

Final Report Paper:  
Scratch Programming for Middle School Students

By Mark R. Feltner, M.A.

RET, Baylor University

4 August 2015

This report is a summary of the written work I have produced while working on the Scratch programming lesson plans for middle school students. The various sections of the report are as follows:

1. Observation Report Number 1: Teacher Summer Professional Development “Problem-Based Learning (PBL) in the Classroom” Observation Dates: 25 - 26 June 2015 (**page 2**)
2. Observation Report Number 2: Teacher Summer Professional Development “Project-Based Learning (PBL) in the Classroom” Observation Date: 20 July 2015 (**page 6**)
3. Thoughts on Learning to Use Scratch – Personal Log (**page 10**)
4. Computational Thinking: Practical Considerations to Implementing CT in the Public School Classroom (**page 16**)
5. Lesson Plans: Using Scratch Programming in the 8<sup>th</sup> Grade Classroom. Lesson: Learning the Bohr Model (**page 19**)
6. What Is Computational Thinking? How Can We Assess CT in the Classroom? (page 31)
7. Assessing Computational Thinking (CT) in the Classroom: Using the Scratch Program for Student CT Development (**page 35**)
8. “Teaching Students to Use Scratch” Mini Lesson Plans (**page 37**)

Observation Report Number 1:  
Teacher Summer Professional Development  
“Problem-Based Learning (PBL) in the Classroom”  
Observation Dates: 25 - 26 June 2015  
Presented by Judy York, Educational Specialist (*jyork@esc12.net*)  
ESC Rgn 12/LexLabs

Observation Notes by: Mark Feltner, M.A.

Lesson began promptly at 9:00 AM. Intro to PBL, encouragement to participate in forthcoming PBL trainings. “This will be ongoing for the next 2 years.”

Ideas from presenter: edutopia – “25 Attention-Grabbing Tips for the Classroom.”

Some time spent getting ppt’s logged into provided computers, passwords, and online pre-questionnaires completed (those emailed by Dr Li to ppt’s).

Notes taken in order that information was presented:

---

Step 1: What is your internet/technology burden? You will need to practice w/students. It takes up-front time in the first 5 days.

Task 1: Ppt’s pre-assigned to teams using color-cards. Group Task Question: “What country is most responsible for the global warming the Earth is currently experiencing?”

(my thoughts: This Q is somewhat vague, can be interpreted in various ways. *Is this intentional?* A: Yes, it is.)

During this phase of the group work, teacher/instructor provides NO assistance other than computer/tech/log-in support.

15 min. later:

3 teams: 2 appear to be working on the task; one group is having ‘teacher talk’ and socializing. Getting to know e.o.

Presenter engaged one group that was missing a key concept from original question. York: “And I want *evidence* for *what* you find.”

After 25-30 min, teams are beginning to write, compile findings on wall boards, etc. 25-30 min with grown, competent adults – how long will this take with children? (Time concerns!)

5-10 min later: York is asking each group, “Did you find any data/relationship with carbon output and consumerism?” (This seems to be a probing question that was not directly asked in the original prompting question.)

At 10:10, a few people are still completing their required online surveys (45 min later). @45 min, students are called to regroup.

Video: “Your Beacon Moment” – w/Brent James, lead singer of group Shinedown.

Following video, York asks: “When was your beacon moment when you knew our calling was to teach or stay in teaching?”

On 3x5 index card, ppt’s write when, what, who. Then: with assigned color-group (red, green, yellow, etc), 1)Find a place to talk; 2) Share your beacon moment.

My thoughts: This led to some excellent discussions of teachers’ insights into their own profession, and some of the stories were deeply moving. It was a real experience, not contrived. But, for purposes of training, I am not sure how this was directly relevant to the topic of teaching teachers how to incorporate Problem-Based Learning (PBL) into the classroom. Should teachers use this technique with students? If so, how will it advance this strategy? I am not entirely clear on this aspect.

York addresses this concern: Why a beacon for the students? It helps them break out of clique mentality. “In the first five days, you’ve got to get them used to the idea that groups are going to change.”

10 min later – time to discuss, reflect on thought process involved in approaching original question re. global warming. “It’s not so much about the content, but the process of getting to the content itself.” I agree.

Problem-Based Learning:

“Students learn about a subject through the experience of creating a problem.”

Defined: “A systematic teaching method that engages students in learning knowledge and skills through an extended inquiry process structured around complex, authentic questions and carefully designed products and tasks.”

- *Learn NC.org*

---

New section:

“How to be taught” (what we’ve been doing vs “Learn how to learn” (what we will/should be doing.)

5-Day Plan:

Day 1: Creative Problem

Day 2: Eli Pariser Effect (TED Talks: Eli Pariser)

Day 3: Challenge of the Status Quo

Day 4: Product & Productive Communication

Day 5: Go Global

My thoughts: Not clear at this point if York’s seminars over the next 2 years will cover these 5 days in 5 sessions, or what. Is today only covering day 1?

---

Types of Scientific Disclosure

- Exploring
  - Analyzing & Interpreting
  - Explaining
  - Evaluating
  - Persuading & Argumentation
-

## 5 Facilitation Principles

- Keep conversations evidence based.
- Make THINKING visible.
- Don't stop at one (one group, one conversation, one idea, everything)
- Separate ideas from individuals.
- Explore ideas with words, actions, images & symbols.

York says: "You're going to have to be a *master* of formative assessment. The students should work harder than the teacher."

Last task: "20 minutes individually researching opportunities to GO GLOBAL in your classroom."

