CS160: Lecture 19

John Canny

Human Learning and Help Systems

- Why study human learning for HCI?

Why Study Human Learning?

Ans: People need to *learn* new applications, often using various forms of Help.

Ans: The way people learn should influence the design of help systems, and perhaps the entire system e.g.

- Knowledge is "situated" in particular contexts, so help should reflect that (scenarios/common tasks)
- Learning is layered on old knowledge in a roughly novice → expert trajectory
- Learning involves a concrete → abstract progression
- Fluency with abstraction varies across the population, esp. with degree of schooling

Transfer Learning

- Learning is a process of building new knowledge using existing knowledge.

- Knowledge is not acquired but constructed out of existing "materials".

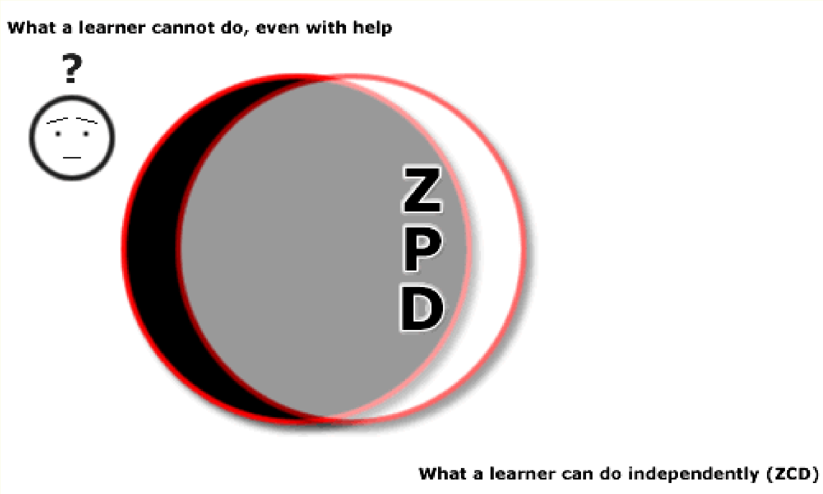
- The process of applying existing knowledge in new settings is called ***Transfer***.

Transfer Learning examples

- You've learned basic edit commands from MS Notepad, and you transfer to MS Word.
- You've installed a simple program (lets say Quicktime), and you transfer that knowledge to an MS Office installation.
- You've done a mail merge in MS Office 2000 and you transfer to Office XP.

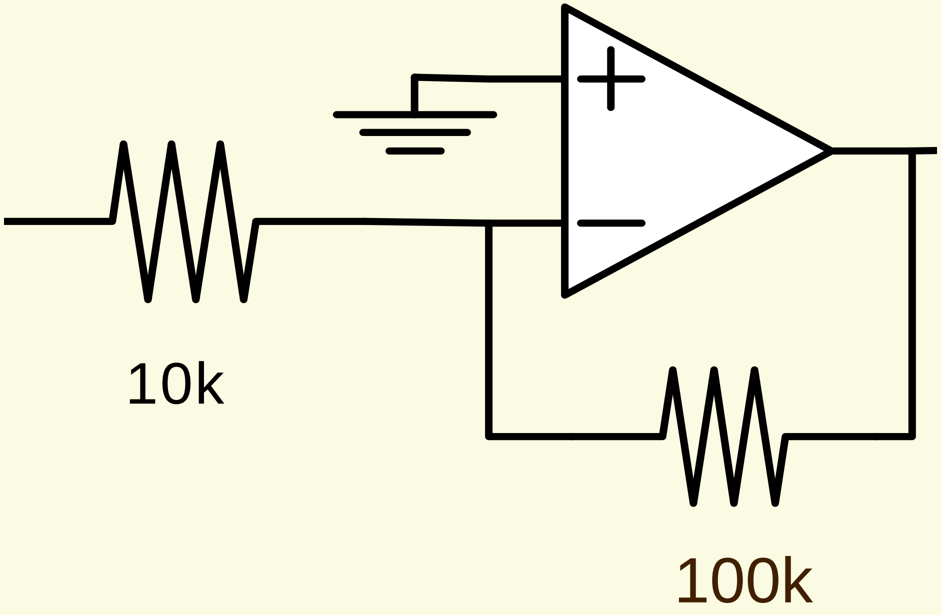
ZPD: Zone of Proximal Development

- Learning is layered and incremental.
- In societies, learners are helped by others.
- In fact learners have a “zone” of concepts they can acquire with help.
- This is the Zone of Proximal Development (ZPD).


