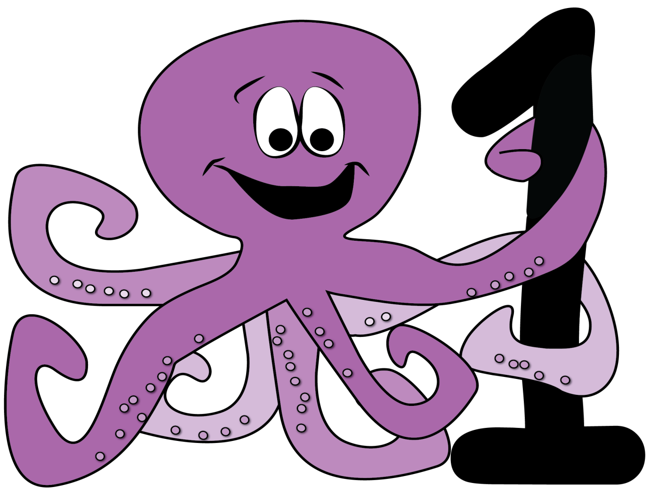
Module 1

Digital Storytelling

2014-2015



Diana Franklin
Computer Science
dfranklin@cs.ucsb.edu

Danielle Harlow
Education
dharlow@education.ucsb.edu

Table of Contents

Module 1 Digital Storytelling

Purpose

Module 1 introduces upper elementary school students to computational thinking and programming with our block-based, programming environment (LaPlaya). For computational thinking, this module teaches students about the importance of order in a program, breaking down complex tasks into their basic components, properly associating code with triggering events, and managing the complexity of several multiple objects acting at once. Further, students will learn programming concepts by implementing these ideas in LaPlaya. These include sequential programming, event driven programming, costume changes, and scene changes. Through the module, students will learn about and develop skills with the engineering design process. As a culminating project, they will apply all three areas to create an animated story.

Module 1 takes approximately 12 – 13 hours to complete. Approximately half of activities are conducted in the classroom (Fired Up); students complete the remaining activities on computers (Wired Up).

Overview of Lessons

Day	Activity	Location (Time)	Description
1	Computer Science and Programming	Classroom 60 min.	Students watch a video introducing them to computer programming (Google's *Made with Code*) and discuss how computer science relates to their everyday lives. Following, students explore challenges to receiving and giving directions through in a set of FiredUp activities (*Following Directions* and *Drawing Shapes*).
2	Sequencing	Lab 45 min.	Students create their first programs in LaPlaya and are introduced to sequential thinking in a series of connect the dots tasks.
3	Introducing Engineering Design Thinking	Classroom 15 min.	Teachers lead a classroom discussion: What is design thinking? What is the process?
4	Breaking Down Actions	Lab 45 min.	Students break down motion into smaller steps by identifying both direction and type of movement. They program a bear sprite to move in different directions to reach a honey sprite through several scenarios.
5	Defining an Engineering Problem	Classroom 45 min.	Teachers first lead a classroom discussion on how engineers define a problem. Following, teachers introduce students to design-

			notebooks.
6	Event-Driven Programming – When Sprite Clicked	Lab 45 min.	Students begin to engage users in their programs by creating scripts that run on an event: when a user clicks on a sprite. Students program fruit to grow and fall off a tree (*Fruit Trees, Part 1* & *2*), and planets to say their names when clicked (*Planets*). As a bonus, students add sound to a virtual piano (*Bonus: Piano, Part 1*).
7	Brainstorming Digital Stories	Classroom 45 min.	Teachers lead a classroom discussion on characteristics of creating a good story. Students brainstorm story ideas to implement in LaPlaya in their final project.
8	Storyboarding Digital Stories	Classroom 45 min.	Students create storyboards in their design-notebooks for their digital stories.
9	Event-Driven Programming – When Key Pressed	Lab 45 min.	Students expand their expertise in event-driven programming by assigning actions to different keys. Tasks include programming a flying rocket (*Rocket* and *Rocket Countdown*), a car cruising California (*California Geography),* and a basket for collecting apples (*Fruit Trees, Part 3*). As a bonus, students can program a digital piano (*Bonus: Piano, Part 2*).
10	Thinking about a User	Lab 45 min.	Students learn the difference between being the user of software and being the designer of software. In these tasks, students analyze finished LaPlaya lessons to determine what actions may be hidden and discuss how they can engage the user in their own programs.
11	Flow Charts & The Application Cycle	Classroom 45 min.	Students create flow charts for their digital stories, and plan how the story will move from scene to scene. Students learn about the application cycle of the Engineering Design Process (programming, testing and evaluating, and redesigning) and will be given instructions on how and when to work on their digital story using LaPlaya.
12	Event-Driven Programming – On Other Sprite Clicked	Lab 45 min.	Students learn how to program sprites to act when a user interacts with another sprite. Students program cars to drive when a user clicks a traffic light (*Intersections, Part 1* & *2*), candy to fall when the user clicks a piñata (*Piñata, Part 1*), and fruit to fall when a user

			clicks a tree (*Fruit Trees, Part 4*).
13	Initializing Programs	Lab 60 min.	Students see the importance of initialization through a demo of two stories which vary in what was initialized (*The Tortoise and the Hare parts 1* & 2). Afterwards, they initialize candy to start inside a piñata sprite (*Piñata, Part 2*) and animals to start behind the starting line in a series of races (*Animal Sprint: Horse, Cat* and *Rooster*).
14	Animating Sprites	Lab 45 min.	Students learn how to use costume changes to create animations. Students create increasingly complex dances for different characters (*Ballerina, Pick A Dancer, Dance Party!)* and then draw and animate their own dancer (*Draw Your Own*).
15	Changing Scenes	Lab 45 min.	Students use a combination of background and costume changes to create multiple scenes. They practice changing scenes through the life cycle of a plant.
16	Creating Digital Stories	Lab 45 min.	Students create their own digital stories using their programming knowledge and the materials they created in their design-notebooks.
17	Finishing Digital Stories	Lab 45 min.	Students finish their digital stories.
18	Sharing Your Work	Lab 45 min.	Students share their digital stories with their classmates or others.

1: Computer Science & Programming

Teacher Lesson Plan

Lesson Rationale-

This brief lesson introduces students to computer science and computer programming. Both ideas are connected to the world around students and what they will be doing in this module.

Objectives-

Students will able to articulate what computer science is and what computer programming is. Students will also be able to see programming and code in the world around them and relate computer programming to the LaPlaya interface.