\*\*\* end of notes from this training \*\*\*\*

## Observation Report Number 2:

Teacher Summer Professional Development

“Project-Based Learning (PBL) in the Classroom”

Observation Date: 20 July 2015

Presented by Eric Halfmann, Educational Specialist (*ehalfmann@esc12.net*)

ESC Rgn XII

Observation Notes by: Mark Feltner, M.A.

At beginning of session, ppt asked salient question regarding Dr Wang’s online survey:

(paraphrasing): “At end of survey, it asks for lesson plan. Is that a l.p. with or without PBL raining and content? L.p.’s as we have them now, or modified after this course?”

In my view, it could be helpful in survey to provide a brief description/clarification for teacher participants. This will increase response accuracy and participation rate and decrease confusion.

### Projects vs Project-Based Learning

Project-Based Learning (PBL)	Projects
<ul style="list-style-type: none"> <li>Require teacher assistance</li> </ul>	<ul style="list-style-type: none"> <li>Can be completed w/o teacher assistance</li> </ul>
<ul style="list-style-type: none"> <li>Driving Q’s</li> </ul>	<ul style="list-style-type: none"> <li>Similar end product for all</li> </ul>
<ul style="list-style-type: none"> <li>Resembles work done in real world</li> </ul>	<ul style="list-style-type: none"> <li>Not authentic</li> </ul>
<ul style="list-style-type: none"> <li>Teacher’s work is front-loaded</li> </ul>	<ul style="list-style-type: none"> <li>Little scaffolding</li> </ul>
	<ul style="list-style-type: none"> <li>Based around a tool, not a driving question</li> </ul>

Great source for comparison of Project vs Project-Based Learning:

<http://www.teachthought.com/learning/project-based-learning/difference-between-projects-and-project-based-learning/>

With PBL, teacher's work is front-loaded.

With PBL, there is a Driving Question. Students don't know answers as they progress.

With traditional Project, the project follows after teaching has occurred. Students typically know/have an idea of the answers expected.

At this point, Halfmann describes projects that have NO 'real-world' application. Insists that PBL's should mimic the types of projects seen in business/work environments, and insists that, if at all possible, students should be made to work in groups, as this is how 'real world' work is done. (Note: I take some issue with the notion that work must always be done in group settings, as most/all of the original thinking and breakthroughs created by human beings occur following periods of individual reflection and self-study.)

Halfmann: "A PBL should take 3-6 wks. PBL's should mirror things that are done in the real world. That's what PBL is all about."

Excellent graphic: Project Lesson Plan (LP) vs PBL LP (this graphic is included in Halfmann's PPT, which was provided in Schoology.com to participants.)

This comparison list comes from Teachthought.com

---

Halfmann: "PBL is supe-structured. It is based around our TEKS. It is tightly structured and it starts w/the TEKS."

"You are spending 3 weeks teaching something, so it *must* be built around the TEKS."

Comment from PPT presentation:

"It's about feedback, constantly checking in with them (the students). Your job is to make sure things are going well. Make sure they've had the chance to practice presenting to another group."

"There are times where projects fail. That is something that happens."

"With PBL, you do *not* have the option of last-minute planning. You have to prepare *weeks* ahead of time. There is a *lot* of effort on the teacher's part at the beginning."

"You want to make sure it's feasible for your students, and for the teacher. Set kids up for success."

"The due date, and work, is planned ahead of time. A firm due date is *essential*. You don't tweak it. Set it up in advance, and *stay with it*."

Very helpful: Presenter is leading participants in a conceptualization activity.

“Creating A Product”

- This is only an idea. Ideas can be changed.
  - Review TEKS/standards.
  - Match a fitting product to those standards.
  - Could this potentially be a good PBR? Is the idea: - authentic - challenging - feasible
- 

**The “Driving Question”\***

To create your driving question use the words ‘how do...?’

Example: “How do.... You create a roller coaster that won’t let the marble fall off?”

Example: “How do we help teachers better understand and relate to the students of the school?”

Students answer this question.

(\* ) This question is what the teacher has in mind as the PBL is conceived.

Keep the question simple and direct. Don’t include too many restrictions or specifications.

---

**Problem Statements**

- Help students understand their roles.
- Ensure the task is to solve a real-world problem.
- Maintain students’ interest.
- Tie into the entry event
- Provide expectation clarity.

How can we as \_\_\_\_\_

Do/create/(action verb) \_\_\_\_\_

So that \_\_\_\_\_?

Example: How can we as food chemists\_\_\_\_\_



Create/introduce healthy substitutes to common food chemistry

So that our product has fewer negative effects (or more positive effects) for the human consumer?

---

Then:

### **The Entry Event**

The purpose for an entry event (video, etc) sparks students' interest without giving them an answer to the question.

Video is the most common way.

Use Schoology for formative assessments if you have mobile devices.

\*\* A list of online tools was presented in PPT slide. Programs available for both Apple and Laptops.

\*\*\* end of notes from this training \*\*\*\*

Log:

Documenting the Learning Process of Using Scratch

By Mark Feltner, M.A.

RET, Baylor University

Summer 2015

This log summarizes my thoughts and experiences while learning to use the Scratch computer program. I am a certified public school teacher (high school science, composite, grades 8-12) with no prior experience in dealing with computer codes or programming. I have 19 years of teaching experience, including 8 years in the public sector, as well as experience teaching for private schools, private corporations, and university settings.

This report is in two sections.

- I. Thoughts on Learning to Use Scratch – Personal Log (pg 2)
- II. Computational Thinking: Practical Considerations to Implementing CT in the Public School Classroom (pg 7)

## Section I:

## Thoughts on Learning to Use Scratch – Personal Log

In viewing samples, I see that each sprite has a running program of its own. Each set of programs (scripts) appears randomly about the page. Q: Does location on screen in any way affect commands/groupings? (A: No.)

