COMPUTATIONAL THINKING

Over the past several years, we have been captivated by "computational thinking" as a way to describe the learning and development that take place with Scratch. In this section, we share: (1) our definition of computational thinking as a set of concepts, practices, and perspectives, (2) an instrument for assessing student proficiency with computational practices, and (3) a self-reflection instrument to help teachers assess how they support computational practices in the classroom.

These definitions and instruments were developed in collaboration with Wendy Martin, Francisco Cervantes, and Bill Tally from Education Development Center's Center for Children & Technology, and Mitch Resnick from MIT Media Lab. Additional computational thinking resources are available at http://scratched.gse.harvard.edu/ct

COMPUTATIONAL CONCEPTS

CONCEPT	DESCRIPTION
sequence	identifying a series of steps for a task
loops	running the same sequence multiple times
parallelism	making things happen at the same time
events	one thing causing another thing to happen
conditionals	making decisions based on conditions
operators	support for mathematical and logical expressions
data	storing, retrieving, and updating values

COMPUTATIONAL PRACTICES

PRACTICE	DESCRIPTION
experimenting and iterating	developing a little bit, then trying it out, then developing some more
testing and debugging	making sure things work – and finding and solving problems when they arise
reusing and remixing	making something by building on existing projects or ideas
abstracting and modularizing	exploring connections between the whole and the parts

COMPUTATIONAL PERSPECTIVES

PERSPECTIVE	DESCRIPTION
expressing	realizing that computation is a medium of creation "I can create."
connecting	recognizing the power of creating with and for others "I can do different things when I have access to others."
questioning	feeling empowered to ask questions about the world "I can (use computation to) ask questions to make sense of (computational things in) the world."

ASSESSING DEVELOPMENT OF COMPUTATIONAL PRACTICES

The following instrument can be used to assess students' development of fluency with computational thinking practices (experimenting and iterating, testing and debugging, reusing and remixing, abstracting and modularizing). The first column indicates a question for the student (as part of a design journal prompt or interview, for example). The second, third, and fourth columns indicate how low, medium, and high levels of proficiency might be manifested.

EXPERIMENTING AND ITERATING	LOW	MEDIUM	HIGH
Describe how you built your project step by step.	Student provides a basic description of building a project, but no details about a specific project.	Student gives a general example of building a specific project in a certain order.	Student provides details about the different components of a specific project and how they were developed in a certain order.
What different things did you try out as you went along with your project?	Student does not provide specific examples of what s/he tried.	Student gives a general example of trying something in the project.	Student provides specific examples of different things s/he tries in a project.
What revisions did you make and why did you make them?	Student says s/he made no revisions, or only states s/he made revisions but gives no examples.	Student describes one specific revision s/he made to the project.	Student describes the specific things s/he added to the project and why.
Describe different ways you tried to do things in your project, or when you tried to do something new.	Student provides no examples of trying something new.	Student provides an example of trying something new in the project.	Student describes specific new things s/he tried in a project.
TESTING AND DEBUGGING	**LOW**	**MEDIUM**	**HIGH**
Describe what happened when you ran your project that was different from what you wanted.	Student does not describe what was different when s/he ran the project from what s/he wanted.	Student describes what went wrong in the project, but not what s/he wanted it to do.	Student gives a specific example of what happened and what s/he wanted to have happen when s/he ran the project.
Describe how you read through the scripts to investigate the cause of the problem.	Student does not describe a problem.	Student describes reading through the scripts but does not provide a specific example of finding a problem in the code.	Student describes reading through the scripts and provides a specific example of finding a problem in the code.
Describe how you made changes and tested to see what happened.	Student does not describe what problems s/he had or the solution.	Student provides a general example of making a change and testing it out to see if it worked.	This student provides a specific example of making a change and testing it out to see if it worked.
Describe how you considered other ways to solve a problem.	Student does not provide an example of a solution to a problem.	Student provides a general example of a solution to the problem.	This student provides a specific example of a solution to the problem.

