Education Technology

CIS 7000-001

Andrew Head & Danaé Metaxa

Last time

Audits are a method from social science used to uncover **discrimination in opaque human processes**, usually related to hiring or housing

Beginning in the 2010s, AI auditing emerged to apply the same method to **algorithmic decision-making** in the criminal system, healthcare, surveillance, and more have had significant impact

Auditing is beginning to be **required by law**, leading to a whole new era (with accompanying challenges) of auditing

Everyday users also engage in auditing behavior

Today

Learner-centered design

Intelligent tutoring systems

Constructionism

Announcements

Start thinking about student presentations, to be held 4/23 and 4/28.

Presentation slot sign-ups posted on Canvas, due 4/7.

You will be working with partner.

Do a deep dive into a research paper on one of the topics from class. Give a presentation of that work. Tie it back to the concepts from class. Situate the paper to related work under the theme.

Use this as an opportunity to deepen your HCI knowledge and contribute interesting food for thought to your peers.

Learner Centered Design

Beyond User-Centered Design

[Soloway et al. 1994]

Conventional **user-centered design** focuses on understanding the needs of **experts** and helping them do their work more efficiently and pleasantly.

But in many situations, we are trying to improve the process of **learning**, where efficiency of task completion is not the main goal.

Learner-Centered Design

[Soloway et al. 1994]

Professionals	**Students**
Know domain	Do not know domain
More motivated	Sometimes less motivated
Relatively homogeneous	Relatively diverse
Growth less of an issue	Growth a core issue

Scaffolding as the solution

[Soloway et al. 1994]

Scaffolding generally refers to techniques where we provide **support to learners** while they are learning a new task.

We might provide scaffolding for...

- learning the **task**. **coach** students to learn about how to do a practice in a target domain.
- using a **tool**. make it **adapt its functionality** to user competence.
- the **interface** to tools. support the **media and modes of expression** that learners will try to employ.

This lecture

The technologies we talk about today are going to be largely about helping people learn to do a new **task**, sometimes (but not always) involving the use **computational tools**.

When there are tools, the **interface** is often optimized to invite broad sets of students to access it.

Intelligent Tutoring Systems

The complexity of a skill

Solve this algebra problem:

What is x?

$$2.3x = 23.44 + 19.01$$

What are the steps you had to follow to solve this problem?

2.3*x = 23.44 +19.01	realize you can add numbers on RHS
2.3*x = 42.45	and that you can divide by a coefficient
x = 18.46	read off the value of x

The complexity of a skill

So to recap: you needed to realize you could add constants, carry out the addition, recognize an opportunity to simplify by dividing, carry out a division, and read off the result.

A seemingly simple skill involves **recognition and application of a considerable number of simple procedures**.

While they are tacit to someone who has mastered them, they can be rather complex. For instance, how do the rules change when a summand is negative? Depends on which one... How does division change as the number of decimal points change? If you have to subtract, when do you invoke carry-over rules?

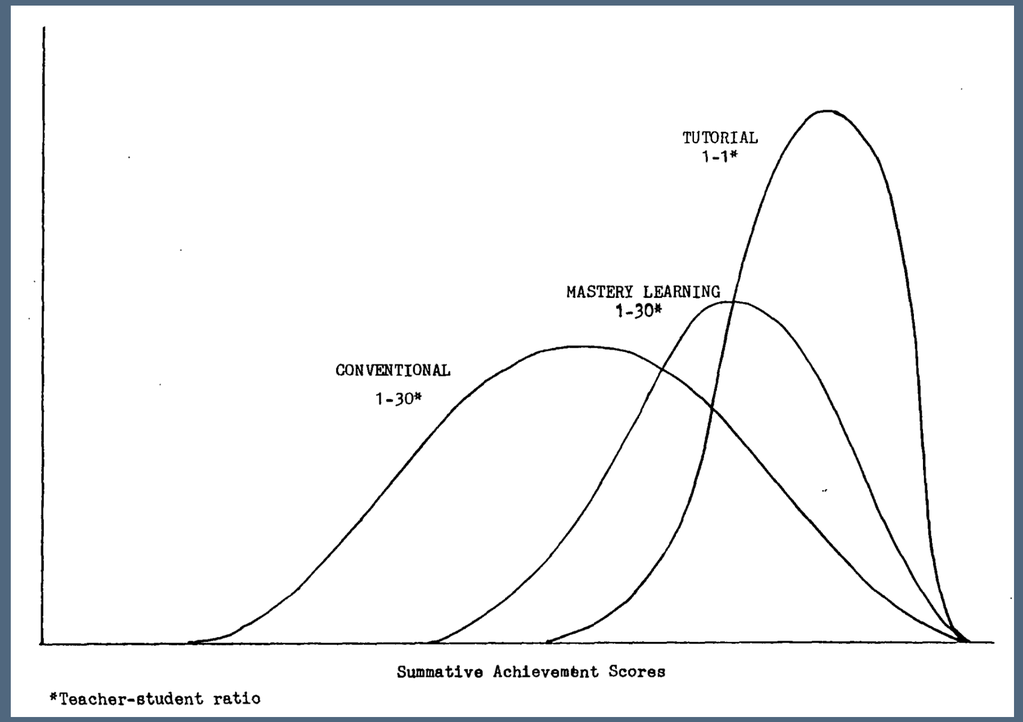
A skill like this is essential. How can we make sure everyone masters it?

Getting Two Sigmas Higher

[Bloom 1984]

Bloom reports on 6 studies from his lab that show that learners working with tutors **achieve "2 sigmas" better** than those in the conventional classroom (average tutored student performed better than 98% of classroom students).

This galvanized many in learning sciences and technology to strive to close the gap between classroom and tutor learning.