Appears overwhelming when first viewing a completed program, for example those on scratch.mit.

Notes:

Judging by programs I've seen, there is an extremely limited use of Scratch as an interactive game that will educate the end user (that is, user gains comparatively little new knowledge as compared to what the creator had to master to write the program.)

Therefore: User projects/assignments should give students:

1. A simple task;
2. Students design/fulfill the task with Scratch;
3. Alter the task;
4. Create new permutations of original task.

Q: Should steps 5, 6, etc, include making project into an *interactive* activity for viewer?

Ultimate objective for educator:

Get students to use iterative thinking while engaging with chemistry concepts.

- Primary focus: Iterative thinking practice for students;
- Secondary focus: Chemistry concepts applied.

Objectives, as seen from students' perspective:

- Primary: Create product desired (as directed by teacher)
- Secondary: Conceptualize chemistry lessons to complete obj 1.

Worth noting: Students need not understand iterative thinking to engage in the practice. Indeed, it may be more productive if they are somewhat unaware of it. When given a task, particularly in the form of a challenge or a game, students are often so caught up in the activity itself that they forget they are supposed to be 'working' because they are having fun.

Interesting situation: Teacher is using linear technique to start students on a non-linear/iterative process.

Feedback loops: pos = creates more of prior behavior; neg = less of prior behavior

If done right, this should be a largely self-correcting process w/less teacher involvement than standard lessons, labs.

---

Programming Idea:

Create Bohr-type atomic model using 5 elements (given in original program) as a model. Periodic table is provided. Students add 1, 2, 3, etc, new models (elements of their choice – some will stay small; others will want big, challenging atoms).

Q: Work in pairs? 3 is too many, in my view.

\*\* Also: This is *not* primarily about teaching atom construction. Several top-rate programs already exist that allow students to do just this thing online (that is, build/construct atoms). **The difference here is that the students are writing a program rather than being the end-user.**

### Scratch Programs

The final products are primitive, comparatively simple.

**If the ultimate purpose of writing a program is for the entertainment of others, then Scratch is not a good choice.**

**But if the ultimate purpose is to teach iterative thinking skills, then the Scratch program is an excellent choice.**

The question, then, is ‘Where does the focus lie: On the final product, or on the process itself?’

(Goal-driven vs. Process-driven, in terms of systems thinking.)

CT = Computational *Thinking*. Emphasis is on the style of thinking. Otherwise, programming would be minimized for programmer, and end result would be maximized for entertainment value (Minecraft?). (High enjoyment/slick end product coupled with low effort/low design demands.)

### Iterative Lesson Plans

Within an iterative framework, students (just like adults/teachers/researchers) will need time to reflect, correct, adjust. An important aspect is time to think.

How can we facilitate this, provide for this?

- Activities in class are broken into 15-20 minute pieces.
- Students have time between assignments, between classes, to think/reflect.
- Critical component: During classes, students have time to experiment, fail, learn, improve, try again.
- They have the time and opportunity to work outside of class on project, if desired, independently of teacher input and oversight, optionally, at students' discretion, as desired.
- Open source scripts: May make so-called 'cheating' easier, but allows or chance to learn by way of modelling. The true innovation comes when students go past, rather, than the scripts they have viewed, copied, imported. (But isn't viewing finished scripts the way I am partially learning this program right now? Of course it is.)

Idea for classroom assignments. Student learn by model: Perhaps require they expand/extend it into their own, original script. Maybe make this a requirement? ("One original twist/extension to original program"?)

### Thoughts on Educating the Educators

Teachers are going to need a course, a guided course, in the use and application of Scratch. Given the demands on teachers' time and schedules, it seems highly unlikely that most STEM teachers will be able to teach themselves this program to any effective level without a dedicated opportunity to do so.

In my view, teachers will need:

1. Training in Computational Thinking, its definition, merits, and practice;
2. Training in Scratch programming;
3. Manual (book, for example);
4. Sample lessons relevant to course, content.

Important (critical?) to emphasize the difference in the CT process (iterative) vs. final product (linear).

My perspective right now: Still unsure how effective it will be using class time or iterative strategies. When class teachers realize that each student is learning different things content-wise, and not all are covering the same required content, teachers may drop this style of learning altogether.

## Difficulties

After working with Scratch program for a while:

Frustrated. Following book's instructions on changing backdrop, and can't get sprite icons back again. Ok – found solution. Couldn't readily find buttons to change screen. Screen is very busy, with a lot of symbols, tools, visual clutter.

At one point, 'Saving' after backdrop change seemed to have gotten stuck. Had to exit out, reload program. Some changes were lost.

Altering Scripts:

Altering scripts systematically. What worked for firs then sprites did not work for the 11<sup>th</sup>. Don't know why. Reverted to earlier saved version of the program and it worked. Why? What was different? Could see no appreciable difference. Was a frustrating experience to have to stop, exit out, come back in and see things working perfectly.

Scripts:

by changing zoom, you lose the ability to delete scripts (the right-click function of mouse appears disabled). Why? Is this just my computer?

NOTE:

Graphics in Scratch are NOT easily manipulated. At least, not by myself. As a beginning programmer/program language user, many things are not intuitive. I would need to develop a solid skill-set before making major changes to graphics with Scratch.

What I need at this point is a digital way to easily build a Bohr model of the atom with certain key information recorded around the image. How easiest to do this? PPT? similar program? I do not have time to master Scratch graphics – best to go with what (little) I currently know.

---

As I work to educate myself on this program, I am wondering: What is the fastest way to learn a new programming language for someone who's never programmed before? In this sense, I am the target audience for such a thing, as this is my first encounter with programming of any kind.

