CREATIVE TECHNOLOGY RESEARCH LAB | COLLEGE OF EDUCATION UNIVERSITY OF FLORIDA

Project TACTIC: Teaching All Computational Thinking through Inclusion and Collaboration

TACTICal Teaching Brief

Helpful Strategies for **Utilizing the Universal Design For Learning Framework** in Computer Science Education

Introduction

The Universal Design for Learning (UDL) framework provides guidelines that can help teachers proactively plan for the academic diversity present in classrooms. This is especially important when it comes to CS education due to the historic underrepresentation of women, people from different cultural backgrounds, and people with disabilities in CS fields. If we can think about ways of meaningfully engaging ALL learners through the UDL framework, then we can address the barriers that are inherent in my CS activities.

UDL is a proactive approach to planning of curricular opportunities. A UDL approach takes into account learner variability when considering goals, methods of instruction, assessments and materials. It is organized around the principles of providing students with multiple means of representation, expression and engagement with their learning.

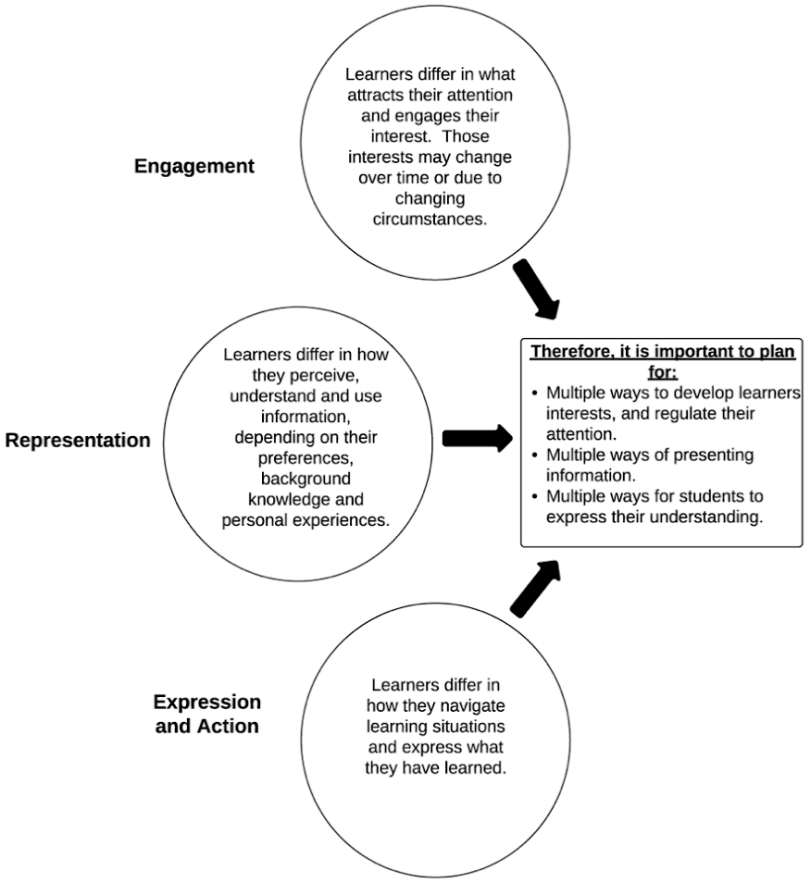
Scenario

Mr. Gibson is going to be teaching CS within his mathematics instruction this year. At the beginning of the school year, he attended a school-wide professional development (PD) workshop on Universal Design for Learning. In this workshop, Mr. Gibson learned about how the three UDL principles could be used in planning instruction that is engaging and accessible to all his students, including those with disabilities. He wonders how he might apply these principles to the CS activities that will take place this year with all of his students, including three students with disabilities who are included in his 3rd grade class.

1. **Rachel** has a learning disability related to math;
2. **Roberto** has a social communication disorder that kept him from verbally expressing his needs; and
3. **Connie** has an emotional behavior disorder as well as a speech/language impairment and often does not interact with her peers.