Standards Emphasized-

CSTA Computer Science Standards:
L1:6:CT- The student will be able to Demonstrate how a string of bits (code) can be used to represent alphanumeric information & Understand the connections between computer science and other fields.
L1:6:CPP- Identify a wide range of jobs that require knowledge or use of computing.
L16:CD- Understand the pervasiveness of computers and computing in daily life (e.g., voice mail, downloading videos and audio files, microwave ovens, thermostats, wireless Internet, mobile computing devices, GPS systems).

Materials-

Teacher	Student
Device to show a YouTube video with (TV, Projector, Computer, etc.)	Student worksheet titles "Computer Science & Programming" and pencil/pen Design Notebook (optional)

Learning Tasks

(Total Approx. Time: 15-30 minutes)

Approx. Time	Learning Tasks	Purpose/Instructional Strategies
3-5 min.	Introduce main concepts that the student: Computer Science, Code, and Programming.	Students use the terminology of the field and vocabulary used throughout the module. *You may want to spend extra time reviewing these concepts to ensure that students understand them fully.*
3-5 min	Watch the Google film "Made w/Code" by going to www.madewithcode.com and	This video introduces how code is small bits of information embedded in the world around us (DNA, words, numbers, etc.). It

	clicking "WATCH THE FILM" *(note that this is a YouTube video and some computers may have restricted access to this site)*.	also reinforces the idea that coding and computer programming are valuable careers, and especially important for girls (who are vastly underrepresented in this field).
5-10 min.	Class discussion about the video and how the concepts they learned relate to one another. Students will answer the questions posed to them on their worksheets afterwards.	Emphasize the importance of computer science (there will be 1.5 million new jobs in the field by 2018, it teaches important problem-solving skills, and can be used in many other fields) and how code (which is used to program computers) is all around us and the blocks in LaPlaya are also code.
3-5 min.	Students will answer the questions posed to them on their worksheets after the discussion. These questions help them relate the concepts and tie them to everyday life and the LaPlaya interface they will be using.	Give students a few minutes to respond to their worksheet questions about these concepts. You may want to walk around and help students who may be struggling to find the connections between concepts or relating them to everyday life or the LaPlaya interface.

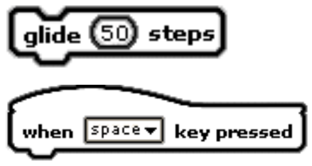
Name/Number: ____________________

Computer Science & Programming

What is **computer science**?	The study of COMPUTERS and their PROCESSES including hardware, software, their applications, and their impact on society.
What is **code**?	Small bits of INFORMATION (numbers, letters, words, symbols) that come together to create INSTRUCTIONS for something (or a PROGRAM).
What is **programming**?	Writing the INSTRUCTIONS a COMPUTER must follow (a PROGRAM). This also includes thinking about how a USER may interact with that PROGRAM.

Your teacher will show you a video about **code** and how it's all around us and discuss the importance of **computer science** and **programming**.

Think about the questions below and respond:

What is an example of **code** that you see in your everyday life?	______________________ ______________________ ______________________
Are these blocks examples of code? Why or why not?  glide (50) steps when [space ▾] key pressed	______________________ ______________________ ______________________ ______________________ ______________________
How does **code** relate to computer **programming**?	______________________ ______________________ ______________________
Why do you think it is important to learn about **computer science**?	______________________ ______________________ ______________________

Following Directions Fired Up

Teacher Lesson Plan

Lesson Rationale-

Computers follow a set of directions exactly in the form they are given. Unlike humans, computers are not able to interpret what one says. The ability to communicate a set of directions to other humans in a precise way is a precursor to being able to communicate a set of directions to a computer. The purpose of this exercise is to realize the challenge in such limitations. The end goal is students recognize how precision is needed to make computers do what they want it to.

The result of the lesson will be red and green "tents" for use in determining which students need help in the computer lab.

Objectives-

Students will be able to:

- Follow written directions
- Recognize the precision necessary for written directions to be successful.

Standards Emphasized-

Materials-

Teacher	Student
One 8x10 thick red paper for ½ of the students and one 8x10 thick green paper for ½ of the students	

Learning Tasks-

(Total Approx. Time: 15-30 minutes)

Approx. Time	**Learning Tasks**
5 min.	Pass out red paper to half of the students and green paper to half of the students.
10 min.	Verbally give them directions to make the "tent". Sample directions are given below.
5-10 min.	Explain what the result is for in the lab. Discuss what would have made the directions easier.

Sample directions:

1) Place the piece of paper on your desk so that it is wide, not tall.

2) Take the right-hand edge of the paper and bring it over to meet the left-hand edge, flattening it to create a fold in the middle of the paper.

3) Unfold it and fold it backwards along the same line.

4) Rip the paper along that line.

5) Find someone with a different color piece of paper than you have and trade one of your halves for one of their halves.

6) For each of your two separate pieces of paper (that are different colors):

    a) Place it on your desk so that it is wide, not tall.

    b) Take the right-hand edge of the paper and bring it over to meet the left-hand edge, flattening it to create a fold in the middle of the paper.

Now you have two pieces of paper that will stand up on a desk. If you are doing fine, working on your project, then place the green paper over the red paper. If you need help, place the red one over the green one to signal that you need help. Do not stop working – keep trying to figure out your problem. I will come and help you when I can.

Drawing Fired Up

Teacher Lesson Plan

Lesson Rationale-

Computers follow a set of directions exactly in the form they are given. Unlike humans, computers are not able to interpret what one says. The ability to communicate a set of directions to other humans in a precise way is a precursor to being able to communicate a set of directions to a computer. The purpose of this exercise is to strengthen students' skills in giving directions to other people, including the specific skills listed in the objectives below. The end goal is students recognize how precision is needed to make computers do what they want it to.

Objectives-

Students will be able to:

- Write procedures.
- Develop the ability to sequence information logically.
- Incorporate the following language into a procedural text:
    - Appropriate terminology
    - Time linking words
    - Action verbs
    - Precise use of adverbs and adjectives
    - Simple present tense
    - Reference to the reader in a general way (or not at all)
    - Appropriate headings.