Possible means to educate the novice:

1. A textbook, manual, as I am doing (this is slow: tedious and frustrating (to me)).
2. Trial-and-error, by reading other code sources (not very helpful for me at beginning stages, though will be helpful later).
3. Teacher-student method with tutor/teacher? (Using structured practice, a linear model, to teach the tools necessary to think in iterative fashion.)

\*\* Will we need much training time up front to educate students in the use of Scratch before we launch into first assignments?

Later: More work with reading, practicing with programs. This is a bit overwhelming – screens w/names that have menus inside of pull-out side/help menus. I find myself getting lost down the rabbit hole, so to speak.

## Section II:

### Computational Thinking: Practical Considerations to Implementing CT in the Public School Classroom

This section describes potential areas of concern that should be taken into consideration when designing a Computational Thinking (CT) component for use in public school classrooms. For a new classroom program to be effective, it must be practical, efficient, and deemed effective by those who will use it. This is especially true when the program is different in form than the prevailing model of programs currently in use; specifically, the current classroom model of linear instruction techniques (lecture, practice, lab/hands-on, examination) versus CT, which is iterative in its approach to learning (imagine, create, play, share, reflect, imagine, and so forth (Resnick, 2007)).

A linear model of teaching typically includes a body of information, instruction and modeling that flows from the authority (teacher) to the novices (students) with the intent to ensure that the students master as much of the content in as little time as possible. Efficiency is the goal. An iterative model, in contrast, relies on the creativity and participation of the students, rather than the structured guidance of the teacher, in an effort to allow students to learn from their own questions, make mistakes, and make improvements as they learn not only what to think, but how to think about the topics at hand. In an iterative model, introspection, experimentation and self-directed learning is the goal, which, in contrast to the linear model, is not necessarily a very efficient process. An iterative model is viewed as a superior method by which to train future scientists and mathematicians, whereas the linear model is viewed by many as a superior model for ensuring that all students achieve a minimum level of fact-based knowledge in order to complete a course and advance to a higher grade level. Clearly these two approaches differ not only in utilization of time and resources, but in the thought processes of instructor/teacher and students alike.

Below are some areas for consideration when developing iterative models of instruction for use in traditionally linear settings. This paper assumes the necessary material and instructional requirements have been met (classroom, teacher, computers, etc). No solutions are presented herein; only potential challenges to CT implementation are discussed below.

Please note that the following areas are not discrete categories that can be approached independently of one another. Rather, they are all interconnected and overlapping. An increase in *complexity*, for example, will likely result in an increase in *time* needed to complete an activity, while possibly decreasing the instructor's view of the *apparent utility* of the activity. Thus, a change in one aspect will almost necessarily require changes in all others.



Five areas of concern:

1. Time.
  2. Complexity.
  3. Linear versus Iterative Strategies.
  4. Content and Assessment.
  5. Apparent Utility, Effectiveness of New Techniques.
- 

1. **Time.** Public school teachers are usually provided with a list of content that is expected to be covered in the course of the school year. In recent years, the list of ‘required’ concepts continues to grow, while classroom resources often remain constant or even decrease. As population grows, so too does the average class size; this serves to decrease teacher effectiveness. Very many teachers, then, are presented with a multi-faceted challenge when designing lesson plans. A primary question teachers must face on a daily basis is, ‘How to present the information to the students in a way that will convey as much information as possible, to the greatest number of students possible, in the least amount of time possible?’

Thus, the introduction of a new program into the classroom must not only be superior in its effectiveness to traditional methods of education, but must be effective in its use of time as well. Time is at a premium in the public school classroom, and a competent, motivated teacher will first and foremost consider the ‘cost versus benefit’ of a program from a time-based perspective: ‘Is this the best use of my class time?’

2. **Complexity.** There are two areas of ‘complexity’ to be considered: the level of complexity as pertains to the instructor, and level of complexity as pertains to the students. If either the students or the teacher believe the content to be beyond their capabilities, the work will go unfinished and the project will likely be abandoned. Further, any future attempts to follow an iterative approach will likely be met with increased resistance, particularly from teachers and administrators who viewed prior attempts as failures and a waste of time and resources. It is imperative, then, to develop a program that is fairly easily grasped by teachers, as they typically have numerous other demands already placed on their time, attention, and energy.
3. **Linear vs Iterative Strategies.** As described above, teaching methods in the classroom today are focused on maximum efficiency in limited time. It is an industrial model of ‘maximum units of output for energy put into the system.’ Units of output in this case are high school graduates, and graduation is largely dependent on assessment tools, namely tests, at both the local school and state levels. Therefore, possible assessment strategies for an iterative approach must be considered by the iterative program’s designers.

Simply put, it is entirely possible, even likely, that the knowledge and skills developed by students working within an iterative framework will not be readily demonstrable in standard classroom examinations that were designed for assessing students educated in a linear environment. Standardized examinations are the primary method by which the State determines funding and resources for schools and school districts. It is unlikely that district administrators will be willing to risk lowered test scores for the benefit of introducing any new method of student learning, regardless of how beneficial it is to students' intellectual development. Testing is money, and money keeps the schools open. It is therefore critical to ensure that students' gains in knowledge are readily demonstrable to teachers, administrators, and State officials alike.

This topic ties in very closely to point number 4, **Content and Assessment**, below.

4. **Content and Assessment.** There is, of course, a need to prove that students are learning relevant material in a demonstrative way. Considering the industrial model of education (assembly line, units of output) vs a more holistic approach (the process behind the thought rather than specific bits of required information), one can probably see the underlying difficulty with today's educational environment as it grapples with introducing the so-called 'Project-Based Learning' technique that is now coming into vogue.

