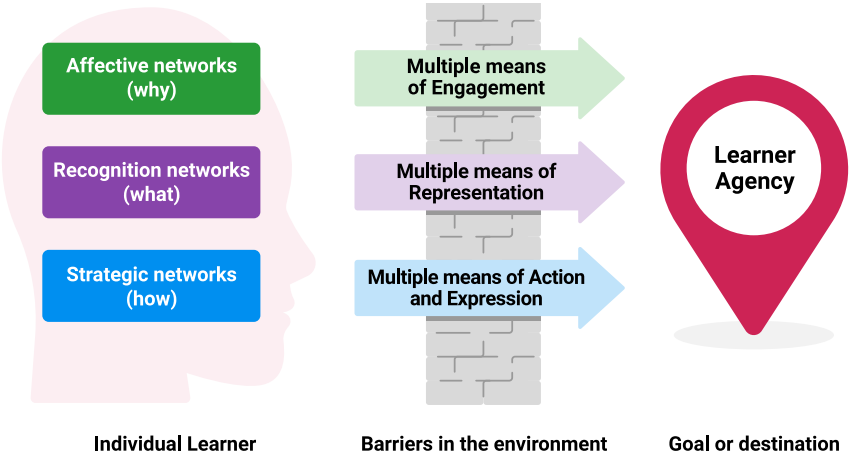
Raspberry Pi Foundation

Pedagogy Quick Read

Universal design for learning: Reaching all learners in CS education

Universal design for learning (UDL) is a research-based framework. Rooted in neuroscience, the primary goal of UDL is to build learner agency by reducing barriers to learning and implementing flexible instruction. A growing body of research suggests that the application of UDL in K–12 computer science (CS) contexts provides opportunities to reach the widest range of learners, especially those with disabilities.[1, 2, 3]



UDL in CS education

Universal design for learning is a foundational framework for making K–12 computer science accessible for all learners, especially those with disabilities. Research increasingly demonstrates the effectiveness of UDL across K–12 education, highlighting its potential in CS. Traditional CS instruction often relies on rigid, uniform methods that limit flexibility, while constructionist approaches may lack the necessary structure and scaffolding. UDL instead offers an approach that can incorporate the benefits of both approaches. For instance, some approaches to teaching through block-based programming can exclude some learners, while fast-paced or abstract computing tasks pose challenges for others. These barriers underscore the need for flexible and individualized learning in CS. Additionally, studies revealed educators (e.g., general, special education, computer science, coaches, etc.) need more professional development to implement UDL in a K–12 CS context effectively.[1] Robust resource creation and sustained professional learning are a top priority in the K–12 space.[3]

The myth of the average learner

Assuming all learners can be taught the same material oversimplifies the complexity of learning and creates barriers. Contributor to the original UDL framework, Todd Rose, dispels the idea of an "average" learner and affirms a variability, or scale, to which learners participate based on cognitive, social-emotional, family background, and academic factors.[2] This is known as learner variability. Rose's analogy that one-size shoes cannot fit everyone is the same for instruction. Whether in a CS or general education setting, understanding learner variability is essential to reducing barriers and meeting the needs of all learners.

Summary

The framework (UDL 3.0) from the Center for Applied Special Technology (CAST) describes '**learner agency**' as the ability to actively participate in making choices relating to their own learning goals. Below are the three main categories of UDL:

Multiple means of engagement

- The "why of learning" emphasizes the importance of learners' curiosity, motivations, sustained effort, and goals to keep them engaged
- Providing learners with choice, feedback, and clearly communicated expectations in how they participate creates a sense of ownership
- Engaging learners can boost their motivation and help them stick with challenges, leading to a deeper understanding

Multiple means of representation

- The "what of learning" presents information in multiple ways with the goal of clarifying language, explaining symbols, and clearly introducing new terms
- Presenting content through visuals, audio, text, and hands-on activities ensures all learners can access and understand the material in ways that address their needs

Multiple means of action and expression

- The "how of learning" offers learners the ability to express their understanding in a variety of ways through the use of accessible and assistive technologies
- Focus on goal setting, progress monitoring, and supporting executive functioning, helping learners demonstrate their understanding in ways that work for them

CS pedagogies aligned with UDL

To support effective instruction in computational thinking and programming, several key research-supported CS pedagogies (see table) are valuable for teachers to leverage. These pedagogies also relate to the UDL 3.0 guidelines, helping teachers understand how and why these strategies address learner agency.