REUSING AND REMIXING	LOW	MEDIUM	HIGH
Describe if/how you found inspiration by trying other projects and reading their scripts.	Student does not describe how s/he found ideas or inspiration from other projects.	Student provides a general description of a project that inspired him/her.	Student provides a specific example of project that inspired him/her and how.
How did you select a piece of another project, and adapt it for your project?	Student does not describe how s/he adapted scripts, ideas or resources from other projects.	Student identifies scripts, ideas or resources s/he adapted from other projects.	Student provides specific examples of scripts, ideas or resources s/he adapted from other projects and how.
How did you modify an existing project to improve it, or enhance it?	Student does not describe modifying another project.	Student provides a general description of modifications s/he made to another project.	Student provides specific examples of modifications s/he made to other projects and why.
How did you give credit to people whose work you built on or are inspired by?	Student does not give credit to others.	Student names people whose work inspired him/her.	Student documents in project and/or on the Scratch website the people whose work inspired him/her.
ABSTRACTING AND MODULARIZING	**LOW**	**MEDIUM**	**HIGH**
How did you decide what sprites are needed for your project, and where they should go?	Student provides no description of how s/he selected sprites.	Student provides a general description of deciding to choose certain sprites.	Student provides a specific description of how s/he made decisions about sprites based on goals for the project.
How did you decide what scripts are needed for your project, and what they should do?	Student provides no description of how s/he created scripts.	Student provides a general description of deciding to create certain scripts.	Student provides a specific description of how s/he made decisions about scripts based on goals for the project.
How did you organize the scripts in ways that make sense to you and others?	Student does not describe how s/he organized scripts.	Student provides a general description of how s/he organized the script.	Student provides specific examples of how s/he organized the script and why.

SUPPORTING COMPUTATIONAL PRACTICES IN THE CLASSROOM

The following instrument can be used to help you reflect on how you are supporting computational practices in your learning environment – which may be a classroom, a library, or another learning environment. The purpose of the instrument is to help you notice the types of opportunities to learn that you are designing and supporting.

EXPERIMENTING AND ITERATING: developing a little bit, then trying it out, then developing some more

The activity provided opportunities for students to...	NONE	SOME	LOTS
build a project step by step			
try things out as you go			
make revisions based on what happens			
try different ways to do things, or try new things			

NOTES FOR NEXT TIME:
If none, how can I make room, or build time, for more?
If some, how can I deepen, or strengthen, those activities?
If lots, what have I noticed, or learned?

TESTING AND DEBUGGING: making sure things work – and finding and solving problems when they arise

The activity provided opportunities for students to...	NONE	SOME	LOTS
observe what happens when you run your project			
describe what is different from what you want			
read through the scripts to investigate the cause of the problem			
make changes and test to see what happens			
consider other ways to solve the problem			

NOTES FOR NEXT TIME:
If none, how can I make room, or build time, for more?
If some, how can I deepen, or strengthen, those activities?
If lots, what have I noticed, or learned?

REUSING AND REMIXING: making something by building on existing projects or ideas

The activity provided opportunities for students to...	NONE	SOME	LOTS
find ideas and inspiration by trying other projects and reading the scripts			
select a piece of another project, and adapt it for your project			
modify an existing project to improve or enhance it			
give credit to people whose work you build on or are inspired by			

NOTES FOR NEXT TIME:
If **none**, how can I make room, or build time, for more?
If **some**, how can I deepen, or strengthen, those activities?
If **lots**, what have I noticed, or learned?

ABSTRACTING AND MODULARIZING: exploring connections between the whole and the parts

The activity provided opportunities for students to...	NONE	SOME	LOTS
decide what sprites are needed for your project, and where they should go			
decide what scripts are needed for your project, and what they should do			
organize the scripts in ways that make sense to you and others			

NOTES FOR NEXT TIME:
If **none**, how can I make room, or build time, for more?
If **some**, how can I deepen, or strengthen, those activities?
If **lots**, what have I noticed, or learned?