The effective delivery of course content is the primary concern for education today. Classrooms are structured for maximum economical utility (i.e., 30 students per teacher, rather than 10:1, or 5:1). Rooms are designed primarily for lecture style dissemination of information. Administrators are trained to look for 'data-driven' results, not less tangible measures such as student's holistic knowledge and problem-solving abilities, despite the fact that this is how science progresses in the 'real world' and how societies have advanced in history.

In fluid, dynamic situations, people must be able to think and not just regurgitate facts; despite this, the State of Texas has a complex and highly specific system of ranking schools based on measurable data of discrete categories that typically involves student responses to forced-choice (e.g., multiple choice) answer formats. This type of evaluation does not lend itself to allowing time for students to practice, err, practice again, err again, and eventually succeed at independent tasks. Core teachers (those whose performance is assessed annually by the State by ranking students' performance on multiple choice tests) are acutely aware of every minute lost in today's classroom settings. Iterative strategies, requiring more time by their very definition, will thus be at odds with the prevailing paradigm of today's public school classroom.

In the article by Taylor, Harlow & Forret (2010), they report on a field study of middle schoolers using the Scratch program in a math class:

“ Scratch does not facilitate learning in any one particular mathematical area, but it can enable each child to process their mathematical activity through a digital medium so the

understanding that emerges is shaped in alternative ways. *What is learned, therefore, will be different for each child.*” (emphasis added for effect by Feltner.)

As beneficial as this sounds, the government still requires that all children, by way of standardized exams, be able to demonstrate the *same* knowledge by giving the *same* answers to the *same* questions on the same exam.

So, how can iterative strategies be successfully introduced into a linear model of education? How can CT program designers ensure that skills and content being taught under an iterative model will not only develop students’ long-term thinking, initiative, and problem-solving skills, but also deliver the immediate-term requirements of specific content in each course? Is it possible to ensure that students can demonstrate particular skill sets on state exams if less time is available to practice the content in the linear/exam-type format?

5. **Apparent Utility, Effectiveness of New Techniques.** The human element cannot be overlooked here. Most new programs are initially met with a fair amount of skepticism, if not outright doubt. New ideas and techniques and suggestions and technologies are often introduced to teachers with such frequency that it may be initially difficult to convince teachers (and indeed administrators) of the value of yet another ‘best new thing.’ The validity and relevance of a CT program must be clearly and solidly presented from the outset to help avoid some of the negative scenarios described above.

If any of the above points are viewed as system failures, then those responsible for implementation of the new iterative strategies (teachers, administrators, students themselves) will likely be less willing or unwilling to engage in similar activities in the future. All people involved in the execution of such a new program must be knowledgeable and aware of the benefits of an iterative approach. The importance of this cannot be overstated.

## Lesson Plans: Using Scratch Programming in the 8<sup>th</sup> Grade Classroom

### Lesson: Learning the Bohr Model

The following lesson plans are intended to be used in conjunction with teaching middle school students the basics of the atom. These 20-minute lessons should be given across three successive days, rather than being combined into one or two longer lessons of 40 or 60 minutes each. It is also assumed that the students have had some former exposure to the Scratch programming language. Students are encouraged to work on their projects outside of class.

#### Contents

Lesson 1: Introduction to Scratch Program “The Bohr Model of the Atom”

Lesson 2: Developing Your Bohr Model Atom

Lesson 3: Finishing Your Bohr Model Atom

For these lessons, the teacher will need to access the Scratch program, ‘The Bohr Model of the Atom.’ This program shows a periodic table in the lower half of the screen with a large illustration of a Bohr model atom in the top half. In the original program, only the first five elements are completed; the students’ task is to complete a similar model of a particular atom.

Students should work in groups of three. This will work best if you have a pre-assigned list ready before class begins. You may assign each group of students to an atom, or you may let students choose their own, as you see fit. This may be an excellent chance, however, to keep a class focused on one section of the periodic table, such as the non-metals, the metalloids, et cetera.

The outline of each day’s lesson is listed below. Each day’s lesson is followed by a Closing Activity, which is meant to serve both as a summary and a quick assessment of students’ understanding of the day’s concepts. Please note that although students are working in teams, the Closing Activity is an individual activity.

Graphics credits: [http://duch.sd57.bc.ca/~rmcleod/Chemists\\_Corner/Bohr.html](http://duch.sd57.bc.ca/~rmcleod/Chemists_Corner/Bohr.html)

## Lesson 1: Introduction to Scratch Program “The Bohr Model of the Atom”

### 20 minutes

1. Teacher accesses Scratch program, ‘The Bohr Model of the Atom.’ Teacher projects this screen where students can see it.
2. Lead students to Scratch web site: [mit.scratch.edu](http://mit.scratch.edu)
3. Students create accounts, one per team. Each team member writes down the name and password, as well as the MIT web link to the Scratch site.
4. Have student teams fill out Document 1: ‘Student Teams Log’. Keep log in a secure place as a record of student log-in codes. (Some students will forget their passwords and will need your help!)
5. Students log in to Scratch web site, access program ‘The Bohr Model of the Atom.’
6. Give them time to look around. Show them the ‘look inside’ button on their screen and encourage them to tinker with the commands and see how the screen changes as they alter commands.
7. Save Work. Have students save their work before exiting. Remind them that the original ‘Bohr Model’ program is available for them to view at any time, and that they can access this program outside of class as well, if they wish.

End of activity.

After the computer work is completed, administer ‘Closing Activity for Lesson 1.’

Closing Activity for Lesson 1

In the space below, draw a Bohr model illustration of hydrogen (H on the periodic table).