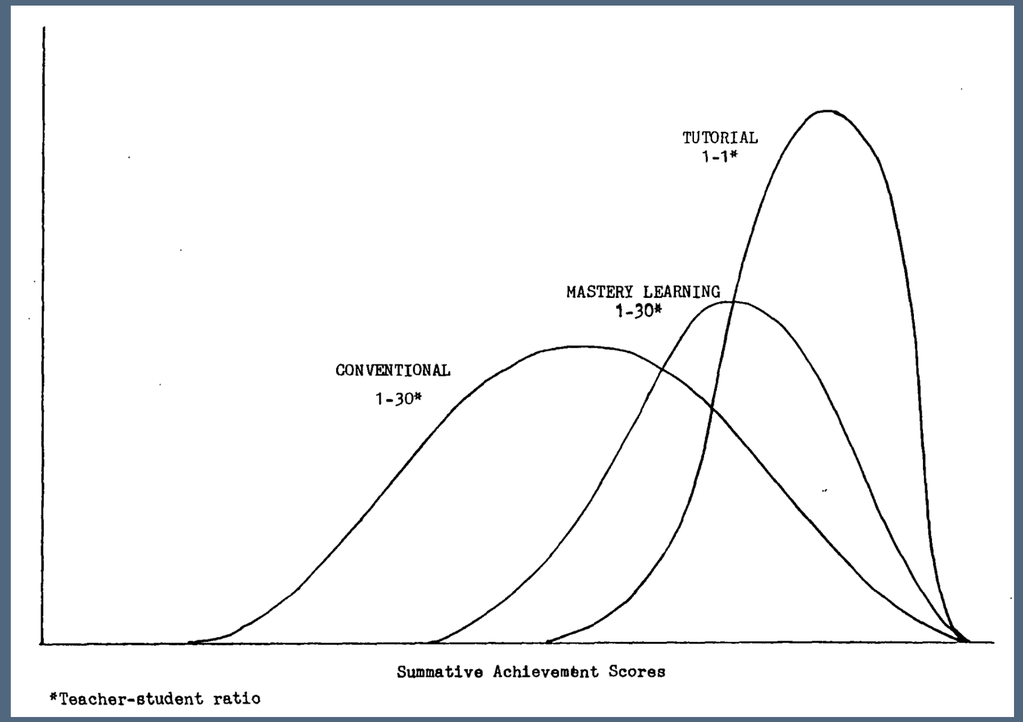


Getting Two Sigmas Higher

[Bloom 1984]

Bloom's group also tested classrooms that implemented **mastery learning**, where students were given feedback and assessments to check that students were competent in a skill before being taught new material.

This group saw a **1-sigma improvement**. Not as good as the tutoring group, but still better than the conventional classroom.

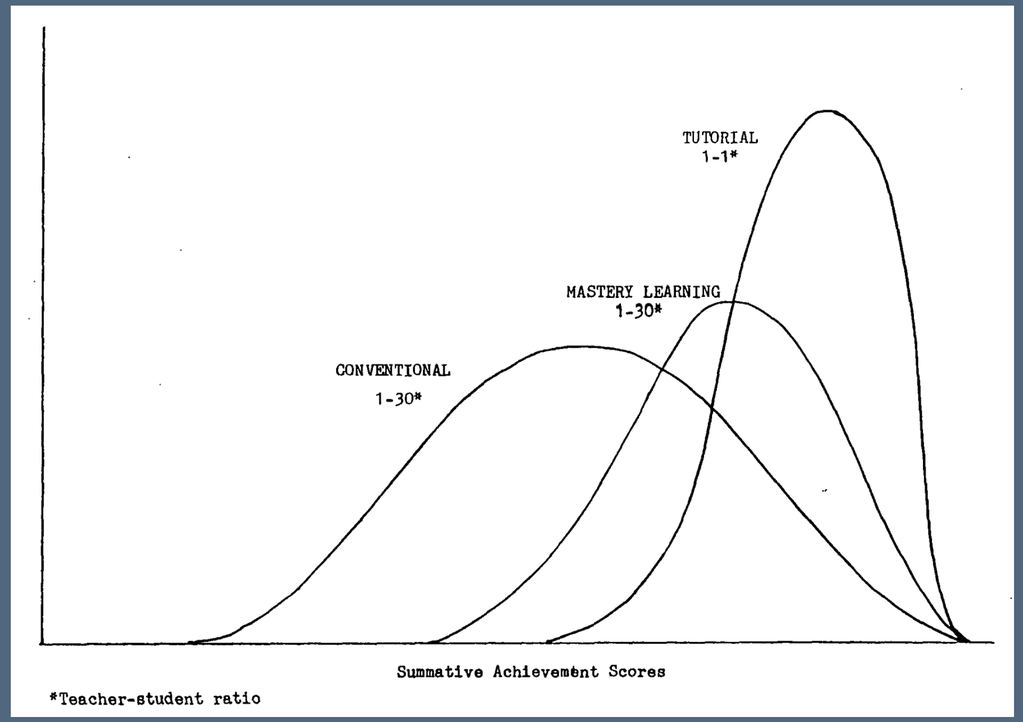


Getting Two Sigmas Higher

[Bloom 1984]

Could computers help close the gap between the conventional classroom and human tutoring?

Maybe computers they could serve as tutors. In the right conditions, they can perceive, communicate, and adapt, key traits of human tutors.



Intelligent Tutoring Systems

Intelligent tutoring systems are computer systems that provide interactive and adaptive coaching for learning a new skill.

It does this by **setting up exercises** and **providing helpful instruction**, **hints, and feedback** adapted to the learner.

Components of ITSs

[VanLehn 2006]

A learner has to develop **knowledge**. That knowledge has **components**. (think of the maneuvers involved in solving an algebra problem).

The learner is trained through two loops:

In the **outer loop**, the ITS assigns the learners tasks to help them learn.

In the **inner loop**, the ITS gives learners support in performing and mastering the steps involved in solving an individual problem.

Loops in ITSs

[VanLehn 2006]

Outer loop: Student just showed a misconception in an algebra problem. Assign them another problem with the same gotcha so that they can try to fix their misconception and solidify the correct knowledge.

Inner loop. The ITS sees a student take this step:

$$2+3x=20 \rightarrow 5x = 20$$

Inner loop pauses the user and probes them. Minimally it could ask "Was that step you just took appropriate?" Maximally, it could show a user what they should have done instead.