Standards Emphasized-

Common Core State Standards
CCSS.ELA-Literacy.RI.4.3 - Explain events, procedures, ideas, or concepts in a historical, scientific, or technical text, including what happened and why, based on specific information in the text.
CCSS.ELA-Literacy.W.4.2 - Write informative/explanatory texts to examine a topic and convey ideas and information clearly.
CCSS.ELA-Literacy.W.5.2 - Write informative/explanatory texts to examine a topic and convey ideas and information clearly.
CCSS.ELA-Literacy.SL.4.4 - Report on a topic or text, tell a story, or recount an experience in an organized manner, using appropriate facts and relevant, descriptive details to support main ideas or themes; speak clearly at an understandable pace.
CCSS.ELA-Literacy.SL.5.4 - Report on a topic or text or present an opinion, sequencing ideas logically and using appropriate facts and relevant, descriptive details to support main ideas or themes; speak clearly at an understandable pace.

Materials-

Teacher	Student
Worksheets with pictures on one side and	Pen/pencil

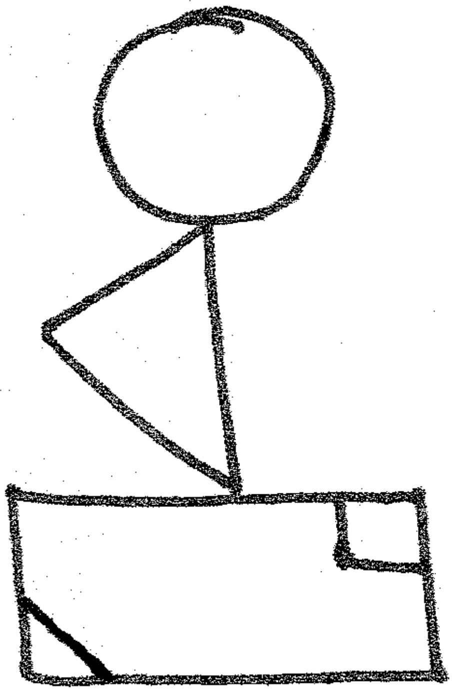
space to write directions on the other side (cut in half to separate Picture A from Picture B) Projector to show class drawings (optional)	

Learning Tasks-

(Total Approx. Time: 15-30 minutes)

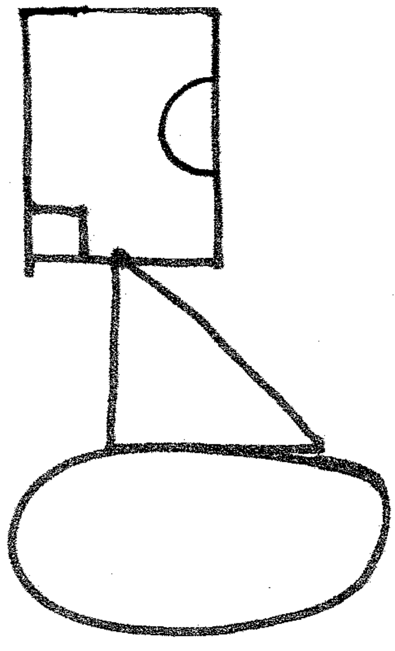
Approx. Time	**Learning Tasks**
0-5 min.	Divide the class into two groups. Give one group Picture A and Picture A Instructions and the second Picture B and Picture B Instructions. Then, task each student with writing simple "how to" instructions for someone in the other group to follow on the lines provided in the "Picture A/B Instructions".
5 min.	Have students cut or tear the picture away from their instructions. Collect all instructions from each group and randomly distribute to these to the other group of students (students with picture A will get instructions for picture B and vice versa). Now, task students with creating a drawing based on the instructions provided.
5 min.	After students create the pictures, show both pictures to the entire class. Pair students up (one from each group) to discuss what could have been improved about the directions. They may reference the drawings were created from their directions.
5 min.	As a class, ask students to share about the activity. Prompt them to consider: What aspects of the instructions helped them accurately complete the drawing? What aspects of the instructions could have been improved? Students will like start to introduce how instructions could be more specific, such as how big to draw a circle, which way to orient a triangle, etc. Then, ask the same pairs of students to collaborate and edit one another's directions,
5 min.	Finally, present the anchor chart (see attached) and explain the elements of procedural writing. Now, ask students to write directions for Picture C using the elements of procedural writing.

Picture A



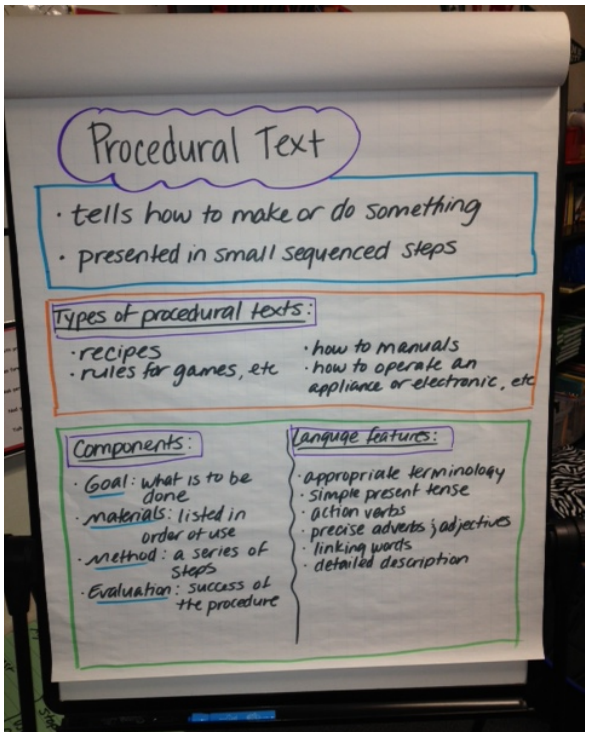
Picture A Instructions

Picture B



Picture B Instructions

Procedural Writing



Procedural Text

- tells how to make or do something
- presented in small sequenced steps

Types of procedural texts:

- recipes
- rules for games, etc
- how to manuals
- how to operate an appliance or electronic, etc

Components:

- Goal: what is to be done
- materials: listed in order of use
- method: a series of steps
- Evaluation: success of the procedure

Languge features:

- appropriate terminology
- simple present tense
- action verbs
- precise adverbs ; adjectives
- linking words
- detailed description

The purpose of a procedure is to tell the reader how to do or make something.

The information is presented in a logical sequence of events, which is broken up into small sequenced steps.

Types of Procedural Texts
Texts that instruct how to do a particular activity: recipes, rules for games, science experiments, road safety rules, how to do it manuals.