What a learner cannot do, even with help
?
ZPD
What a learner can do independently (ZCD)

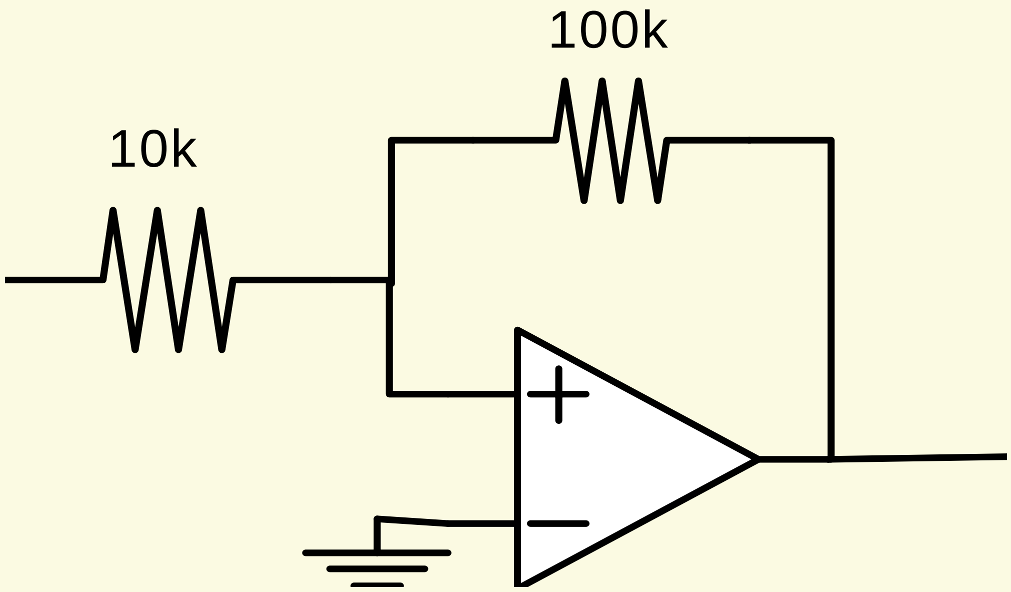
ZPD example

- Who knows what this is?



ZPD example

- What about this?



Learning new applications

- Applications should be designed to fit in the target users' ZPD: it should assume the knowledge they typically have, and a realistic amount of "support."

- People often learn how to use computer systems with the help of others, but you have to be realistic about this in your application context.

Learning by doing

People learn best by ***doing***:

- It marshall's all of their "processors" (cognitive, perceptual, sensori-motor).
- It forces them to apply a conceptual model to figure out how the system will behave.
- It allows them to observes differences between the system's actual behavior, and what they anticipate from their conceptual model. This helps them refine and improve their model.

Q: What's a good example of this?

Learning by doing

Q: What's a good example of this?

A: Programming!

Imagine a computer engineering course where you learn how to program only "by being told."

In contrast, medical schools and some CS departments are experimenting with "Problem-Based Learning," where there are only projects (no lectures). Lecture-style material is only available as a "library" resource to help with project work.

Learning and experience

- Learning is most effective when it connects with the learner's ***real-world*** experiences.
- The knowledge that the learner already has form those experiences serves as a foundation for knew knowledge.

Learning and transfer

- Transfer is certainly enhanced by similarity between the old and new contexts.
- What other factors affect transfer?