Why ITSs

[VanLehn 2011]

There have been some indications that ITSs can provide high-quality learing experiences. Van Lehn did a review in 2011 of 44 studies, finding that:

- ITSs regularly led other a **.76-sigma improvement** over conventional unaugmened learning situations.
    - This is not quite 2 sigmas, but...
- Van Lehn found that human tutors only achieved .79 sigmas in these studies! He raises the question of whether Bloom's famous 2 sigmas may be overinflated, and points out a gap in one of the initial studies' designs.
- .76 is quite close to .79

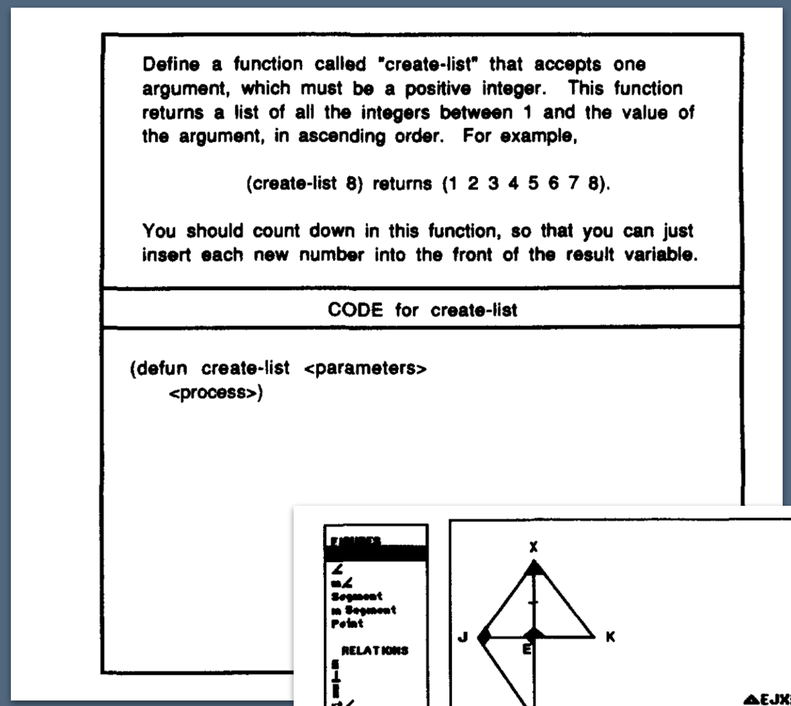
But at what cost?

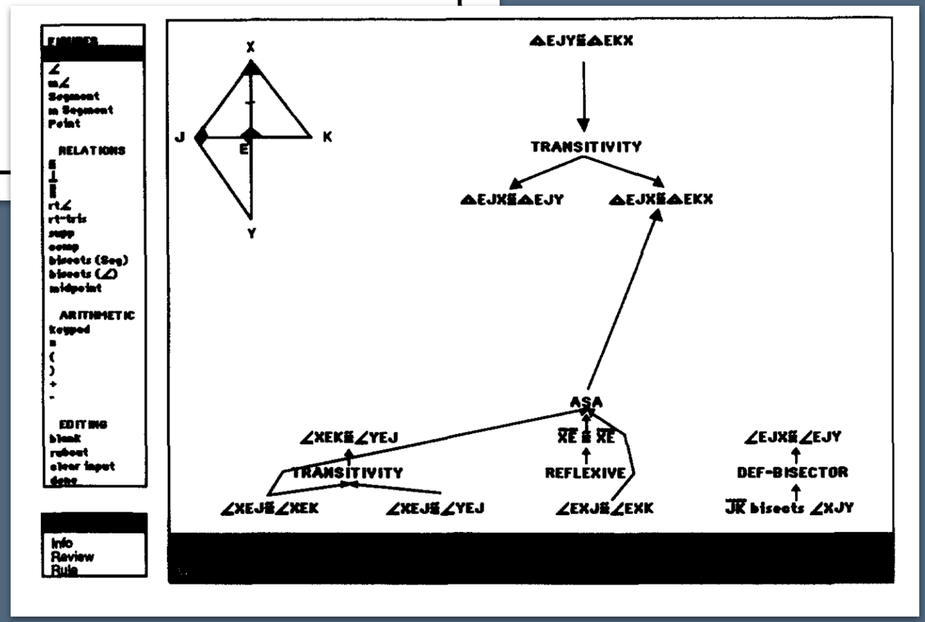
Cognitive tutors

[Anderson et al. 1995]

An influential line of work in intelligent tutoring systems that was based at Carnegie Mellon University, picking up steam in the '80s.

Included systems for programming, geometry, and algebra instruction, with promising results.





What makes a cognitive tutor?

[Anderson et al. 1995]

Cognitive skills are modeled by **production rules**, wherein a learner recognizes a situation ("if") and applies an action ("then").

IF	the goal is to prove two triangles are congruent
THEN	set as a subgoal to prove that corresponding parts are congruent.

IF	the goal is to get the second element of the list
THEN	code *car* and set as a subgoal to pass to car an argument that is the tail of the list.

Actions aren't just things the learner does, it is also updates to their conception of what is left to be done.

Production rules

[Anderson et al. 1995]

What is a production rule you would apply here?

What is x?

$$2.3x = 23.44 + 19.01$$

Did you include what was recognized, the physical action, and the update to the goal?

Give me an example of *another* valid production rule that could apply.

Give me an example of an *incorrect* production rule.

What makes a cognitive tutor?

[Anderson et al. 1995]

A learner learns a skill when they have mastered the relevant **production rules** and **declarative knowledge**.

The tutor adapts instruction by **modeling** what production rules the learner has (both correct an incorrect), providing correction and practice towards mastery of the correct rules.

This requires **inferring** what the learner knows from their actions.

The burden of tracing knowledge

[Anderson et al. 1995]

ITSs can require the externalization of quite a bit of work.

TABLE 1
A Teletype Depiction of a Hypothetical Student Completing the Code for the Function create-list With the Tutor

The Problem Description

Define a function called "create-list" that accepts one argument, which must be a positive integer. This function returns a list of all the integers between 1 and the value of the argument, in ascending order. For example:

(create-list 8) returns (1 2 3 4 5 6 7 8).

You should count down in this function, so that you can just insert each new number into the front of the result variable.

Cycles in which the tutor communicates with the student.

1. (defun create-list **(num)**
(loop)

You will need local variables in this function. So before you code anything else, there is a particular LISP operator you need to code.

2. (defun create-list (num)
(let (( )
<BODY>))

Which of the following do you want to do?
PRESS: IF YOU WANT TO
1 Initialize a variable to hold the list of digits between 1 and the value of the parameter.
2 Initialize a counter variable to keep track of how many times we go around the loop.
3 Have the tutor choose.

3. (defun create-list (num)
(let **((count 1)**
<OTHER-INITIALIZATIONS>)
<BODY>))

In this problem we won't want to start counting at the same value each time we call the function, so we can't initialize the counter to a constant like 1. How is the starting value being communicated to the function?

4. (defun create-list (num)
(let ((count **num)**
(numlist (list num)))
(lop))

Did you mean loop (y or n)? **y**

5. (defun create-list (num)
(let ((count num)
(numlist (list num)))
(loop
(cond ((equal count 1) numlist))
<LOOP-ACTIONS2>)))

You do want to return the variable "numlist" here, but just typing it into the code won't accomplish that. What LISP operator will allow you to return something from a LOOP?