Mr. Gibson also realizes that beyond the needs of these three students, his class has a lot of academic, social, and cultural diversity. Meeting each of the students' individual needs will be a challenge this year! In thinking about the PD he attended this summer and the students in this class, he wonders how UDL can help him plan his integrated math and CS lessons in a way that will engage all his students, including Rachel, Roberto, and Connie. Because Ms. Gomez, the special education teacher, helped lead this PD, Mr. Gibson decided to meet with her to brainstorm some ideas about how UDL might be leveraged to provide greater opportunities for success for all learners, including those with disabilities.

Figure 1: The Three Principles of UDL
Adapted from: **National Center on Universal Design for Learning: The Three Principles of UDL**



Do any of these challenges sound familiar?
Can you relate to any of these?

Common Challenges

- Teachers believe that UDL is important, but do not know where to start or how to find time to plan in this manner. It looks really time consuming.
- Although UDL is intended to address whole-class instruction, it is unclear how individual student needs fit within this framework.
- UDL seems to make sense conceptually, but it's hard to see how it would apply in CS activities.

Universal Design for Learning within Computer Science Education

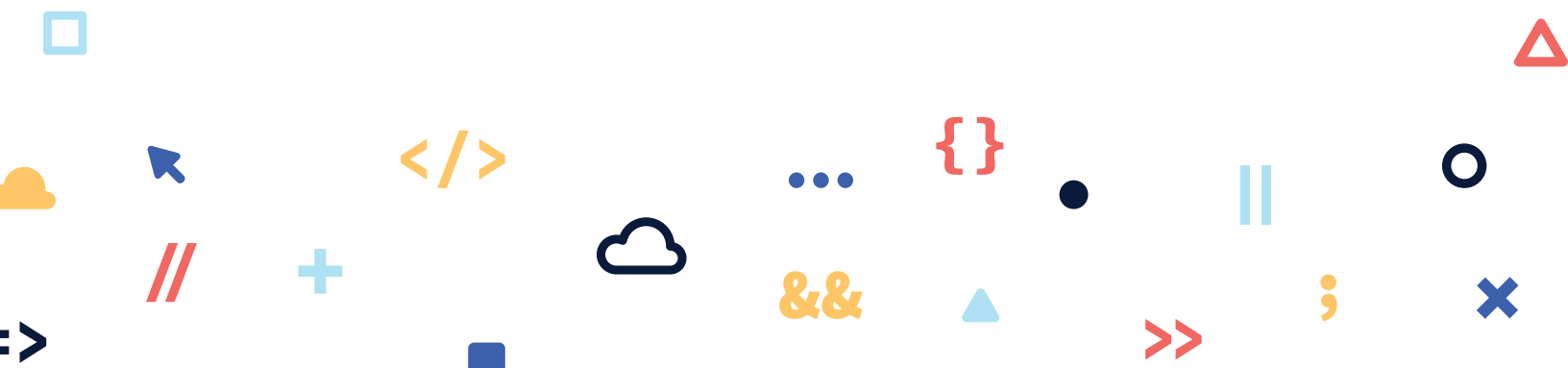
Figure 2: Universal Design for Learning within Computer Science Education

Multiple Means of **Representation**	Multiple Means of **Action and Expression**	Multiple Means of **Engagement**
Provide options for perception	*Provide options for physical action*	*Provide options for recruiting interest*
• Model computing using physical representations as well as through an interactive whiteboard, videos • Give access to modeled code while students work independently • Provide access to video tutorials of computing tasks • Select coding apps and websites that allow the students to adjust visual settings (such as font size & contrast) and that are compatible with screen readers	• Provide teacher's codes as templates • Include CS Unplugged activities that show physical relationship of abstract computing concepts • Use assistive technology including larger/smaller mice, touch-screen devices • Select coding apps and websites that allow coding with keyboard shortcuts in addition to dragging & dropping with a mouse	• Give students choices (choose project, software, topic) • Allow students to make projects relevant to culture and age • Minimize possible common "pitfalls" for both computing and content • Allow for differences in pacing and length of work sessions • Provide options to increase or decrease sensory stimulation (for example listening to music with headphones or using noise cancelling headphones) • Allow for differences in pacing and length of work sessions