Pedagogy	Description	Connection to UDL guidelines
Use-Modify-Create[4]	A three-stage progression for developing computational thinking and supports learner agency. Learners: 1. Use existing creations (e.g., run a program) 2. Modify the program 3. Create their own program	**Engagement:** Gives learners prompts that guide them in knowing when and how to ask for help **Representation:** Provides information incrementally, such as by using sequential highlighting, to scaffold understanding **Action & expression:** Provides faded scaffolding that allows learners to become more independent over time
TIPP & SEE (Title-Instructions-Purpose-Play & Sprites-Events-Explore)[5]	Presented as a metacognitive approach using a mnemonic to support learners in planning how to approach programming tasks in Scratch. Here, metacognition refers to the process of learners becoming more aware of how they learn.	**Engagement:** Provides a structured approach to a problem and encourages learners to reflect **Representation:** Explicit support for decoding text and symbols and guiding information processing **Action & expression:** Helps learners plan for solving problems when programming, and supports monitoring
Multiple entry points[2, 6]	Offering varied challenges and scaffolding through different project versions. • Remix/play code (e.g., a worked example) • Buggy code (e.g., identify and fix known errors) • Exploded code (e.g., Parsons problems) • Spicy expansions (e.g., extension challenges)	**Engagement:** Varied approaches can appeal to different learner interests and foster collaboration **Representation:** Gives learners multiple views of a problem, providing opportunities for perception and comprehension **Action & expression:** Differing activities enables learners to express their understanding in different ways
Debugging[7]	Use of metacognitive strategies to find errors in the program. • Collaborative problem solving • Tinkering • Rubber ducking • Thinking aloud	**Engagement:** Approaching authentic problems by debugging and using different strategies to solve problems **Representation:** State and reinforce lesson goals and learning objectives with the use of "I can" statements **Action & expression:** Gives options for expression and communication. Aids executive functions of planning and problem solving.

Moving towards "for all"

While the landscape of CS education continues to evolve, particularly with the rise of emerging technologies, the goal remains the same: to provide high-quality computing education to the widest range of learners, regardless of race, gender, class, or ability. A step towards reaching this goal includes considering learning instruction and environments that are accessible and universally designed from the outset, rather than retrofitting. Creating flexible and meaningful learning opportunities provides learners with the opportunity to demonstrate knowledge, purpose, and authentically engage limitlessly in CS contexts.

References

[1] Israel, M. et al. (2022). Equity and Inclusion through UDL in K–6 Computer Science Education: Perspectives of Teachers and Instructional Coaches. the-cc.io/qr26_3
[2] Sentance, S. et al. (2023). Computer Science Education: Perspectives on Teaching and Learning in School. (Chapter 10) the-cc.io/qr26_4
[3] Cobo, A. E. (2023). Creating Pathways to Inclusion in K–12 Computer Science Education: A Case Study on the Scratch Educator Meetup. the-cc.io/qr26_5
[4] Lee, I. et al. (2011). Computational thinking for youth in practice. the-cc.io/qr26_6
[5] Salac, J. et al. (2020). TIPP&SEE: A Learning Strategy to Guide Students through Use–>Modify Scratch Activities. the-cc.io/qr26_7
[6] Barrett, J., and Israel, M. (2023). Scaffolding Block Coding Through Multiple Entry Points. the-cc.io/qr26_8
[7] Salgarayeva, G., and Makhanova, A. (2024). Making Computer Science Accessible through Universal Design for Learning in Inclusive Education. the-cc.io/qr26_9

Raspberry Pi Foundation