What is in (and not in) a cog tutor?

[Anderson et al. 1995]

What of a learner does a cognitive tutor model?

What does it not model?

What does this mean for what it can help people learn and how effectively it can help them learn?

ITS for physical tasks

[Ren et al. 2025]

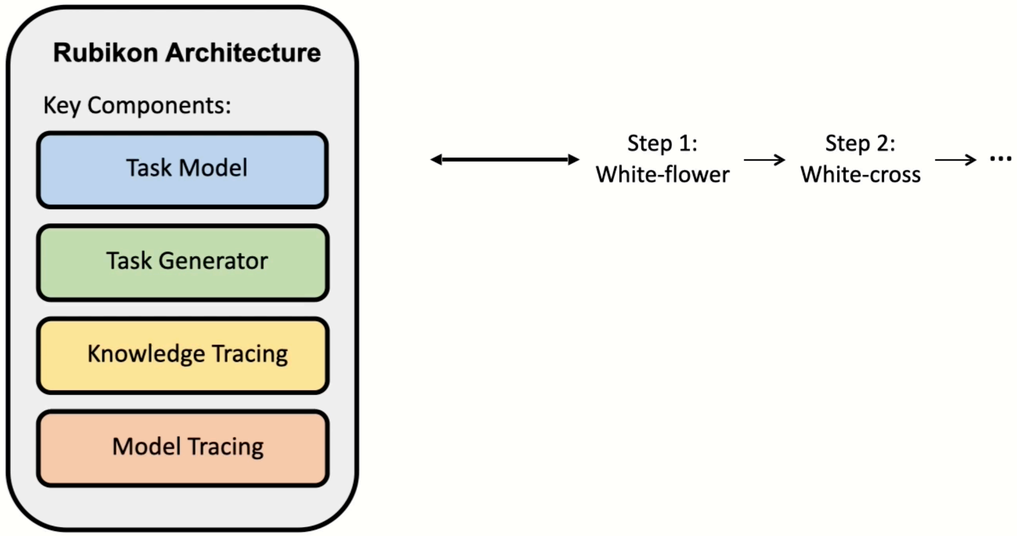
Outer loop? Choice of starting configurations (no setup needed!!)

Inner loop? Listening and responding to user moves.

Feedback? Greying out irrelevant squares. Point out next actions.

Model of learner? Kinds of puzzles they have been able to do.

Rubikon: Intelligent Tutoring for Rubik's Cube Learning Through AR-enabled Physical Task Reconfiguration



Affect and social dimensions

[Lee et al. 2011]

Gidget, teaching programming through a debugging game.

There are a series of **levels** covering different concepts. **Feedback** is given throughout.

The message design personalizes the fault to the machine and away from the user. This led to less attrition among self-described programmers.

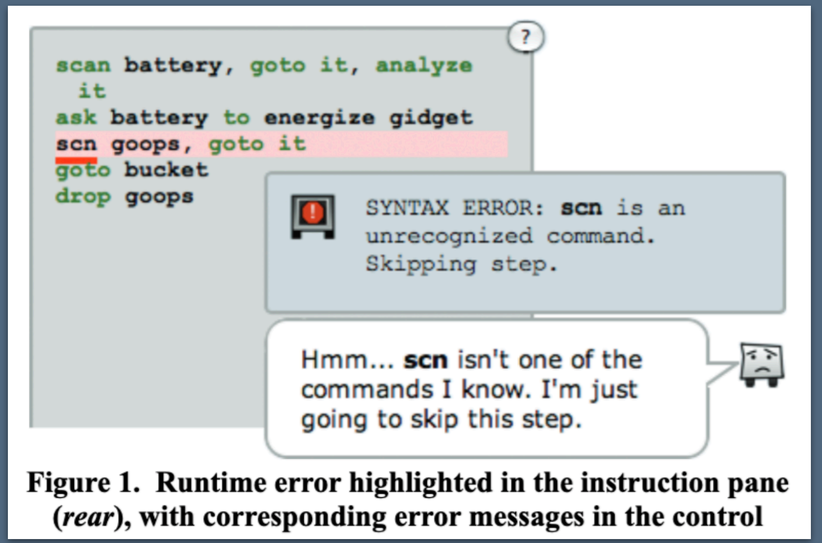


Figure 1. Runtime error highlighted in the instruction pane (*rear*), with corresponding error messages in the control

