|  |  |  |
| --- | --- | --- |
| **Name:** John McClellan | **Contact Info:** jmcclellan@readingschools.org | **Date:**11/01/17 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Lesson Title :** Coding with Scratch | **Unit #:** 1 | **Lesson #:** 2 | **Activity #:** 3 |
| **Activity Title:** Tutorials with Scratch |

|  |  |
| --- | --- |
| **Estimated Lesson Duration:** | 4 Days |
| **Estimated Activity Duration:** | 2 Days |

|  |  |
| --- | --- |
| **Setting:** | Classroom |

|  |
| --- |
| **Activity Objectives: The student will be able to:** |

1. Access the Scratch Website and open a blank programming file.
2. Identify commands in the command window and manipulate each one
3. Personalize programs from walkthroughs
4. Relate the various scratch syntax to the pseudo-code they have written last lesson

|  |
| --- |
| **Activity Guiding Questions:** |

* What is Scratch?
* How can you make objects move?
* How does Scratch compare to other languages?
* How complicated can you make Scratch programs?

| **Next Generation Science Standards (NGSS)** | |
| --- | --- |
| **Science and Engineering Practices (Check all that apply)** | **Crosscutting Concepts (Check all that apply)** |
| ☒ Asking questions (for science) and defining problems (for engineering) | ☐ Patterns |
| ☐ Developing and using models | ☐ Cause and effect |
| ☒ Planning and carrying out investigations | ☐ Scale, proportion, and quantity |
| ☐ Analyzing and interpreting data | ☐ Systems and system models |
| ☒ Using mathematics and computational thinking | ☐ Energy and matter: Flows, cycles, and conservation |
| ☐ Constructing explanations (for science) and designing solutions (for engineering) | ☒ Structure and function. |
| ☐ Engaging in argument from evidence | ☐ Stability and change. |
| ☐ Obtaining, evaluating, and communicating information |  |

| **Ohio’s Learning Standards for Science (OLS)** |
| --- |
| **Expectations for Learning - Cognitive Demands (Check all that apply)** |
| ☒ Designing Technological/Engineering Solutions Using Science concepts **(T)** |
| ☐ Demonstrating Science Knowledge **(D)** |
| ☐ Interpreting and Communicating Science Concepts **(C)** |
| ☐ Recalling Accurate Science **(R)** |

| **Ohio’s Learning Standards for Math (OLS) and/or**  **Common Core State Standards -- Mathematics (CCSS)** | |
| --- | --- |
| **Standards for Mathematical Practice (Check all that apply)** | |
| ☐ Make sense of problems and persevere in solving them | ☒ Useappropriate tools strategically |
| ☒ Reason abstractly and quantitatively | ☒ Attendto precision |
| ☒ Construct viable arguments and critique the reasoning of others | ☒ Look for and make use of structure |
| ☐ Model with mathematics | ☐ Look for and express regularity in repeated reasoning |

|  |
| --- |
| **Unit Academic Standards (NGSS, OLS and/or CCSS):** |

Standard for Mathematical Practice #2 Reason Abstractly and Quantatively; Standard for Mathematical Practice #5 Use Appropriate Tools Strategically; CCSS High School Modeling Domain: *“When making mathematical models, technology is valuable for varying assumptions, exploring consequences, and comparing predictions with data.”*

|  |
| --- |
| **Materials**: (Link Handouts, Power Points, Resources, Websites, Supplies) |

* Chromebooks
* Tutorial Links:
  + Flappy Bird: <https://studio.code.org/flappy/1>
  + Video Tutorials: <https://scratch.mit.edu/help/videos/>
  + Blank program & step-by-step tutorials: <https://scratch.mit.edu/projects/editor/?tip_bar=home>
* Projector (Ipad?)