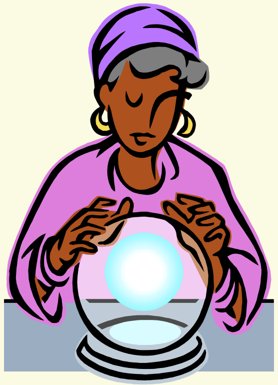
Transfer and understanding

- Transfer depends on thorough learning in the first situation (learning with understanding*).
- The more thorough the understanding in the first situation, the more easily knowledge will transfer.

Understanding

- By understanding we mean that a person has a mental model of why a thing behaves as it does.

- This model allows the person to predict how the thing behaves in other situations, and to “explain” their reasons for that conclusion.



Transfer and Generality

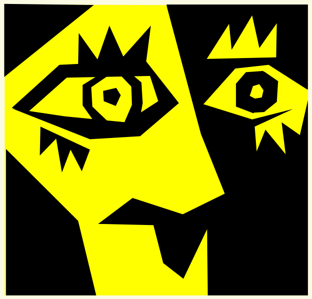
- Generality of existing knowledge: has the learner already seen it applied in several contexts?

Transfer and Motivation

- Motivation: is the new knowledge useful or valuable?
- Motivation encourages the user to visualize use of the new knowledge, and to try it out in new situations.
- Students are usually motivated when the knowledge can be applied to everyday situations.

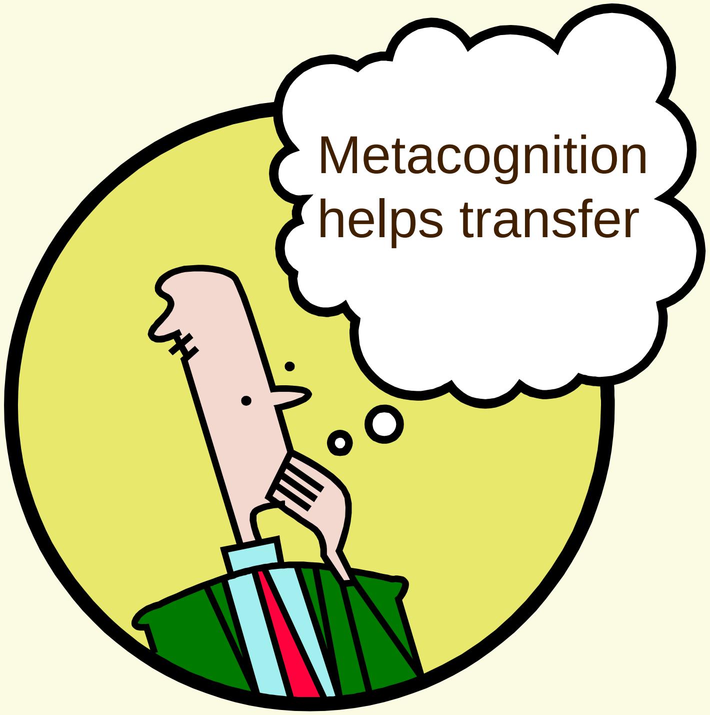
Transfer and Abstraction

- Is the existing knowledge abstract or specific?
- Abstract knowledge is packaged for portability. Its built with virtual objects and rules that can model many real situations.
- E.g. clipart