<p><b>Element Name:</b></p> <p><b>Atomic Number:</b></p> <p><b>Atomic Mass:</b></p> <p><b>Protons:</b></p> <p><b>Neutrons:</b></p> <p><b>Electrons:</b></p>
---

Questions about the Bohr Model

1. The helium atom (He on the periodic table) has two electrons. How many protons does helium (He) have?

---

2. The element of oxygen (O) has an atomic number of eight. How many protons does it have? How do you know?

---

---

---

KEYClosing Activity for Lesson 1

**Element Name:** **Hydrogen**

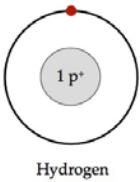
**Atomic Number:** **1**

**Atomic Mass:** **1**

**Protons:** **1**

**Neutrons:** **0**

**Electrons:** **1**



Hydrogen

In the space below, draw a Bohr model illustration of hydrogen (H on the periodic table).

### Questions about the Bohr Model

1. The helium atom (He on the periodic table) has two electrons. How many protons does helium (He) have?

**Helium has two protons. In normal atoms, number of protons and electrons are the same.**

---

2. The element of oxygen (O) has an atomic number of eight. How many protons does it have? How do you know?

**Oxygen has eight protons. The atomic number of an element is determined by its number**

---

of protons; proton count and atomic number are the same.

---

---



## Lesson 2: Develop Your Bohr Model Atom

### 20 minutes

1. Following any introduction the teacher deems necessary, students, in their groups of three, begin online research into their assigned or chosen atom.
2. Computational Thinking: Students are allowed the time and encouraged to write, copy, or edit code, make corrections, improve, and continue writing. This is the process of learning students need to develop for problem-solving mindsets.
3. Save Work. Have students save their work before exiting. Remind them that the original 'Bohr Model' program is available for them to view at any time, and that they can access this program outside of class as well, if they wish.

End of activity.

After the computer work is completed, administer 'Closing Activity for Lesson2.'

Closing Activity for Lesson 2

In the space below, draw a Bohr model illustration of Carbon (C on the periodic table).

<b>Element Name:</b>
<b>Atomic Number:</b>
<b>Atomic Mass:</b>
<b>Protons:</b>
<b>Neutrons:</b>
<b>Electrons:</b>

Questions about the Bohr Model

1. The boron atom (B on the periodic table) has an atomic mass of eleven (11), but the atomic number for boron is only five. What accounts for the difference between atomic number and atomic mass?

---

2. An unidentified element has 10 electrons. What is the likely name of this element? How do you know?

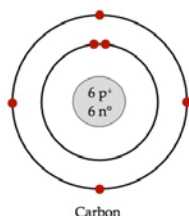
---

---



KEYClosing Activity for Lesson 2

In the space below, draw a Bohr model illustration of carbon (C on the periodic table).



**Element Name: Carbon**

**Atomic Number: 6**

**Atomic Mass: 12**

**Protons: 6**

**Neutrons: 6**

**Electrons: 6**

Questions about the Bohr Model

1. The boron atom (B on the periodic table) has an atomic mass of eleven (11), but the atomic number for boron is only five. What accounts for the difference between atomic number and atomic mass?

The difference in atomic number and atomic mass is accounted for by an atom's number of neutrons.

Atomic number = number of protons;

Atomic mass = Number of protons + number of neutrons

2. An unidentified element has 10 electrons. What is the likely name of this element? How do you know?

This element is probably neon (Ne). You can tell because neon's atomic number is 10, which is also its number of protons. Electrons (negatively-charged particles) and protons (positively-charged particles) are always balanced in non-ionic atoms.

### Lesson 3: Finish Your Bohr Model Atom

#### 20 minutes

1. Following any introduction the teacher deems necessary, students, in their groups of three, begin online research into their assigned or chosen atom.
2. Computational Thinking: Students are allowed the time and encouraged to write, copy, or edit code, make corrections, improve, and continue writing. This is the process of learning students need to develop for problem-solving mindsets.
3. Finish Work. Have students save their work before exiting. Remind them that the original 'Bohr Model' program is available for them to view at any time, and that they can access this program outside of class as well, if they wish.

End of activity.

After the computer work is completed, administer 'Closing Activity for Lesson 3.'

Closing Activity for Lesson 3

In the space below, draw a Bohr model illustration of Freon (F on the periodic table).

<b>Element Name:</b>
<b>Atomic Number:</b>
<b>Atomic Mass:</b>
<b>Protons:</b>
<b>Neutrons:</b>
<b>Electrons:</b>

Questions about the Bohr Model

1. Sodium (Na) and Magnesium (Mg) both have 12 neutrons, yet they have different atomic masses. How is this possible?

---

2. An unidentified atom has 10 neutrons and 9 protons. What is the atomic mass of this atom, and what is its name?

---

---

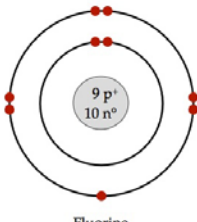
3. How many electrons does Neon (Ne) have? How do you know?

---

---

KEYClosing Activity for Lesson 3

In the space below, draw a Bohr model illustration of Fluorine (F on the periodic table).



Fluorine

**Element Name: Fluorine**

**Atomic Number: 9**

**Atomic Mass: 19**

**Protons: 9**

**Neutrons: 10**

**Electrons: 9**

Questions about the Bohr Model

1. Sodium (Na) and Magnesium (Mg) both have 12 neutrons, yet they have different atomic masses. How is this possible?

It is possible because they have different protons (Na has 11; Mg has 12). This is what makes them different elements.