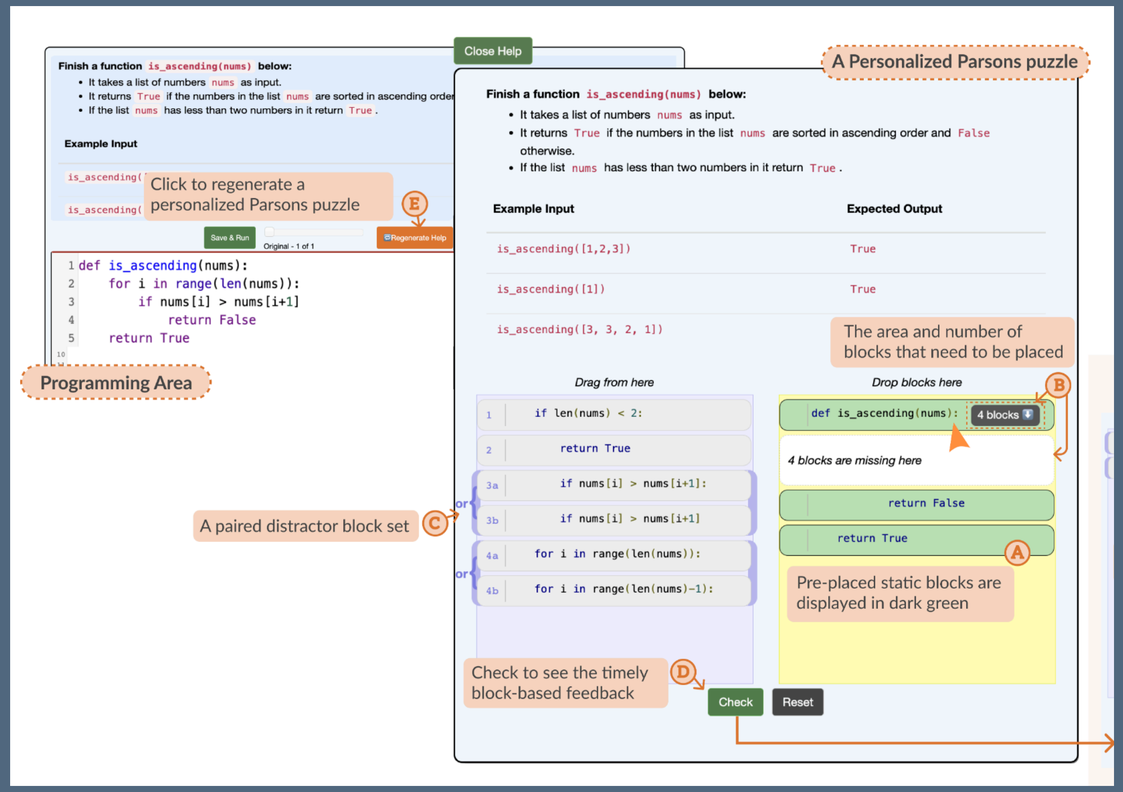
On-demand personalized drilling

[Hou et al. 2024]

Provide interactive puzzles as **on-demand scaffolding** following an error.

Generate these puzzles on demand. **Adapt** them to the user solution (basing them on their code) and generate distractors to make the problem difficult.

Correct lines of code are frozen in place to focus attention on incorrect lines.



Why isn't all learning in ITSs?

They are **expensive** to create. Anderson et al. estimate 10h per production rule supported. Weitekamp et al. describe 1h of instruction potentially taking 200-300h of prep.

They require a **complete formal model** of a skill.

And much of learning is in the **real world**, with real tasks, and real people. ITS might be able to be merged with these in some ways, though they have not been, at least in a sustainable and scalable way.

Constructionism

When learning is natural?

[Papert 1980]

How did you learn your first language?

How did you learn calculus?

What if learning of all subjects and across many stages of life was as effortless as childhood language acquisition?

What makes learning natural?

[Papert 1980]

Papert, MIT computer scientist, self-described mathematician, evangelist for math and technical literacy, draws a comparison to his own learning of technical foundations from play with gear toys.

His experimentation was part of the **natural landscape**.

They were part of the **world of the adults** in his life, so they could be used to relate to these adults.

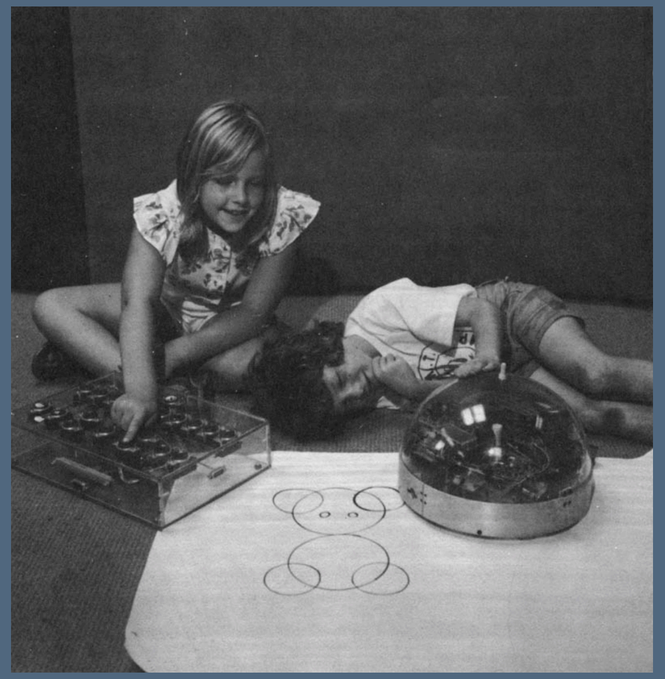
They served as a connection between the **abstract and personal**.

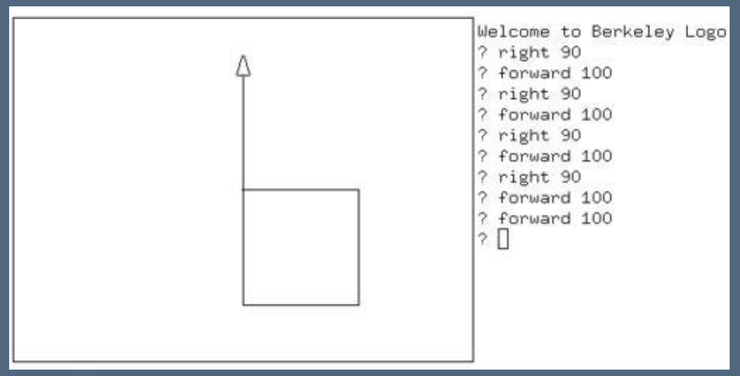
Objects-to-think-with

[Papert 1980]

Gears and the LOGO turtle are two examples.

One important feature is that these objects connect the abstract and the personal. You can develop a connection with them by thinking about what it is to be like them and projecting your experience onto them. And in interacting with, command, manipulating these objects, they **open the doors to powerful abstractions**.





Constructionism

[Papert 2008]

Draws from the construct**iv**ist theories in psychology where learning is considered **reconstruction rather than transmission** of knowledge.

Asserts that learning is the most effective when the activity involves the learner **constructing a meaningful product**.

Control and construction

[Papert 1980]

Under Paper's guidance, it would be...

the child programming the computer

not the computer programming the child

TABLE 1
A Teletype Depiction of a Hypothetical Student Completing the Code for the Function create-list With the Tutor

The Problem Description

Define a function called "create-list" that accepts one argument, which must be a positive integer. This function returns a list of all the integers between 1 and the value of the argument, in ascending order. For example:
(create-list 8) returns (1 2 3 4 5 6 7 8).

You should count down in this function, so that you can just insert each new number into the front of the result variable.

Cycles in which the tutor communicates with the student.

1. (defun create-list **(num)**
(loop)

You will need local variables in this function. So before you code anything else, there is a particular LISP operator you need to code.

2. (defun create-list (num)
(let (( )
<BODY>))

Which of the following do you want to do?
PRESS: IF YOU WANT TO
1 Initialize a variable to hold the list of digits between 1 and the value of the parameter.
2 Initialize a counter variable to keep track of how many times we go around the loop.
3 Have the tutor choose.