Texts that instruct how to operate things: how to operate an appliance, a machine, a photocopier, or computer.

Components of Procedural Writing
A procedure usually has four components:

1. *Goal* – states what is to be done
2. *Materials* – listed in order of use, includes items needed to complete task
3. *Method* - a series of steps
4. *Evaluation* - how the success of the procedure can be tested

Headings, subheadings, numbered steps, diagrams, and photographs are often used to help clarify instructions.

Language Features of Procedural Writing

- Appropriate terminology
- Simple present tense (*Stir the mixture until it boils*).
- The reader is often referred to in a general way (*you cut, one cuts, cut*).
- Action verbs *(cut, fold, twist, hold etc)*
- Precise use of adverbs or adjectives (*slowly unwind)*
- Linking words to do with time *(first, when, then)* are used to connect the text
- Detailed information on how *(carefully, with the scissors)*; where *(from the top)*; when *(after it has set)*
- Detailed factual description *(shape, size, color, amount)*

2: Sequencing

Teacher Lesson Plan

Lesson Rationale-

This lesson introduces students to the KELP-CS programming interface, which we call LaPlaya. LaPlaya runs through an internet browser and uses block-based programming. In LaPlaya, students program sprites with scripts. **Sprites** can be images of animals, people, backgrounds, or inanimate objects. **Scripts** are a short program or collection of blocks that instruct a sprite how to behave and under what conditions to react. Scripts are constructed of blocks that are dragged onto the programming area and snapped together.

This activity allows students to get familiar with creating scripts (drag from the block list into the script pane) and running scripts (reset the program then press the green flag). They also learn that scripts must start with a block that tells when to run. The rest of the instructions must be placed in the proper order.

Objectives-

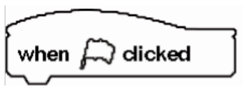
Students will be able to:

- Drag and drop blocks into programming areas.
- Combine multiple blocks to create script.
- Identify different areas of the LaPlaya interface (maneuver through block categories).
- Recognize that a script's action takes place on the stage.
- Learn that sprites are programmable agents.
- Organize blocks in the correct order for a simple script.
- Run a script using green flag button.
- Create a line on the stage using the pen down block.

Vocabulary-

Stage: Picture with the sprites on it where all of the action occurs
Sprite: A picture on the stage that you can control using scripts
Sprite List: Area of LaPlaya where you select what sprite you are programming.
Block: One command for a sprite to follow
Program: A set of instructions that are given to a computer to follow
Block List: Shows all available blocks in a lesson. Organized by block type such as motion, looks, and sound.
Script: A very short sequence of blocks (instructions) for a sprite to follow
Script Window: Area of LaPlaya that visualizes all written scripts.

New Blocks -

Block	**Block Name**	**Block Description**
	When Green Flag Clicked	Event block that starts a script. Script runs when a user clicks on the green

		flag button.
	Glide [ ]to [ ]	Motion block that moves a sprite with a speed (slowly, normally, quickly) to a specific sprite.
	Clear	Pen block that removes all pen lines.
	Pen down	Pen block that starts drawing.
	Pen up	Pen block that ends drawing.
	Set pen color to __	Optional: Pen block that changes the pen color.

Standards Emphasized-

Materials-

Teacher	Student

*Learning Tasks (*Total Time: 45 minutes)

Approx. Time	**Learning Tasks**	**Purpose**	**Student Directions**
? min.	*Task 1* Starting a Roof	Teacher uses task one as a demonstration to introduce students to LaPlaya. Teacher shows students different parts of the interface (sprites, scripts, stage, sprite list, blocks, block list, scripts window) and shows how to create a script. Students will then complete the task on their own at the computer.	In this task, you are learning how to make LaPlaya draw a line between two sprites. The script you need to make is already shown. You just need to make it yourself. 1) Click on the pen in the sprite list (if it is not already chosen) 2) Look for the proper blocks to put into the script by clicking on "Motion" and "Start" category buttons. 3) When you find the block, click and drag it onto your script area. 4) Once you have all of the sprites, you are ready to run it. 5) Click on the "ready" button

			to get everything ready. 6) Click on the "green flag" button to make it go.
	Task 2 Building a Roof	This builds on the first project. Now that students have drawn one line, they need to draw of a triangle. They draw a line to connect dots from 2 to 3 to 1.	Now you are ready to make a triangle - the roof of the house! You already completed one line. Now do the rest. Make sure you draw in the right order!.
	Task 3 Building a House	In this part, students finish the house and get a fun completion screen.	Now, using what you learned in the first two parts, finish the house!
	Task 4 - Bonus Draw a Cat	In this task, students are given a connect-the-dot script that is not quite in order. Once they run the program (click on green flag), they need to determine how the dots are out of order, and then fix the script.	Uh, oh! Someone did not get this script quite correct. Run this script by clicking on the green flag, and look closely how the drawing is made. Then go and fix the script so it will draw a cat!

Suggestions for Students who Finish Early-

Common Student Errors -

Teaching Hints -

3: Introducing Engineering Design Thinking

Teacher Lesson Plan

Lesson Rationale-

This lesson introduces Engineering Design Thinking to students, which they will be using for the remainder of the module to design and create their own computer program telling a digital story. It is important for students to understand the importance of this design process and how it is used in many fields, not just engineering.

Objectives-

Students will understand that the process of Engineering Design Thinking is much like the scientific process and is used in a wide range of fields (science, engineering, computer science, product design, etc.). Students will also get to know the steps in the design process and will start to think about how they can apply them in designing their own computer program.

Standards Emphasized-

CSTA Computer Science Standards:
L1:6:CT- Understand and use the basic steps in algorithmic problem-solving (e.g., problem statement and exploration, examination of sample instances, design, implementation, and texting) & Make a list of sub-problems to consider while addressing a larger problem.
L1:6:CL- Identify ways that teamwork and collaboration can support problem solving and innovation.