2. An unidentified atom has 10 neutrons and 9 protons. What is the atomic mass of this atom, and what is its name?

The atomic mass is 19 ( $10n + 9p = 19$  atomic mass), and its name is Potassium (K).

3. How many electrons does Neon (Ne) have? How do you know?

Neon has 10 electrons. The number of electrons in a balanced (non-ionic) atom is always equal to its number of protons, and neon has 10 protons. Neon is element number 10 on the periodic table, and the element number is based on the number of protons that a particular element has.



## What is Computational Thinking (CT)?

Essentially, Computational Thinking (CT) can be thought of as problem-solving with, and using, computers. CT skills are developed when learning to program and debug computers, though the kinds of iterative problem solving skills commonly developed during computer programming – try, fail, reflect, adjust, try again - is important in all areas of life (Marji, 2014).

With the proliferation of technology and tools that allows students to design their own media, there is a growing interest in encouraging the type of critical skills among young people that will allow them to view themselves not only as end-user/consumers of digital technology, but as active creators and programmers as well (Brennan & Resnick, 2012). Now, more than ever before, it is critical students to develop these skills as part of their ongoing education.

## Assessing CT in A Classroom Setting.

Computational Thinking is more of a process of learning to problem-solve via test-retest-improve methods than it is an act of memorization and utilizing strictly fact-based knowledge. Computational Thinking skills grow over time and apply to many different aspects of student' learning and thinking; thus, measuring the development of CT skills in students will not be easy (Brennan & Resnick, 2012).

While measuring the growth of CT skills in students can prove challenging, it is nonetheless possible to collect evidence that these skills are beginning to take shape in students' thought processes. Identifying evidence of computational thinking skills is something that can be achieved by classroom teachers. Following are seven (7) key concepts students may demonstrate during the development of their Scratch programs (as described in Brennan & Resnick, 2012).

These same concepts are used on the teacher checklist, provided at the end of this section (“Assessing Computational Thinking (CT) in the Classroom: Using the Scratch Program for Student CT Development”).

### Seven (7) Key Concepts of Computational Thinking

1. **Sequences.** A sequence of programming instructions dictates the actions or results that should be produced. For instance, an object on the screen can be programmed to move a certain distance and then make a noise.
2. **Loops.** Instead of a single repetition of an action (as described above), a loop provides the mechanism for running the same sequence multiple times.
3. **Parallelism.** This is a sequence of two or more instructions that are happening simultaneously. This could include multiple items moving and interaction onscreen at the same time, or an image onscreen performing two or more activities in parallel as a response to the same stimulus (e.g., when an icon is clicked).

4. **Events.** When one event in the program occurs and causes another event to happen as a result. When an object onscreen is clicked with the mouse, for example, the object may rotate or move, etc.
5. **Conditionals.** In Scratch programming, this typically includes use of the ‘if’ block in the scripts. Thus, a certain result or will occur only under certain conditions.
6. **Operators.** These allow programmers to use numeric calculations in their programs. Scratch allows for not only addition, subtraction, multiplication, division, but string operations as well.
7. **Data.** This includes the storage and retrieval of data. Scratch uses variables and lists to maintain data. In a program, this would include things like keeping score in a game, or the calculation of other numerical values.

### Suggestions for Assessing Computational Thinking in a Classroom Setting

In addition to a formative assessment-based checklist (described above), teachers may find the following techniques helpful in evaluating students’ CT skills. These suggestions may be utilized in different ways, though classroom teachers may find it helpful to use an interview or student-presentation format as a chance to speak and interact with students regarding their products and thought processes.

**I. Incorporating artifacts.** Use students’ Scratch programming as part of your evaluation of their development. Assessments, particularly formative assessments, should include the examination of individual projects. Students’ projects are rich, contextualized examples that can be analyzed in a variety of ways by both teachers and students themselves. (Brennan & Resnick, 2012)

**II. Illuminating processes.** Analyzing students’ actual coding scripts is one way to begin assessing their learning, but having conversations with students about their work can reveal rich and insightful thought processes that are at work behind the actual programming. As Brennan & Resnick (2012) ask, “What understanding does the designer have about particular concepts? What practices did they employ?” Asking students about past and future intentions may increase a teacher’s understanding of the students’ growth in CT skills.

**III. Checking in at Multiple Waypoints.** Using a formative approach to assessment, checking students’ progress as their current projects develop is highly useful. Programs take shape over time and checking in with students periodically will help reveal CT processes at play in the students’ learning.

**IV. Valuing Multiple Ways of Knowing.** Computational Thinking is a process that grows and evolves over time, often outside of a student's own awareness. Thus, checking whether students understand definitions of key programming concepts is not enough to measure CT. Rather, a better question may be, 'can the student learner put these concepts into a meaningful design? Can they analyze and critique others' code, as well as their own?' As students learn to integrate new and different skills, their problem solving abilities will grow as well.

## References

Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada.

Cuny, J., Snyder, L., & Wing, J.M. (2010). Demystifying computational thinking for noncomputer scientists. Unpublished manuscript in progress, referenced in <http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf>

Marji, M. (2014). *Learn to program with scratch*. William Pollock, Publ., San Francisco, CA, USA.

Assessing Computational Thinking (CT) in the Classroom:  
Using the Scratch Program for Student CT Development

The following tool can be used as part of an ongoing formative assessment in the classroom. This sheet is meant to aid teachers in recording evidence of students using and/or understanding seven key concepts identified as part of Computational Thinking.

Teachers are encouraged to make notes in the fields below, or simply record Yes/No responses to whether students or teams are demonstrating certain skills on a particular day. The chart below assumes an ongoing project with three observation opportunities for the teacher (each observation on a different day).