Constructionism at scale

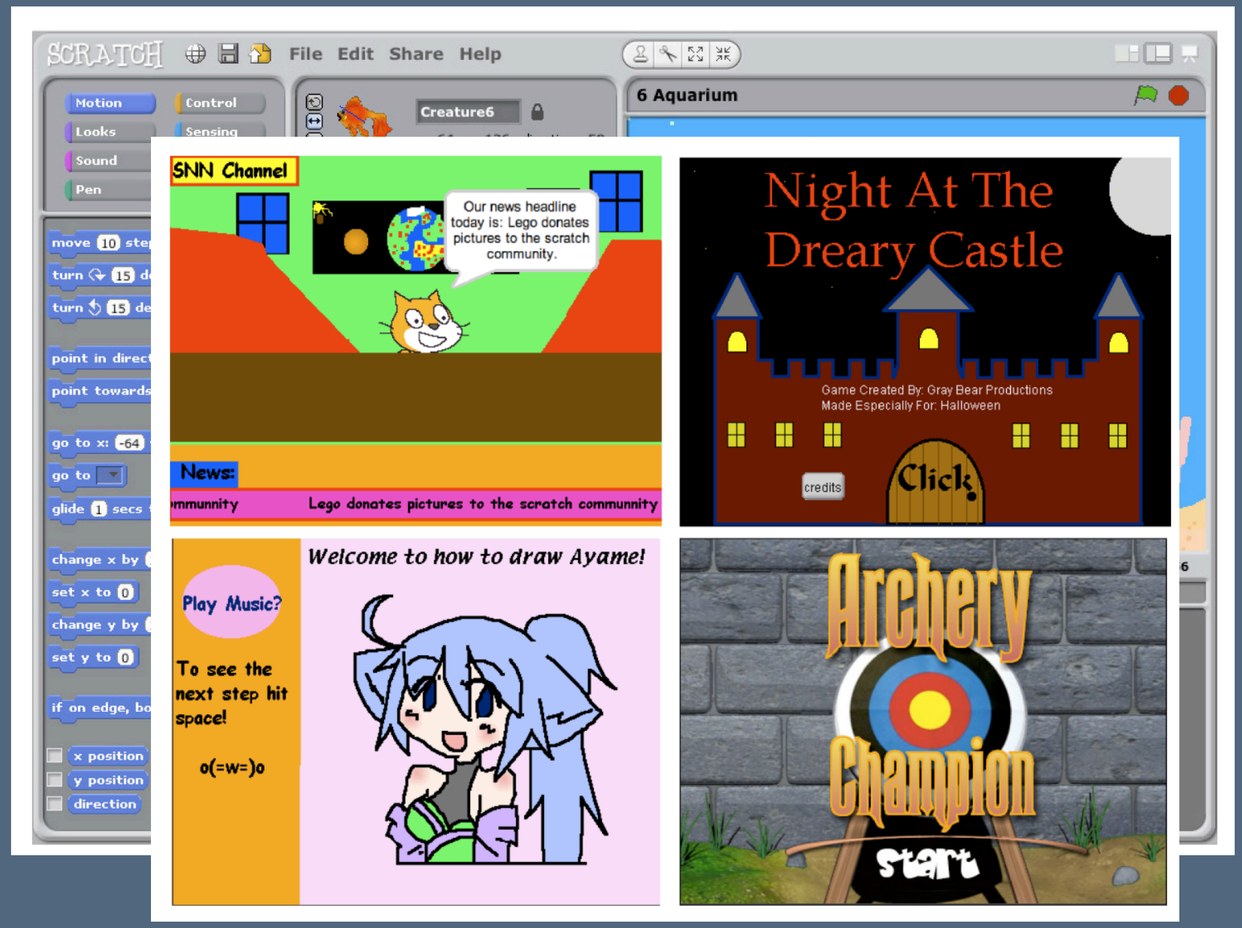
[Maloney et al. 2010]

Scratch programming environment.

In the world: with visuals and audio kids care about; often for and among peers.

Control: kids make a variety of kinds of experiences.

Encountering the abstract: coding, message passing.

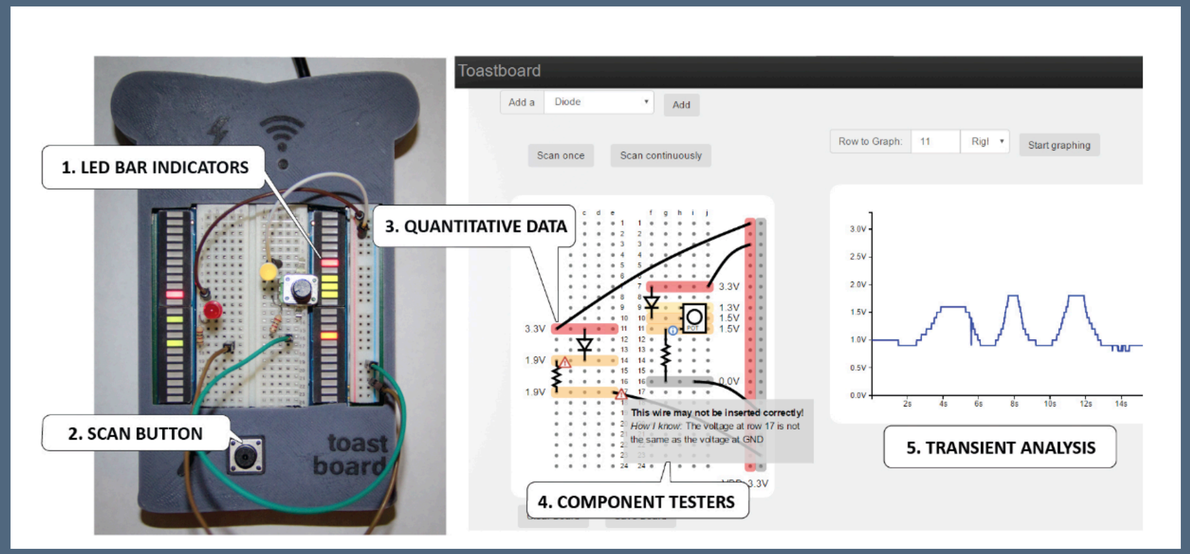


Constructionism for grownups

[Drew et al. 2016]

ToastBoard, a bread board that supports debugging electronics prototypes during interactive device design.

Easing **learning through making**. Scaffold **formulation and testing of hypotheses** around a complex technical artifact.