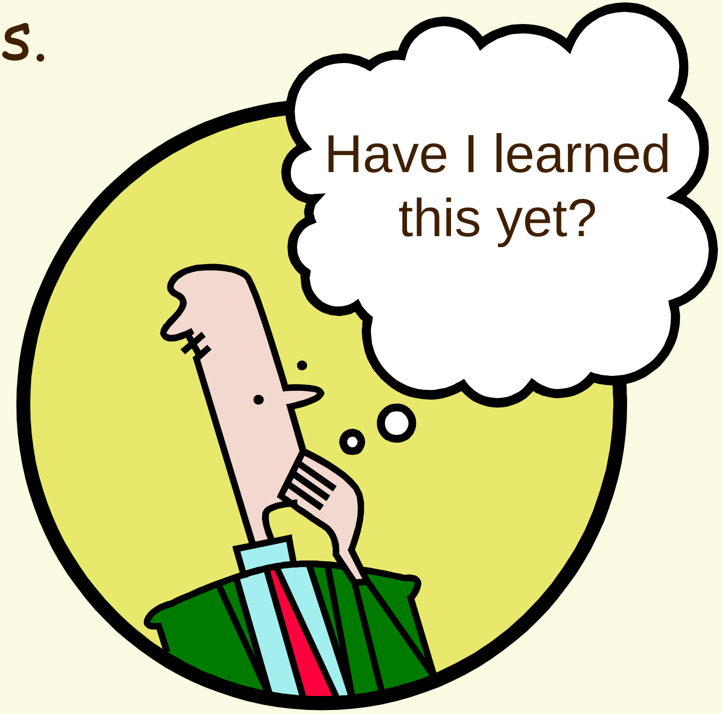
Metacognition

- Metacognition is the learner's conscious awareness of their learning process.



Metacognition

- Strong learners carefully manage their learning.
- For instance, strong learners reading a textbook will pause regularly, check understanding, and go back to difficult passages.
- Weak learners tend to plough through the entire text, then realize they don't understand and start again.



Piaget: Stages of learning

Jean Piaget observed very systematic progression of knowledge in children through stages:

1. Sensori-motor (acting, observing, remembering)
2. Semiotic or symbolic (naming things)
3. "Concrete" operations (relationships, transformations)
4. Propositional or formal thought

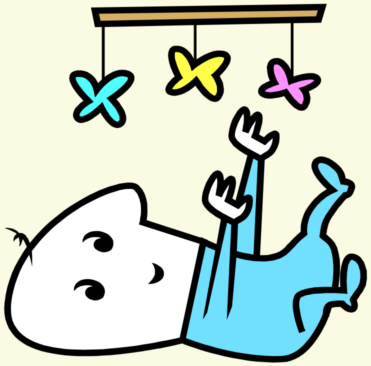
Piaget: Stages of learning

Jean Piaget observed very systematic progression of knowledge in children through stages:

1. Sensori-motor (< 2 years)
2. Semiotic or symbolic (> 1.5 years)
3. "Concrete" operations (2-7,7-11 years)
4. Propositional or formal thought (> 7 years)

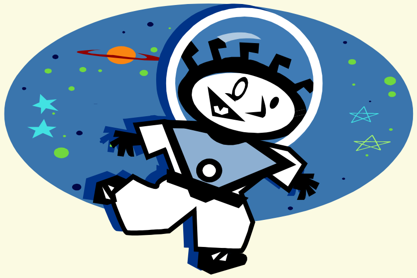
Sensori-motor stage (< 2 years)

- Conditioned behaviors, and first hand-eye coordination.
- Grasping, manipulating things.
- Some indirect manipulation.
- Object persistence.



Semiotic stage (>1.5 years)

- Children continue to play with "missing" objects, and may use gesture to invoke them.
- This soon turns to imaginary play.
- Drawing.
- Speech – naming first the things that are present.
- Then referring to things that are not present, and to the past and future.



Concrete thought (2-7,7-11 years)

- Concrete thought: a system of (real) objects, relationships, and operations on them.

- Children "understand" things by being able to relate them to similar things, and to predict the consequences of their actions.

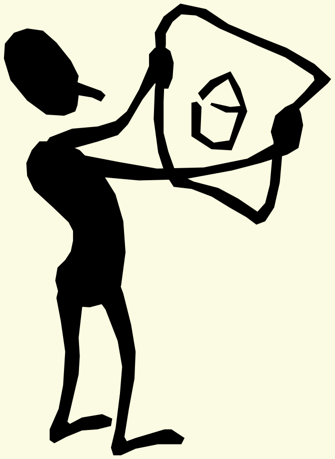
- They can plan and act to achieve a desired outcome.

Concrete thought

- But early concrete thought is still tied to direct experience – it is not "de-centered."
- E.g. children in this stage can navigate through their neighborhood, changing their route if needed.
- i.e. they can mentally model and predict the results of their actions.
- But they cannot indicate that route abstractly, say on a map.