**Formative Assessment: Teacher Observation across Three (3) In-Class Sessions**

Does the student/team show knowledge in the application of:

<b>Computational Concept</b>	<b>Day 1</b>	<b>Day 2</b>	<b>Day 3</b>
<b>Sequences.</b> Programming instructions that dictate actions or results that should be produced.			
<b>Loops.</b> Provides the mechanism for running the same sequence multiple times.			
<b>Parallelism.</b> Two or more instructions that are happening simultaneously.			
<b>Events.</b> One event in the program occurs and causes another event to happen as a result.			
<b>Conditionals.</b> A certain result or will occur only under certain conditions. In Scratch, this typically includes 'if' block in the scripts.			
<b>Operators.</b> Allow programmers to use numeric calculations in their programs.			
<b>Data.</b> Storage and retrieval of data (keeping score in a game; calculation of numerical			

values).			
Notes.			

**References**

Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada.*

## “Teaching Students to Use Scratch”

### Mini Lesson Plans

Mark R. Feltner, M.A.

July 2015

**Purpose:** This paper outlines a three-day introduction to computer programming using the Scratch programming language. The intended audience are middle- to high-school aged students. These lessons derive from the book, Learn to Program with Scratch, by M. Marji (2014). This text, while required for using the mini-lessons outlined below, is also highly recommended for those who wish to learn and teach programming using the Scratch system.

**Description:** The following lesson plans are intended to be taught over three separate days, for approximately 45 minutes each. Forty five minutes allows for a lesson to fit into a traditional 50-60 minute class period, as well as serving as part of a lengthier 90-105 minute period (block scheduling). It is important that these three introductory lessons be taught across three separate class periods and not combined into one or two lengthier sessions. Students will need time to learn and practice the skills outlined below.

**Materials:** The instructor will need a copy of Learn to Program with Scratch. Students will need access to the internet. Though Scratch can be accessed by hand-held devices, full-size computers (desktops, laptops) are strongly recommended. The Scratch program is open-source and available at <https://scratch.mit.edu>. Model programs are also available there for viewing, use, and alteration.

**Notes on Usage:** The following mini-lessons are suggestions of how to get the students familiarized with some of the most basic yet critical aspects of the Scratch program. The timeframes presented below are guidelines. Students will encounter difficulties and have questions – please allow them to explore on their own as they learn, as it is key to their learning process.

#### Contents

Day 1: Getting Started (45 min)

Day 2: Motion and Drawing (45 min)

Day 3: Looks and Sounds (45 min)

## Day 1: Getting Started

45 minutes

1. (10 min) Students create their own account at <https://scratch.mit.edu>. Make sure they have their usernames and passwords recorded somewhere, for example in a log that the teacher keeps at the main desk.
2. (5 min) What Is Scratch? Teacher projects image of book, pp 2-3, and leads students in brief discussion of '**Try It Out 1-1**' (p 3).
3. (5 min) The Stage. Describe the three main parts of the Scratch screen: Stage (top left) the Sprite List (bottom left), and Scripts tab (right side). Have students explore x-y coordinates of Stage, as described in '**Try It Out 1-2**' (p 4).
4. (10 min) Sprite List. Lead students through '**Try It Out 1-3**' and allow them 5 min to become familiar with the Sprite list, thumbnails, and available sounds.
5. (10 min) Blocks Tab. This is the students' first encounter with the different types of commands that can be written. Give students 5 minutes or so to become familiar with the ten different command categories (motion, looks, sound, pen, data, events, control, sensing, operators, more blocks). Students at this point only must know where the blocks are located and have an idea of the types of overall categories available. Lead students through activity '**Try It Out 1-4**'.
6. (5 min) Saving Work. Students will name their program and save it. Shut down.

**\*\*End of Mini-Lesson, Day 1\*\***



Day 2: Motion and Drawing

45 minutes

1. (5 min) Log-in and re-familiarization of Scratch program screen. Provide assistance to students with log-in difficulties.
2. (15 min) Scripts and Scripts Area. Students will write a simple script to move their Sprite on screen. Lead them through '**Try It Out 1-5**'. Make sure students get practice at assembling and disassembling the blocks on the right side of the screen. Point out that they can see immediate effects of their scripts on the Stage.
3. (10 min) Adding another Sprite and copying scripts. Lead students through '**Try It Out 1-6.**'
4. (10 min) Costumes Tab. Here, students learn how to change a sprite's appearance. If possible, demonstrate page 9's illustrations (projector, etc), then lead students through '**Try It Out 1-7**'.
5. (5 min) Saving Work. Save work and shut down.

\*\*End of Mini-Lesson, Day 2\*\*

### Day 3: Looks and Sound

45 Minutes

1. (5 min) Log-in and re-familiarization of Scratch program screen. Provide assistance to students with log-in difficulties.
2. (15 min) Sounds Tab. The students will probably have fun with this one! The sprites can also play sounds, either pre-recorded from the Scratch sound bank, or recorded by the students themselves. Guide students through Fig 1-11 (pg 10), and then the practice exercise '**Try It Out 1-8**' (pg 10).
3. (15 min) Backdrops Tab. The name of the Costumes tab will switch to Backdrops when you select the thumbnail of the Stage in the Sprite List (Marji, pg 11). Use this tab to alter the Stage's background images. Lead students through '**Try It Out 1-9**' (pg 11).
4. (5 min) Toolbar. Lead students through the various components of the Toolbar, as illustrated in *Fig 1-14: Scratch's toolbar* (pg 11). This can be brief, as this is for familiarization and not expected to be fully memorized at this point.
5. (5 min) Saving Work. Save work and shut down.

**\*\*End of Mini-Lesson, Day 3\*\***