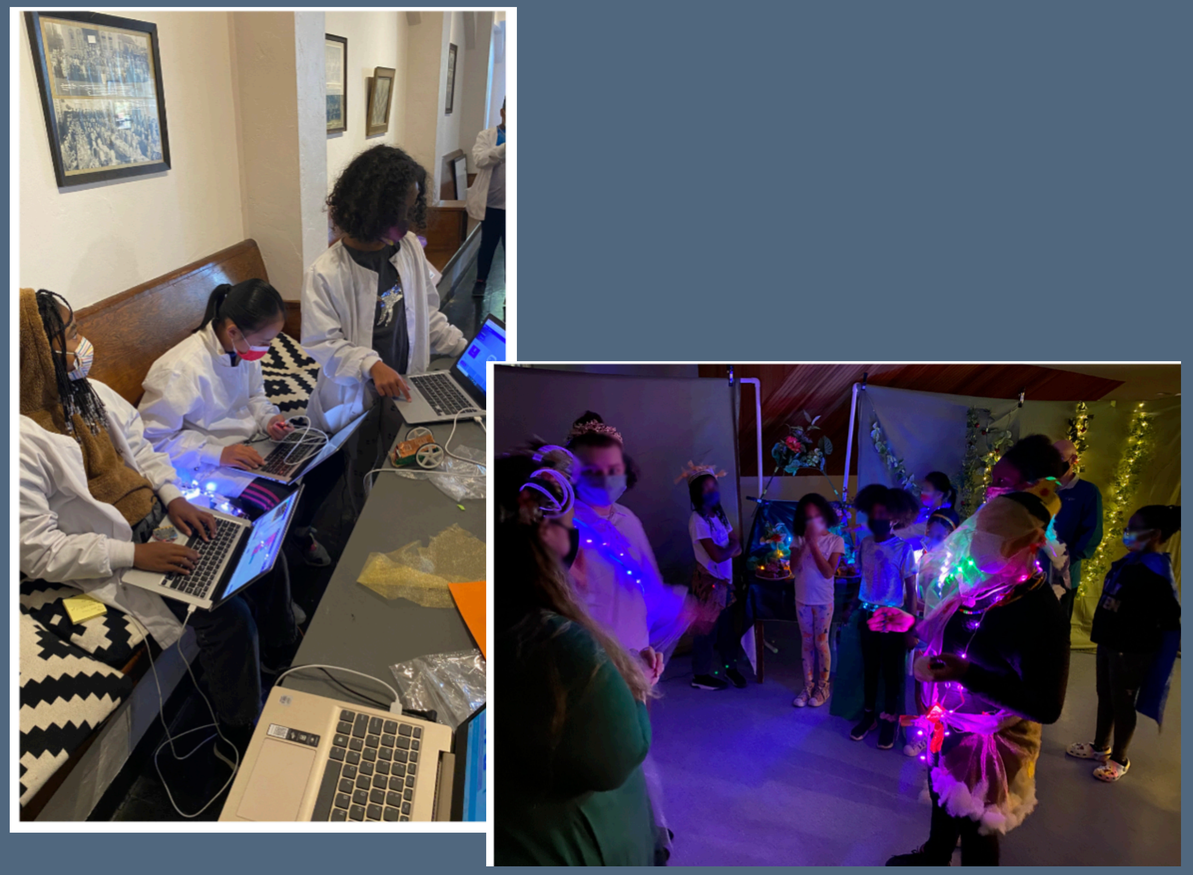


Socials, wearables, and STEAM

[Fey et al. 2025]

Encouraging skill development through **construction to support social play**.

In a social wearables Edu-LARP, students create and program teir own wearables and then take them with them on fantasy missions.

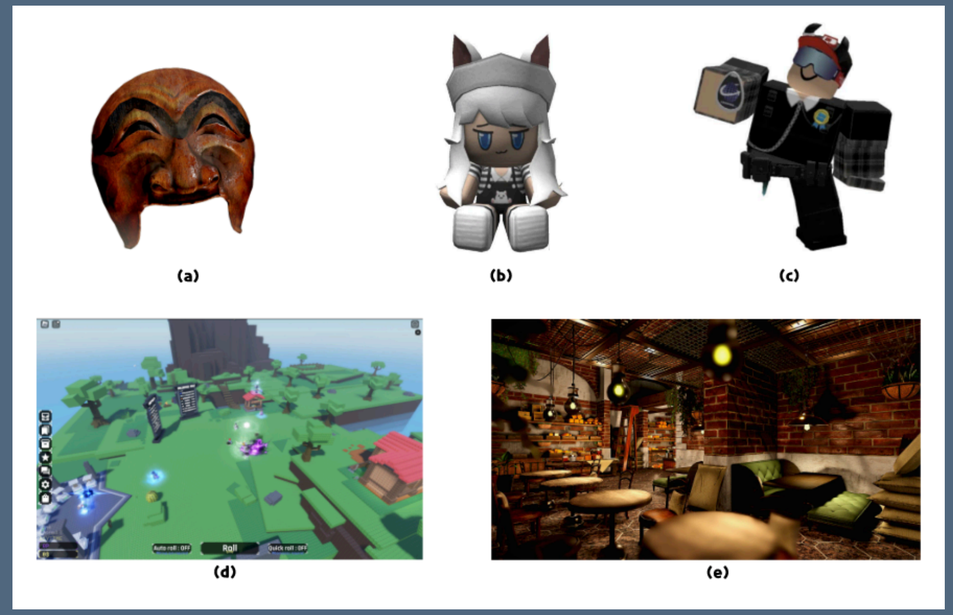


Studying constructionist spaces

[Choi et al. 2025]

Platforms for construction abound. One modern platform is Roblox, which hosts more than 40 million games.