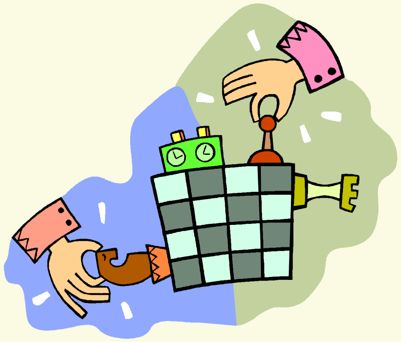
Concrete thought

- Concrete thought includes rich spatial and temporal relationships.
- Visual design is a “concrete” process.



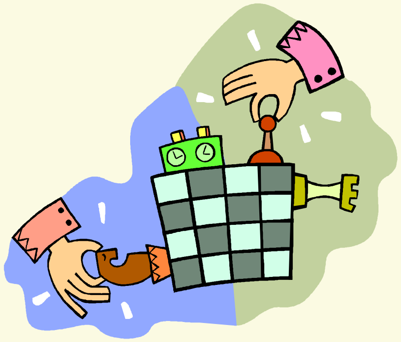
Formal thought (11+ years)

- Objects and operations no longer need to relate to the world. Things don't need to be true or consistent. Thinking is a "game".
- "Operations" are more abstract, and often complementary e.g. joining-separating.
- Children learn a number of principles, like reversibility, proportion, chance.



Formal thought caveats

- Researchers have found that the transition to formal thought is not as reliable as Piaget had thought.
- Many features of this stage are missing in children who do not attend school.
- This stage corresponds with the transition from learning from experience (pre-school), to learning from texts (school).

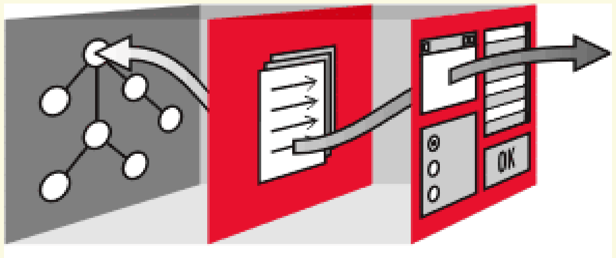


Thought styles

- Designers and other visually-oriented people usually favor concrete thought – context-dependent, rich representations.
- Technologists and mathematically-oriented people favor formal thought – context independent, sparse representations, rich consequences.

A mismatch

- Many interface researchers (technologists) tried to build UI design tools using abstract interface specs (UIMSes)
  - the designer specifies rules about the interface and the system finds a solution satisfying them.



- Real designers hated this idea. They lost control over spatial relationships and overall layout which was lost in the rules.

Piaget's progression

- The Piagetian progression can be a good model for the progression in learning new concepts, like how to use a computer program.

- Look for a Sensori-motor → Symbolic → Concrete → Abstract progression in your own learning, and in your users'.

Break

Help models

- What kind of help works best for you?
- Do you ever “read the manual”?
- Is help usually “where you need it?”
- What are some differences between help you get from people and from systems?

Types of Help

- **Quick Reference:**
  - Reminders of common command names or options.
  - Good to place on a card, or for small devices, on the device itself.
  - Use a few main categories to avoid long search..
  - E.g. for an editor, categories like “basic”, “search/replace”, “load/save”, etc.

Types of Help

- **Task-specific help**
  - User needs help on how to apply a command, or to complete a task.
  - Can be part of a "how-to" system for common tasks.
  - Should be easily accessible from the command line (if text).
  - Make "options" windows ***obvious*** and ***easy*** to find!
  - E.g. add "advanced" button in the dialogue to apply any command.

Types of Help

- **Full explanation**
  - User wants complete understanding, to get best value out of the application.
  - This part explains the "why" more than the "how".
  - E.g. How do compiler options affect performance?
  - What are various installation components used for? What are the ***uncommon*** commands?
  - Probably need a chapter in the help system for this. More system-centric than task-centric.

