Project ECSITE Unit: Programming as Story Telling

This unit summarizes sequential activities that spanned one high school semester and include specific and general activities comparing the process of designing and writing computer programs with writing stories.

Content Areas: computer science, geometry, numbers & operations, physical science, physics, problem solving, reasoning & proof, science & technology

Grade Level: 6-8, and 9-12.
This unit was developed with high school (grades 9-12) students. It was agreed between the teacher and the Fellow that these lessons might be best used in middle school classes, allowing lessons to be reinforced and built upon in high school.

Computational Thinking Connection:
In this unit, computational thinking is approached as literacy. The teacher and the Fellow mapped the process of story telling to the process of creating original computer programs. The concept builds on the fact that students are already familiar with language arts and the process of designing and writing computer code is approached in an analogous way, professionally. Meaning, programmers no less pre-plan, sketch, and reconfigure code than authors iteratively refine their writing, it’s language, structure, order etc.

Pre-Requisite Knowledge:
The tools for this unit are often free, open-source coding and physical computing platforms such as Scratch and Arduino, respectively. The teacher and Fellow were able to learn or refresh knowledge in these programming environments in a planning period no more than one week in advance. Of note, the teacher in this unit was mainly at a beginner level. The Fellow used the same “story telling” devices to instruct the teacher in new programming content. As such, the analogies were easily translated into classroom activities for students. This suggests, perhaps, that teacher instruction happens best with an assistant like, in this case, the Fellow.

Time Required: 210 minutes per week contact time. 30 minutes preparation time with teacher and Fellow. 90 minutes (x2) class time with teacher and Fellow alternating between separate activities, full group activities and in-class work sessions.

Related Lessons/Activities:
• Code of Me: Unplugged activity where students list and organize the components of a brief autobiographical story in the form of a simple computer program. (Worksheet attached).
• Scratch programming. After learning the basics, students take the “Code of Me” concept and write a short story (fiction or non-fiction) in which they identify variables/characters, set up/setting, loop/ story over time and conditionals / changes in the story ahead of time in writing. Students then work to program the story they’ve prescribed.
• Grid Game: Unplugged activity poses teams of two students as “programmer” and “pixel”. Using simple rules common to computer programs students must pre-list “algorithms” that are used to sequentially guide a blindfolded “pixel” through a maze of obstacles and other “pixels”. This introduces concepts of algorithms, logic, limits and “bugs” in computer programs. (Worksheet attached)

• Arduino and “Physical Computing”: Using the same design method of writing, translating to code, and pre-designing the outcome or changes that might occur conditionally in a story, students began exploring software hardware interfaces. For example, the “characters” in their story might include “LED5, LED6 and LED7” on the Arduino microcontroller.

• Origami Lamp: Students were challenged to pre-design an interactive origami lamp using the Arduino microcontroller and art materials of their choice. Again, they described the interaction in a story, translated it into code, and executed their projects according to their own script. Students were evaluated based on their ability to match their program to their plan.

• PONG, one player to two player: Students, now familiar with the Arduino programming language and environment were challenged to reverse engineer a version of Atari Pong written in the Processing programming environment. Using the skills of writing and translating to code, students wrote in natural language their reading of the code. Once this was complete, they were challenged to re-write a one-player Pong game for two players.

Attachments:
• Code of Me
• Grid Game

Contributors:
Principal Investigator: Debra S Goldberg, Colorado University Boulder
NSF GK-12 Fellow: Zack Jacobson-Weaver, Colorado University Boulder
Mentoring Teacher: Axel Reitzig, Saint Vrain Valley Public Schools

Supporting Programs:
National Science Foundation
eCSite Program for Computational Thinking, Colorado University Boulder
The Code of Me

What things do you need to have a good story?

-Timeline
  -Characters
  -Settings

  -Series of Events in Important Order...
    -Chapters...
    -Actions of the Characters

You can build a story of your life, putting those things in order, by writing sentences. But, you can also build it into a code. Obviously, if we covered everything about you, we would have to write a novel! We will describe FOUR things about your life to keep it simple:

Me  Home  School  and  Hobbies (but just one).

Let’s get started!

TIMELINE: the timeline in a story is usually in the background. Sometimes we focus our attention on it because its useful to know how much time has passed. Computer programs are the same. Time is always passing, and we can ‘check the time’ if its useful to understand the whole story of the program, or just where we are in the story.

CHARACTERS: What are the characters of ME, HOME, SCHOOL and HOBBY? These are the people, places, things and events that make them different from the others.

Me is easy. That’s you. You are the same no matter where you are. We could say you are you anywhere on the globe. That makes ME a global variable.

ME = your name.

ME = _______________________________

**Note: Global variables get defined in the SETTING of the story...keep reading.**
Home, like School and Hobbies, is not the same for everyone, everywhere. The people, places, things and events that make it home have different local characters, so we call them local variables. List four of those things for each.