Materials-

Teacher	Student
Optional- projector or something else to show the design process to whole class.	Student worksheet titled "Engineering Design Thinking" Design Notebook (optional)

Learning Tasks -

(Total Approx. Time: 15-20 minutes)

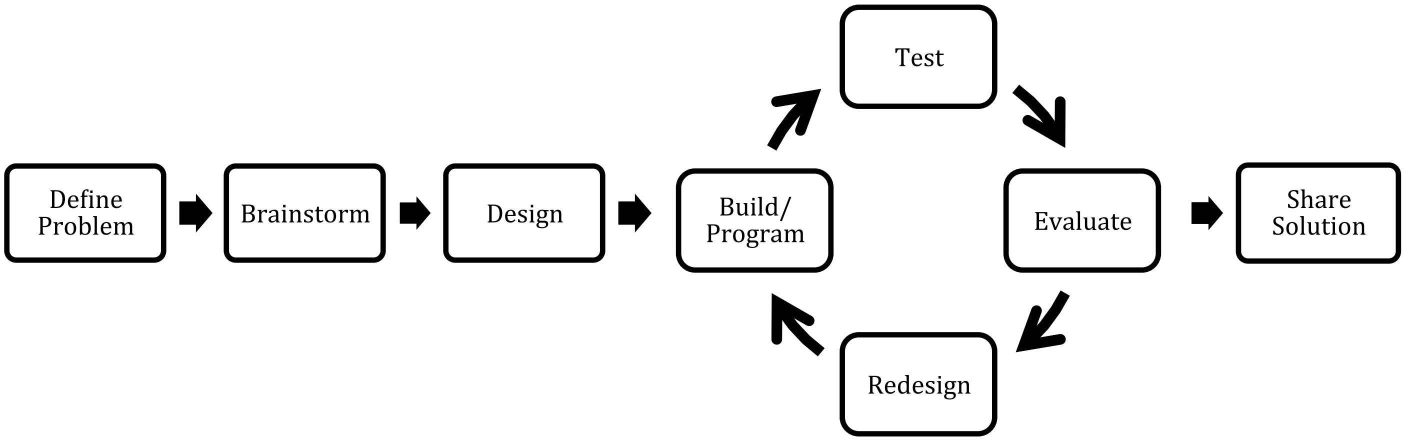
Approx. Time	Learning Tasks	Purpose/Instructional Strategies
3-5 min.	Introduce the process of Engineering Design Thinking and discuss with students how people in many different fields may use this process (scientists doing experiments, engineering solving problems,	The process of Engineering Design Thinking is important not only in science and engineering (as is indicated in the Next Generation Science Standards), but also in a wide array of fields that students should be aware of. Design Thinking is a powerful tool that can be used to solve simple or

	product designers creating a new piece of technology, clothing, toy, etc., architects designing a building, computer programmer creating a new program or app).	complex problems in many contexts and is a valuable skill for students to understand and master. Ask the students to think of ways the design process can be used in other ways (they may think of novel ways they to use this process themselves like trying to reach something high or building a fort).
10-15 min.	Use a simple example of a problem that can easily be solved *(make an alarm for your bedroom door)* and go through the steps of the design process with this example. **Define the problem**- *Someone is stealing your allowance so you want to keep it safe in your room; design an alarm. Constraints are that you only have what is in your backpack to make it out of.* **Brainstorm**- *What can I do with my materials? Put a string across the door and attach something that will make noise to them.* **Design**- *Draw up a plan to figure out what materials you need, how much of them, and where they will all go.* **Build**- *Build the alarm.* **Test**- *Try out the alarm and see if it works on you first.* **Evaluate**- *What does/doesn't work with the alarm? What needs to be fixed?* **Redesign**- *Add things you think you may need, fix what didn't work, make your alarm better!* **Share**- *Ask you friend to come test out the alarm and see if he has suggestions for improving it.*	By going through the process with the class using a fairly simple engineering problem students will be able to work though each step and will get a fuller understanding of the Engineering Design Process. By doing this they will be able to transfer their knowledge and experience with the design process to the new context of computer programming using LaPlaya in the future. You may want to briefly discuss how computer programmers use this design process (like they will be doing). Go over how instead of building something physical (an alarm, machine, toy, building, etc.) programmers use the computer to build programs using code. They run those programs to test them and find any bugs (evaluating) before redesigning and fixing their programs to share with others.

Name/Number: ______________________

Engineering Design Thinking

What is **Engineering Design Thinking**?	The process ENGINEERS (and others) use to come up with SOLUTIONS to PROBLEMS.	
What is the process of **Engineering Design Thinking**?	**Define a Problem**	You or somebody else identifies a problem that needs to be solved. You also identify constraints that you must work within.
	Brainstorm Ideas	Working alone or with other you come up with many ideas to solve the problem and narrow them down to find the best one.
	Design	Planning what you will do so solve the problem. *(This includes storyboarding and creating a flow chart of your computer program.)*
	The Application Cycle	Building/Programming *(on the computer)*, Testing *(Trying it out)*, Evaluating *(what does/doesn't work)*, and Redesigning *(making it better)*. This is a cycle that you will go through multiple times before creating a finished solution to the problem.
	Sharing your Solution	It is important to share you solutions with others not only so they can benefit from them, but also so they can provide feedback to help create an even better solution.



4: Breaking Down Actions

Teacher Lesson Plan

Lesson Rationale-

This lesson introduces students to breaking down actions. In our everyday lives, we act in specific ways, often in a specific order without realizing it. For example, when you brush your teeth you usually put the toothpaste on the toothbrush before cleaning your teeth (not the other way!). In programming, computer scientists break down larger action in to small, discrete pieces as well. They pay particular attention to the order of these actions. When a program runs, the pieces of code they create occur in a specific order, from one command to the next. This activity prompts students to breakdown the movement of a sprite into a series of blocks, by specifying both the direction and the type of motion.

Objectives-

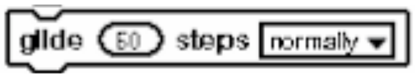
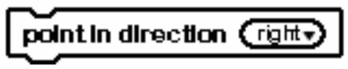
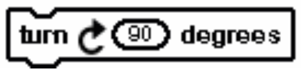
Students will be able to:

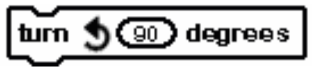
- Create a script that moves a sprite in a direction.
- Recognize that some movement can be done with few blocks (three glide blocks of 50 steps is the same as one glide block with 150 steps)
- Change the number of steps a sprite can make by typing directly what they want into the block.
- Recognize that there are multiple ways to solve a problem.
- Begin to understand that some solutions are more effective than others.

