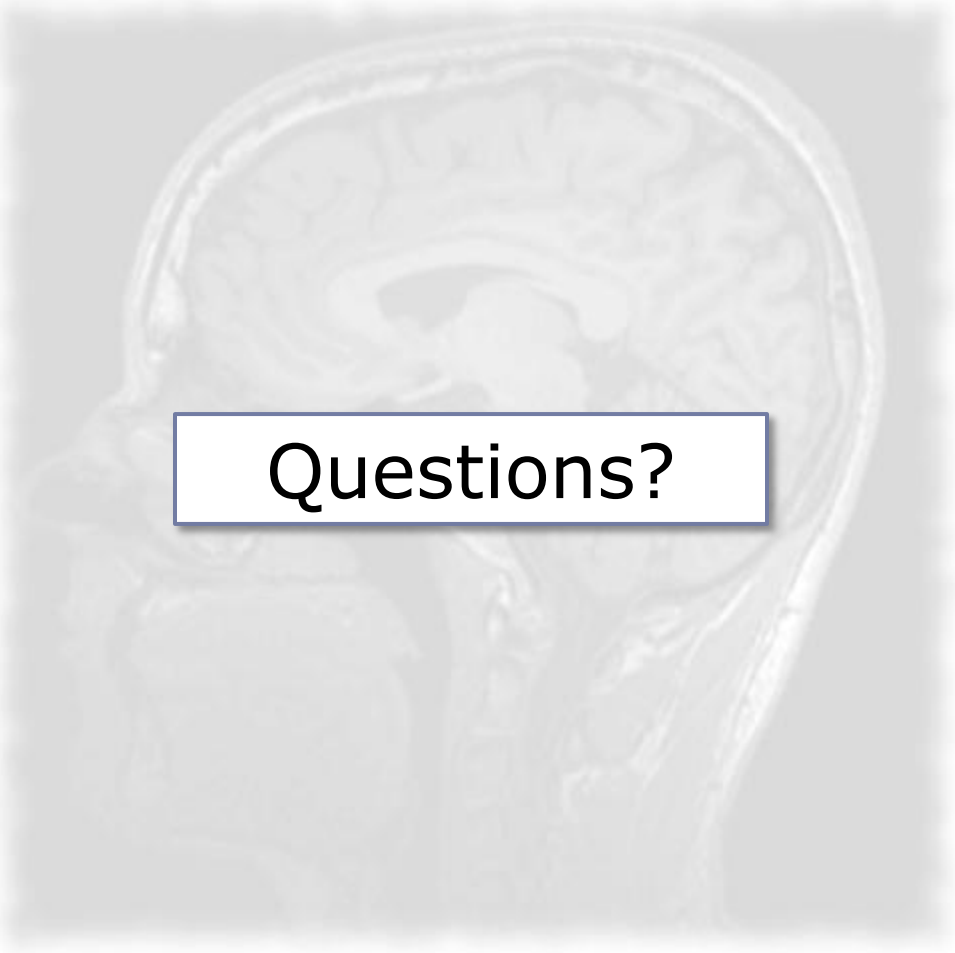
- Programming : Computer Science (Lu, 2009)
  - Proof construction : Mathematics
  - Literary analysis : English
- 1st exposure to CT is programming
  - Should be entrance requirement
- Core CT concepts?
  - Search space, initial & final states, operations, heuristics, efficiency, concurrency, recursion, non-determinism

Perceptual and Conceptual

- Have experts just embodied CT concepts?
  - Perceptual efficiency, cues (Goldstone)
  - What % of programming is skill-based?
- How to model perceptual-conceptual level?
  - Memory limits, primacy/recency, chunking
  - Categorization, learning (Minerva, Beagle)
  - Goals, plans, task-dependency (Soloway; Cant, 1995)
  - Not too general…

Open Questions and Directions

- Which CT concepts are essential?
  - Task-dependent, but perhaps general
  - How to expose CT concepts in a language?
- Evaluating language features
  - Need quantitative models to understand trade-offs
    - Perceptual-conceptual
  - Experts vs. novices
    - Do we need separate languages?


Questions?