HOME=____________________,____________________,____________________,____________________
SCHOOL=____________________,____________________,____________________,____________________
HOBBY=____________________,____________________,____________________,____________________

SETTING: These are the things found throughout the story. Like ME (you), they are global variables. They are the same no matter where you find them. They are specifically defined in the setting. List one thing besides you that is part of your HOME, SCHOOL, and HOBBY.

__________ = _______________________________

A SERIES OF EVENTS in important order make the body of a story. We have to say the order of the events is “important”, because changing the order can change the entire story. The same is true in a computer program. Changing the order can, sometimes change the story as it moves from beginning to end. What’s different about the events of a computer program is they go from the end event, back to the beginning event. This is called a loop.

CHAPTERS: We often organize stories by chapters. Computer programs have similar structures called functions. Chapters and functions organize stories and programs into broad ideas that are different from each other, but related to whole story. What are the three “chapters” of the Code Of Me?

1.____________________
2.____________________
3.____________________

ACTIONS of the characters make up the details in each chapter. They build the broad ideas of chapters with smaller details where characters perform actions like driving a car, or growing a flower. We drive cars and grow flowers by following a specific order of instructions (like a math statement):

-Mr Reitzig adds two to ten to get twelve.
-Zack inserted the key and turned the key to start the car.

These actions in a computer program are called algorithms. Using the “characters” of each “chapter” write one sentence describing an action in each:

HOME:
SCHOOL:
HOBBY:
Sketch the Code Of Me like Scratch

Draw boxes for TIMELINE, CHARACTERS, SETTING, SERIES OF EVENTS, CHAPTERS and ACTIONS. You don’t have to fill them with details...yet.

example:
Assignment: Fill in all of the blanks to write your Code Of Me similar to the language we call JAVA! If you have questions about the weird symbols [*//*, ; , //, {}, ()], just ask.

/*StoryOfMe
Build a story of ME at HOME, SCHOOL and HOBBIES.
ME= You (who you call, "your name")
HOME = Made up of People, Places and Things or Events
SCHOOL = Made up of People, Places and Things or Events
HOBBY = Made up of People, Places and Things or Events
*/

me = your name; // global variable
______ = _______________; //another global variable

setting(); //setup
{
  me = _____; // specific to your story
}

series of events in important order(); //loop
{
  home(); // a function of your story
  {
    home = people, places, things, events; //local variable
    actions involving characters in this place, in order // algorithms
    1.__________________;
    2.__________________;
    3.__________________;
  }
  school(); // a function of your story
  {
    school = people, places, things, events; //local variable
    actions involving characters in this place, in order // algorithm
    1.__________________;
    2.__________________;
    3.__________________;
  }
  hobby(); // a function of your story
  {
    hobby = people, places, things, events; //local variable
    actions involving characters in this place, in order // algorithm
    1.__________________;
    2.__________________;
    3.__________________;
  }
}
}
The GRID: The blindfolded teammate is the “Pixel”. The other teammate is the “Programmer”.

Objective: Teams of TWO define “functions” and “conditional statements” (operations) used to move a Pixel from one color start point, through the grid, to the same color end point. Write them down as a list of possible operations. (5 minutes)

After the Pixel is blindfolded, the obstacles on the grid will be rearranged. The Programmer must write a ‘program’ to move the Pixel through the grid using only the list of possible operations created by the team and without breaking the rules. The first of three teams to finish, wins.

Rules:
1. Pixels must execute functions in written order. Conditional statements can ‘get between’ orders.
   Programmers can re-write (debug) the subsequent functions if conditional statements ‘break’ the program.

2. Pixels cannot go off of the grid. If they are at the edge, a conditional statement must correct their path.

3. Pixels cannot occupy the same space as another Pixel. If two Pixels collide each Pixel must stop, turn 180° and the Programmer must re-write (debug) a program to get to the goal in written order.

4. Pixels cannot move through objects. If objects shift out of place, the program is ‘delayed’ 5000 miliseconds.

Tips:
Write conditional statements imagining things that might happen between objects and Pixels.

Write simple functions like SCRATCH! Moving can be defined by a number of steps, or an amount of time.

All collisions slow you down and may need debugging. Try to write a program that avoids the most chances for collisions.

Computational Thinking:
Programs are limited to certain kinds of actions. You, the programmer, has the ability to increase those actions by creating new ones.

Screens are 2-dimensional grids with “X”, “Y” coordinates! A pixel is what lives at each grid space. Individual pixels can be located by changing its color.

Two pixels cannot have two different colors or “equal” two different things. Hence, no sharing space on the grid with another pixel, OR another object.

Computers are DUMB! The only, EVER do what a program tells them to do. When something goes wrong, it’s never “the computer’s fault”. It’s just doing what the programmer told it to do. You almost always need to do some debugging.