Provide options for language mathematical expressions, and symbols	*Provide options for expression and communication*	*Provide options for sustaining effort and persistence*
• Teach and review content specific vocabulary • Teach and review computing vocabulary (e.g., code, animations, computing, algorithm) • Post anchor charts and provide reference sheets with images of blocks or with common syntax when using text	• Give options of unplugged activities and computing software and materials (e.g., Pseudocode, Scratch, code.org, Alice) • Give opportunities to practice computing skills and content through projects that build prior lessons • Provide sentence starters or checklists for communicating in order to collaborate, give feedback, and explain work • Create physical manipulatives of commands, blocks or lines of code • Provide options that include starter code	• Remind students of both computing and content goals • Provide support or extensions for students to keep engaged • Teach and encourage peer collaboration by sharing products • Utilize pair programming and group work with clearly defined roles • Discuss the integral role of perseverance and problem solving in computer science. • Recognize students for demonstrating perseverance and problem solving in the classroom.

Provide options for comprehension	*Provide options for executive functions*	*Provide options for self-regulation*
• Activate background knowledge by making computing tasks interesting and culturally relevant • State lesson content/ computing goals • Encourage students to ask questions as comprehension checkpoints • Use relevant analogies and make cross-curricular connections explicit (for example comparing iterative product development to the writing process) • Provide graphic organizers for students to "translate" programs into pseudocode	• Guide students to set goals for long-term projects • Record students' progress (have planned checkpoints during lessons for understanding and progress for computing skills and content) • Provide exemplars of completed products • Embed prompts to stop and plan, test, or debug throughout a lesson or project. • Provide graphic organizers to facilitate planning, goal-setting, and debugging • Provide explicit instruction on skills such as asking for help, providing feedback, and using problem solving techniques • Demonstrate debugging with think-alouds	• Communicate clear expectations for computing tasks, collaboration, and help seeking • Develop ways for students to self-assess and reflect on own projects and those of others • Use assessment rubrics that evaluate both content and process • Break-up coding activities with opportunities for reflection such as turn and talks or written questions • Acknowledge difficulty and frustration. Model different strategies for dealing with frustration appropriately

Israel, M., Lash, T., Ray, M. (2017). Universal Design for Learning within Computer Science Education. Creative Technology Research Lab. University of Illinois.



Strategies

- Start small, utilizing UDL principles in one lesson or unit at a time.
- Begin planning by thinking about what is most important in the unit/lesson and then what would make that content difficult for your students. By focusing first on barriers to learning, you can begin to isolate which checkpoints in the UDL framework to begin with.
- Don't attempt to do all the checkpoints in the entire UDL framework. Start with one or two UDL checkpoints and build up to a realistic number. More isn't always better!
- Reflect on how UDL works in other content areas. For example, if multiple means of representation in mathematics means that you provide students with options for manipulatives and the use of video to reinforce learning, these same strategies can be used in CS education by using Unplugged activities and worked video examples.
- Consider Goals, Environment, Materials, and Assessment (see example lesson plan)
- Use the table (Figure 2) and lesson plan template we provided for examples to launch this work. (Note: It's not an exhaustive table.)

Summary

Mr. Gibson is excited about utilizing a UDL approach when planning for his computer science lessons for his students. With Ms. Gomez's help, him saw that starting small can make integrating a UDL approach feasible and that planning for learner differences ahead of time can provides benefits for all of the students in his class including those with disabilities.

Funding for this research was provided by the National Science Foundation (award #1639837). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the NSF.

For More Information, please contact:
Maya Israel at misrael@coe.ufl.edu

Contributing authors to this TACTICal Brief:
Todd Lash, Gakyung Jeong, Quinn Wherfel, and Maya Israel