|  |
| --- |
| **Teacher Advance Preparation:** |

* Make sure you are in a place where the Wi-Fi is strong. This a web-based compiler, and will not work if there is not access to the internet.
* It might help to write out the complete program of the Flappy Bird tutorial as a guidance for the students to work through their pseudo-code.
* Keep the links in a single document, labelled, for the students to easily access the sites. Provide many different options for various functions (as many tutorials as you can find).

|  |
| --- |
| **Activity Procedures:** |

Day 1:

1. On the board, have the “Pre-bell” written, which is to write down the basic premise of Flappy Bird. Write down all the computer functions that they can about the game. This will last about 5 minutes, a few minutes past the bell. While they are working on this, share the link to a blank Scratch document and the Flappy Bird tutorial. Keep this in a Google Doc so that students can easily access it later on.
2. Go over the various parts of Flappy Bird. Students should have a feel by now how to write pseudo-code. See how far you can push your students to write up the pseudo-code. Since the premise is simple, they might be able to write out a complete program in pseudo-code. Spend no more than 20 minutes on discussion and creation.
3. After that, let them access the blank program first and explore the command window. See how much of it they understand on their own, how much of it needs to be explained, and how much of it has no parallel to what they’ve learned. (It’s okay if you don’t understand all of the functions of Scratch either. In fact, tell the students that when you encounter one that no one gets. Students should have confidence to use the coding software even if they don’t understand every single command). Take 5-10 minutes on this.
4. Next, have them click on their Flappy Bird Tutorial. Have them work side by side to one another to help each other out when the tutorial gets too confusing. This should take the rest of the bell to complete and tweak.
5. Have them send their final product to you as an exit slip.

Day 2:

1. Similar to yesterday, have the students brainstorm computer functions for a game, but this time allow for any simple “addicting” game. You might include a few examples to try to spark their ideas. Allow for the students to work in pairs on this, and after about 5 minutes of brainstorming functions, see if they can design some pseudo-code (another 10 minutes). While they are doing that, set up the projector so that the students can easily show off their pseudo-code.
2. Take a few minutes and let the pairs get up, talk about their game, and explain their pseudo-code. This shouldn’t take more than 10 minutes.
3. The rest of the bell, students should be working alone or in pairs on the tutorials laid out in the Google Doc. Let them know that they need to send you final results. They can be as deep or as simplified as they would like. Given that they have a lot of options, allow them to work until the end of the bell. Check to see what they are doing, and if possible, work through some of the tutorials on a projector/TV monitor quietly so that they can kind of see the process and speed in which other people can code.

**Formative Assessments:** Link the items in the Activities that will be used as formative assessments.

* Flappy Bird final result
* Both whole-class and small group discussion on Flappy Bird pseudo-code
* Tutorial results
* Second day pseudo-code informal presentation

**Summative Assessments:** These are optional; there may be summative assessments at the end of a set of Activities or only at the end of the entire Unit.

None for this lesson—will be assessed at end of second lesson.

|  |
| --- |
| **Differentiation:** Describe how you modified parts of the Lesson to support the needs of different learners.  Refer to Activity Template for details. |

The groups were divided into knowledge of coding where a top student was in each group. There were a few absences which made one group (the group with the student with a 504) struggle a little more than others. I had to work with this group a lot more than others and lower the expectations of the challenge for them to being able to construct a

|  |
| --- |
| **Reflection:** Reflect upon the successes and shortcomings of the lesson. |

Students loved the flappy bird work! There was a lot of laughter shared amongst the students as to the amount of control they had. I tried to talk that up as it was precisely the feeling I get when I design a code that works well. Students were able to fly through the flappy bird tutorial, so I immediately got them going on other tutorials. In fact, day 2 was rolled into day 1 for most of the students. For the students who were absent or needed extra time, I allowed for the second day. I did not anticipate things moving more quickly than planned!

The tutorials that were scheduled for day two went well. I allowed the students to divide into the their challenge/EDP groups and decide who was going to do what tutorial. This worked well, and although students preferred to write things down on paper as opposed to using Google Doc, it still worked remarkably well. Students were ready to proceed to the challenge a day early.