Types of Help

- **Tutorial**
  - The tutorial leads the user through a task, scaffolding their actions.
  - Should allow users to act as well as watch (sandboxing).
  - The "best" way to teach!
  - More work to build into the system, but you should leverage your company's other effort:
    - E.g. most software houses conduct regular training sessions for customers – these are ideal tutorial content.

More advanced ideas

- Help is a kind of ongoing learning environment.

- Minimalist instruction (Carroll '92) is a learning approach
  - It shows users what to do,
  - then gives them realistic tasks to solve.
  - It eliminates conventional exercises, tedium and repetition, and encourages users to explore.
  - It also has extensive coverage of error recovery.
    - - users feel confident exploring.

More advanced ideas

- Help could be enjoyable? - at least it's a special case of computer-supported learning..

- "Training wheels" (Sandboxing)
  - Advanced commands are removed until user gains some experience with the system.
  - Also some "dangerous" commands.
  - Users explore freely in this sandbox.
  - Users gained better understanding (IBM trial).

Desiderata for help

- **Availability**
  - Should be accessible anywhere (always include a help key on each major window).
- **Accuracy and Completeness** (hard!)
  - Make sure it matches program version, and that it covers all the commands.
  - As well as commands, common tasks should be described.

Desiderata for help

- **Consistency**
  - Content, terminology, style.
  - These days, online and printed manuals are often the same.

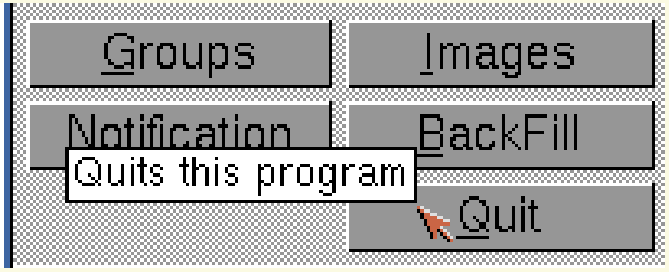
- **Robustness**
  - Help shouldn't crash if the program does (need another thread).
  - Program exceptions can bring up the help system.

Desiderata for help

- **Flexibility**
  - Includes adaptation to context or user skill. Multi-level help is a good idea.
- **Unobtrusiveness**
  - Shouldn't disrupt users work (like the annoying help characters in MS Office). A separate help screen is often good - supports rapid switching.

Context-sensitive help

- Help depends on where it is used:
- Tool tips ↓ or the windows ? symbol:



- Save the user the burden of synchronizing program state with help system state.
- Almost always a good idea to do this.
- Just make sure the user can easily find the main help contents and index.

Online tutorials

- Can be useful, BUT:
  - Users are not the same, some need minimal help.
  - Forcing the user to execute a particular command is boring and annoying, and doesn't help learning.
- So..
  - Make sure users can skip steps.
  - Show users multiple ways of doing things.
  - Give partial information on what to do, with more information available if the user requests it.

Adaptive Help Systems

- Adaptation is a good idea because:
  - It avoids information that is too detailed or not detailed enough for a particular user.
  - It avoids repetition of info the user has already seen.
  - Can make suggestions of new ways to do tasks that the user may not know.
- Weaknesses:
  - Information can disappear (bad if the user forgot it too!).
  - System needs to know user identity and user must use the system for some time.

Initiative

- A Help system works with the user, and ideally should allow a spectrum of control:
- "Help me", "tell me what to do", "show me what to do", "OK, I'll take over now…"
- This is called "mixed initiative".

Initiative

- A good mixed-initiative help system requires links between all parts of the system including a tutorial.
- User should be able to "take over" at any time, then give back control.

Design issues

- Help system design is like other parts of the interface.
  - Start with task analysis.
  - Do paper prototypes.
  - Do user tests at informal and formal stages - look for errors.
  - Use errors as the “objects” to guide the design of the help system.

Summary

- Human Learning:
  - Transfer
  - Zone of Proximal Development
  - Meta-cognition
- Piaget's stages in children's learning.
  - Concrete vs. abstract thought
- Help system design principles and types