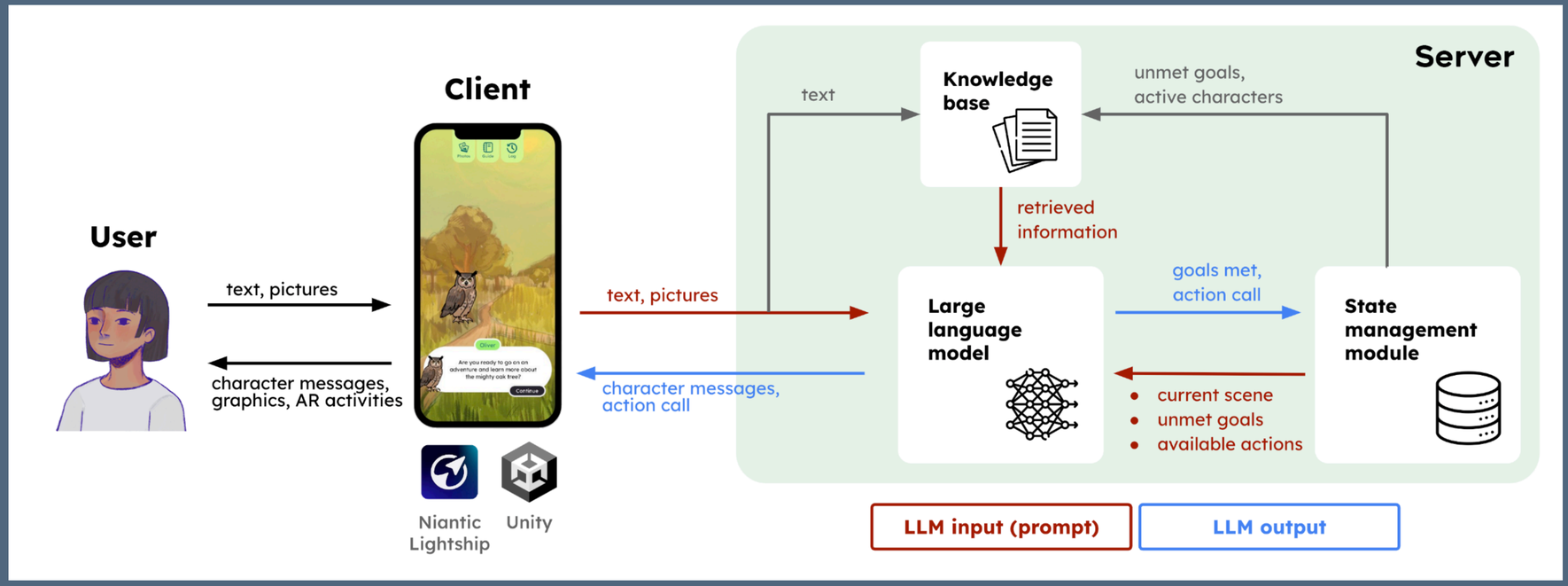
Teens on the platform sometimes join into online communities, where **they teach each other** and build connections.



ITS + constructionism?

[Cheng et al. 2025]

Oak Story, a **AR game** with fixed **learning goals** and **adaptive sequencing and depth** of content.



YOU READ THIS

Summary

When designing for learners, we should adopt a **learner-centered design** perspective that foregrounds learning needs and uses **scaffolding** to teach task and improve interaction with tools.

For complex tasks where the cognitive model is recoverable, we can provide **intelligent tutoring systems** that finely **adapt sequence and form** of instruction to learners' knowledge.

To facilitate natural learning, we can nurture **constructionist experiences** with rich **objects-to-think-with** and environments and communities that let learners make things they want to.

References

Anderson, John R., Albert T. Corbett, Kenneth R. Koedinger, and Ray. Pelletier. 1995. "Cognitive Tutors: Lessons Learned." Journal of the Learning Sciences 4 (2): 167–207. https://doi.org/10.1207/s15327809jls0402_2.

BLOOM, BENJAMIN S. 1984. "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring." Educational Researcher 13 (6): 4–16. https://doi.org/10.3102/0013189X013006004.

Cheng, Alan Y., Carolyn Q. Zou, Anthony Xie, et al. 2025. "Oak Story: Improving Learner Outcomes with LLM-Mediated Interactive Narratives." Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology, September 28, 1–17. https://doi.org/10.1145/3746059.3747698.

Choi, Yubin, Jeanne Choi, and Joseph Seering. 2025. "Leveling Up Together: Fostering Positive Growth and Safe Online Spaces for Teen Roblox Developers." Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (New York, NY, USA), CHI '25, April 25, 1–18. https://doi.org/10.1145/3706598.3713969.

Drew, Daniel, Julie L. Newcomb, William McGrath, Filip Maksimovic, David Mellis, and Björn Hartmann. 2016. "The Toastboard: Ubiquitous Instrumentation and Automated Checking of Breadboarded Circuits." Proceedings of the 29th Annual Symposium on User Interface Software and Technology (New York, NY, USA), UIST '16, October 16, 677–86. https://doi.org/10.1145/2984511.2984566.

Fey, James Collin, Raquel B. Robinson, Chen Ji, et al. 2025. "Social Wearables Edu-Larp: Insights From a Novel Camp Combining Crafting, Coding, and Larping Aimed at Non-Traditional Steam Participants." Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (New York, NY, USA), CHI '25, April 25, 1–15. https://doi.org/10.1145/3706598.3713843.

References

Hou, Xinying, Zihan Wu, Xu Wang, and Barbara J. Ericson. 2024. "CodeTailor: LLM-Powered Personalized Parsons Puzzles for Engaging Support While Learning Programming." Proceedings of the Eleventh ACM Conference on Learning @ Scale, July 9, 51–62. https://doi.org/10.1145/3657604.3662032.

Kulik, James A., and J. D. Fletcher. 2016. "Effectiveness of Intelligent Tutoring Systems: A Meta-Analytic Review." Review of Educational Research 86 (1): 42–78. https://doi.org/10.3102/0034654315581420.

Maloney, John, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. "The Scratch Programming Language and Environment." ACM Trans. Comput. Educ. 10 (4): 16:1-16:15. https://doi.org/10.1145/1868358.1868363.

Papert, Seymour. 1980. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Inc.

Ren, Haocheng, Muzhe Wu, Gregory Thomas Croisdale, Anhong Guo, and Xu Wang. 2025. "Rubikon: Intelligent Tutoring for Rubik's Cube Learning Through AR-Enabled Physical Task Reconfiguration." Proceedings of the 2025 ACM Designing Interactive Systems Conference (New York, NY, USA), DIS '25, July 4, 3549–62. https://doi.org/10.1145/3715336.3735788.

Soloway, Elliot, Mark Guzdial, and Kenneth E. Hay. 1994. "Learner-Centered Design: The Challenge for HCI in the 21st Century." Interactions 1 (2): 36–48. https://doi.org/10.1145/174809.174813.

VanLehn, Kurt. 2006. "The Behavior of Tutoring Systems." Int. J. Artif. Intell. Ed. 16 (3): 227–65.

VanLehn, Kurt. 2011. "The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems." Educational Psychologist 46 (4): 197–221. https://doi.org/10.1080/00461520.2011.611369.

Weitekamp, Daniel, Erik Harpstead, and Ken R. Koedinger. 2020. "An Interaction Design for Machine Teaching to Develop AI Tutors." Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, April 21, 1–11. https://doi.org/10.1145/3313831.3376226.