Vocabulary-

Sequence: A particular order in which event, movements or things follow each other.
Sprite: A picture on the stage that you can control using scripts
Script: A very short sequence of blocks (instructions) for a sprite to follow
Block: One command for a sprite to follow

New Blocks -

Block	**Block Name**	**Block Description**
 glide 50 steps normally	Glide ___ steps [normally]	Motion block that moves a sprite a variable length, such as 50 steps, with a specific speed (slowly, normally, quickly)
 point in direction right	Point in direction [ ]	Motion block that orients a block in four ways: right, left, up, and down.
 turn 90 degrees	Turn __ degrees right	Motion blocks that rotates a sprite 90 degrees right.

 turn ↺ 90 degrees	Turn __ degrees left	Motion blocks that rotates a sprite 90 degrees left.

Standards Emphasized-

Materials-

Teacher	Student

*Learning Tasks (*Total Time: 45 minutes)

Approx. Time	**Learning Tasks**	**Purpose**	**Student Directions**
? min.	*Task 1* Fired Up	Teachers introduce the challenges of breaking down action through a Fired Up activity (*Be a Robot*).	
	Task 2 Honey Walk	Teachers use this task to demonstrate how to program sprites to move. In LaPlaya, the movement of a sprite needs direction and a specific type of motion. Following the teacher demonstration, students complete tasks on their own at their computers.	Help the bear get to her honey! The script you need to create is shown. Look in the categories and find the blocks to create the script. Each square on the screen is 50 steps.
	Task 3 Honey Hike	Students program a bear sprite to move across three grid blocks to the honey sprite	Help the bear get to his honey! This time, the bear needs to get farther. How many steps is it?
	Task 4 Honey Climb	In this task, students need to make the bear go vertically. This requires a turn then glide. The solution is provided for them, and they need to copy it and run it.	Now the bear needs to climb up to the honey pot! The correct script is shown to you. Notice that you need to turn the bear before having her glide to it.
	Task 5 Honey Path	Now the students need to combine both skills - turning and navigating. The path is an L shape.	Help the bear get to his honey! Figure out how to navigate the bear to the honey
	Task 6	In this task, students are asked to find what hidden things happen.	Just as in last project, figure out how to get the bear to

	Honey Adventure	This is to point out that there are more scripts going on than they see and to emphasize that this was done as a design decision for them, the user. The only scripts being hidden are complex scripts, ones that we don't want them to worry about. But we still want fun things to happen. So, their job is to find them! They should complete a very short list like this: Sprite X does Y when Z happens Honeypot tips over when bear hits it Honeypot says "Good job" when Bear hits it Bushes say "" when Bear hits them	the honey. Don't hit the bushes!

Suggestions for Students who Finish Early-

- Invite student to practice the activity by using both relative and absolute direction change.

Common Student Errors -

Teaching Hints -

Breaking Down Actions Fired Up

Teacher Lesson Plan

Lesson Rationale-

It is actually not easy to make a robot act like a human! Robots, like computers, interpret directions exactly in the form they are given. They do not guess what someone means to say. They do exactly what you tell them, and do not act without specific instructions. In addition, computer commands tend to be much simpler than the types of directions 4th graders typically use. he purpose of this exercise is to strengthen students' skills in breaking down actions into their smaller parts. The end goal is for students to recognize the need to do so and be able to do so when necessary in their programming projects as well as use an iterative process for development.

Objectives-

Students will be able to:
- Develop the ability to break a complex task into a sequence of simple tasks
- Develop the ability to sequence information logically.

Standards Emphasized-

Materials-

Teacher	Student

Learning Tasks -

(Total Approx. Time: 15-30 minutes)

Approx. Time	**Learning Tasks**
0-5 min.	Divide the class into pairs of students who sit far apart from each other (Partner A and Partner B). The purpose is to produce a set of instructions that will lead one student from his/her desk to the door. Write the three allowable instructions on the board: Point towards (which wall), Turn (right/left) and Walk (#) steps.
10 min.	The pair goes to Partner A's desk. Partner B attempts to create directions for Partner A. Partner A follows these **slowly** first with eyes open, then when they believe it is correct, with eyes closed.
10 min.	The pair goes to partner B's desk. Partner A attempts to create directions for Partner B. Partner B follows these **slowly** first with eyes open, then when they

	believe it is correct, with eyes closed.
5 min.	Discuss how students solved the problem. Ask how they moved partners in a new direction – did they ask their partner to turn or point in direction. Then discuss the process and proposed solution. Encourage students to test their ideas and modify their directions. This may take multiple rounds! Everyone is not expected to provide good directions on the first try. It is an **iterative** process, and mistakes lead to learning.

5: Defining an Engineering Problem

Teacher Lesson Plan

Lesson Rationale-

This lesson introduces students to the engineering problem that they will be solving through the programming of a digital story. The teacher gives students a topic (California Missions, a book report, a science demonstration, etc.) that students will need to create their program within and other constraints (the LaPlaya programming environment, time, etc.). Students will then write a problem statement about what the problem is, and why it is important to solve.

Objectives-

Students will be able to write a problem statement that includes who has a problem, what the problem is, and why it is important to solve. Students will also be able to identify constraints for their project.

Standards Emphasized-

CSTA Computer Science Standards:
L1:6:CT- Understand and use the basic steps in algorithmic problem-solving (e.g., problem statement and exploration, examination of sample instances, design, implementation, and texting) & Make a list of sub-problems to consider while addressing a larger problem.

Materials-

Teacher	Student
"Engineering Design Thinking" worksheet (for reference) Computer/Projector to show example projects	"Engineering Problem and constraints" worksheet Pencil or pen Design Notebook (optional)

Learning Tasks -

(Total Approx. Time: 30-45 minutes)

Approx. Time	Learning Tasks	Purpose/Instructional Strategies
2-3 mins.	Use the "Engineering Design Thinking" worksheet that students were given previously as a reference to introduce this first step in the Design Thinking process; Define a Problem. Connect this with the	By referring back to what students have already gone over it reinforces the parts of the design cycle and contextualizes them with the visual provided on the "Engineering Design Thinking" worksheet. Going over the new vocabulary is also essential for helping students to understand the parts of an

	definitions at the top of their Engineering Problems and Constraints" worksheet.	engineering problem and differentiating the constraints they must work within.
3-5 mins	Introduce the Engineering Problem that you have decided for you class to work on. Some suggestions include: • A younger class wants to know about natural disasters. How can we share what we know with them? I (the teacher) want to know what happens in the book James and the Giant Peach, but I don't have enough time to read it. Can you summarize it for me?	By introducing and engineering problem in this way it gives students more of a purpose for creating their digital stories. They also include all of the elements necessary for students to complete a problem statement: • Who has a problem • What is the problem • And, Why is the problem important You may have your students create a digital story for any sort of engineering problem (it is up to you), but the 3 parts of the problem statement (see above) should be clear to students)
3-5 mins.	Have students complete #1 – 3 of the problem statement on their "Engineering Problems and Constraints" worksheet.	This is the first, and most important part of defining an engineering problem, and students can and should refer back to this throughout the creation of their programs.
5-10 mins.	Help students identify the constraints they will need to work within when solving this engineering problem (They must use LaPlaya, they will have only a set amount of time, etc.) Students can complete #4 of the problem statement on their worksheet.	The basic constraints include the LaPlaya programming environment and time (which is up to you), but students may come up with interesting ideas for constraints that may be appropriate. Helping students with this process is essential because these limitations should be explicit before they begin working on the more open-ended programming environment so they don't get lost in the many options available.
5-10 mins.	Show students example project(s) from LaPlaya (preferably on a topic that students will not be doing).	These examples give them an idea of what LaPlaya can do and what they are capable of doing with it.
3-5 min.	Go over with students how they will be recording their data and progress throughout the design thinking cycle of programming thei9r digital story (this is up to you).	It is crucial that students get into the habit of recording things and keeping track of what they have done, need to do, and will do. You may decide to have you students create a design notebook or simply a binder of the worksheet provided with some extra paper; How they record things is up to you.

Name/Number: ______________________

Engineering Problems and Constraints

What is an engineering **problem**?	Something that you, another person, or a group of people NEED. Sometimes you might identify a problem yourself or be asked by someone else to help solve a problem they have identified.
What are **constraints**?	LIMITATIONS that you must work within while trying to solve an engineering problem. This could be time, money, materials, or knowledge.
How do you write a **problem** statement?	1. WHO has the **problem** or need? 2. WHAT is the **problem** or need? 3. WHY is it important to solve (the **goal**)? 4. What **constraints** will I need to work within?

For the digital story you will be creating with LaPlaya your teacher will ask you to help solve a problem that they have identified.

Fill out the problem statement below.

1. WHO has the **problem** or need? *(Your teacher, you, someone else?)*	
2. WHAT is the **problem** or need? *(Your teacher will tell you this)*	
3. WHY is it important to solve? *(What is the **goal** of the project?)*	
4. What **constraints** will I need to work within? *(Time, Knowledge of Programming, LaPlaya limitations, something else?)*	

6: Event-Driven Programming: When Sprite Clicked

Teacher Lesson Plan

Lesson Rationale-

In this lesson, students create a program that responds to a user, specifically when a user clicks on a sprite on the stage. To create a fun game, computer scientists can create programs that require a user do something such as click his or her mouse or press a button. These actions are called events. In LaPlaya, students start a script with event blocks (brown blocks, with a curved top). Though they may not have realized it, they already used event blocks in the last two activities (when green flag pressed). In this activity, students will learn how to create a script that runs when a user clicks on a sprite on the stage.

Objectives-

Students will be able to:

- Change the relative size of sprites.
- Engage a user in their programs.
- Create scripts that run when a user interacts with the stage.

Vocabulary-

User: The person playing the game, running the program, etc.
Control Blocks: Blocks that determine when something should happen.
Event: Something that the user does (click sprite, press button, etc.)
Interactive: A program that responds to things the user does.

New Blocks -

Block	Block Name	Block Description
	When Sprite is clicked	Event block that runs a script when a user clicks on the specific sprite on the stage.
	Increase size by ___	Look block that will make a sprite smaller by a variable increment (student types in amount of change).
	Decrease size by ____	Look block that will make a sprite bigger by a variable increment (student types in amount of change).
	Glide __ steps [right]	Motion block that moves a sprite a variable increment in a direction (left, right, up, or down)
	Say ___ for __ sec	Look block that creates a pop up text bubble above a sprite that stays on

		screen for variable time.
	Think ___ for __ sec	Look block that creates a pop up think bubble above a sprite that stays on screen for variable time.

Standards Emphasized-

Materials-

<u>Teacher</u>	<u>Student</u>

*Learning Tasks (*Total Time: 45 minutes)

Approx. Time	**Learning Tasks**	**Purpose**	**Student Directions**
? min.	*Task 1* Growing Apples	Teachers use this task to demonstrate how to create a script with the "When Sprite is Clicked" using a program that changes the size of apple sprites. Following the teacher demonstration, students complete the task independently on their computer. Remember that there are two parts to every script: 1) When you want it to happen (Events) 2) What you want it to do (Motion, Looks, or other blocks)	Program each green apple sprites to get bigger when they are clicked. Hint: Use "increase size by" Remember that there are two parts to every script: 1) When you want it to happen (Events) 2) What you want it to do (Motion or Looks) The script you need to create is shown in the first apple.
	Task 2 Rotten Apples	In this task, students are asked to make the rotten apples fall to the ground when they are clicked. This is to reinforce the "on sprite clicked" that they learned in the previous one with the motion blocks they learned in the prior activities.	Oh, no! Some of the apples are rotten! Program a script in the rotten apples that will make the apple fall to the ground when it is clicked. Remember that scripts have two parts: 1) When they happen (events) 2) What to do (motion / looks
	Task 3 Planets	This task reinforces the on sprite clicked and introduces a new block - the "Say" block. Students	Your friend cannot remember all the planet names and wants something to help him

		program each planet say its name. The name is provided in the sprite list so that students have a guide for spelling. Students need to complete at least 4 plants to complete this project. If students want to, they can make the planets do fun things when they finish. For example, the planet could zoom around the screen, get bigger and then smaller, turn around in a circle, etc.	or her learn them again. In this project, you will use a new block, the "Say" block. For every planet, when the user clicks that planet, program the sprite to say its name. Don't worry if you have forgotten the names. The labels of the sprites tell the names. Look at the sun to see how it's done!
	Task 4 Bonus Piano Part 1	In this optional project, students program a piano where the keys are sprites. Students write scripts so that when each key sprite is pressed, it plays its note.	You can make a piano with LaPlaya! There is a new block under "sound" to plays different notes. All of the keys are sprites. Program each key sprite to play the correct sound when it is pressed! The C key has a picture of the script for you. Hint: Match the sprite name to the key sound.

Suggestions for Students who Finish Early-

Common Student Errors -

Teaching Hints -

- Students may forget to start a script with an event block. To help them remember, tell them that a finished script looks like the top-half of a hamburger. The brown event blocks is the top bun, and the other blocks are the layers of the burger.
- Students can copy/paste scripts from one sprite to another (useful in the planets task!). To copy a script, drag the script over to the other sprite in the sprite list (bottom right corner).

7: Brainstorming Digital Stories

Teacher Lesson Plan

Lesson Rationale-

In this activity students will be going through the brainstorming process to generate ideas for a digital story and then narrow them down to find the best solution. This process teaches students that all ideas are good ones, and those that may sound crazy might be the best ones. This also makes students evaluate their own ideas and come up with a rationale to pick the best one. Students will also write a persuasive piece about this idea and how it solved the engineering problem and fits within the constraints.

Objectives-

Students will be able to generate multiple ideas to solve the engineering problem presented to them. Also, they will be able to use reasoning to narrow down their ideas and ultimately pick the best one. Students will be able to create a piece of persuasive writing about their best idea to solve the engineering problem and address constraints.

Standards Emphasized-

Common Core State Standards:
ELA-LITERACY.W.4.1- Write opinion pieces on topics or texts, supporting a point of view with reasons and information. Introduce a topic or text clearly, state an opinion, and create an organizational structure in which related ideas are grouped to support the writer's purpose. Provide reasons that are supported by facts and details. Link opinion and reasons using words and phrases (e.g., *for instance*, *in order to*, *in addition*). Provide a concluding statement or section related to the opinion presented.
ELA-LITERACY.W.4.6- With some guidance and support from adults, use technology, including the Internet, to produce and publish writing as well as to interact and collaborate with others; demonstrate sufficient command of keyboarding skills to type a minimum of one page in a single sitting.

Materials-

Teacher	Student
"Engineering Design Thinking" worksheet (optional) Computers for students (optional)	"Brainstorming" worksheet "Persuasive Writing" worksheet Pieces of paper & pen/pencil Design Notebook (optional) Computer (optional)

Learning Tasks -

(Total Approx. Time: 45-60 minutes)

Approx. Time	Learning Tasks	Purpose/Instructional Strategies
3-5	Introduce the Brainstorming	Introducing the vocabulary and the

min.	part of the Engineering Design Process (optional to refer back to the Design Thinking worksheet visual) and go over the definition on the "Brainstorming" worksheet and the parts of the brainstorming process (which is iterative).	contexts that brainstorming is used in will help solidify what the process is to students and going over the parts of the process before students actually engage in it themselves should give them an idea of what is expected before beginning.
5-10 min.	Discuss guidelines for step 1 of the brainstorming process; Generate as many ideas as you can as quickly as you can. Emphasize that there are no bad ideas and that students can write or draw ideas on their paper, but they don't need to be pretty or descriptive at this point. You may want to time students and challenge them to come up with the most ideas at the end of that time. We suggest about 5 minutes for this 1st round of idea generation.	In this step it is important that students understand that their ideas, writing, and drawings of those ideas do not need to be too descriptive or perfect. It is important that students come up with many ideas for their digital stories so they have a larger pool to chose from when picking their final, best idea to create their program on. Students can be messy when writing and drawing their ideas on their paper, what it important is that they know what the ideas are so they can evaluate them in the next step of the process.
3-5 min.	Have students evaluate their ideas and write 3 to 5 of their best ones on the "Brainstorming" worksheet.	By doing this students will need to take into consideration the engineering problem and constraints when picking their best ideas.
3-5 min.	As a class pick an idea that was generated (use this as an example) and make that idea better and more elaborate to demonstrate step 3 of the brainstorming process; building on ideas.	Building and elaborating on their ideas may be difficult for students so going through this process as a whole class may help them understand how to do this and will help them when they do step 3 of the brainstorming process on their own.
5-10 min.	Have students build and elaborate on their top 3-5 ideas (from step 2) on a new sheet of paper. You may want to walk around and help students with this process if it appears that they are having difficulty.	This process helps students build on ideas and think about how they can make ideas better. This skill is important and can be applied to other people ideas as well. You may want to discuss that this is what is done all the time; people build on other's ideas and make them better (ex. The iPhone used others' ideas about touch screens and combining internet/phone capabilities and put them together in a new, better way).

3-5 min.	Have students go through and evaluate their ideas and pick the single best one. Students will then write this down on their "Brainstorming" worksheet.	Once again, students are having to take into consideration the engineering problem and constraints when evaluating their own ideas and narrowing them down to the best single idea they generated. This will be the idea that they will create their digital story on using LaPlaya.
5-10 min.	Have students get out their "Persuasive Writing" worksheet and go over what persuasive writing is, what the parts of a persuasive piece are (introduction, body text, conclusion) and explain that they will need to persuade someone else that their idea for their digital story is a good one that solves the engineering problem and fits within the constraints.	Having students learn about and write a persuasive piece is a big part of the common core standards for writing and is important in the engineering design process because it is critical that you are able to persuade others that your ideas are valuable. This skill is used in many fields and in everyday life (ex. If you want to go to Disneyland you need to persuade your parents to go).
5-10 min.	Students will complete the persuasive writing prompts on their "Persuasive Writing" worksheet before creating a finished persuasive piece online. It is optional to have students complete their writing using paper/pencil.	Persuasive writing is an essential skill for students to practice and learn. If students complete their writing on the computer they will also be practicing valuable typing skills and bet getting more comfortable using technology. You may be extended this by having students "pitch" their ideas to the class after they complete their persuasive writing.

Name/Number: ______________________________

Brainstorming

What is **brainstorming**?	A technique to generate IDEAS and come up with CREATIVE solutions to problems. It started in advertising and is now used for business, research, writing, design, and much more.	
What are the parts of **brainstorming**?	**Step 1:**	DRAW or WRITE as many ideas as you can think of as quickly as you can. *They don't have to be pretty or detailed, as long as you know what it means.*
	Step 2:	Pick 3 – 5 of the BEST ideas you came up with. *Write those at the top of a new page.*
	Step 3:	Use the ideas you picked in **Step 2** to come up with more ideas. Make them BETTER and more detailed.
	Step 4:	Pick the BEST idea you came up with to create your project on.

The process of brainstorming:

Step 1: Think of as many ideas as you can for your digital story. Keep the project guidelines in mind when you generate ideas.
Write on a blank sheet of paper.

Step 2: WRITE the best ideas you came up with on the lines below:

____

____

____

____

____

Step 3: *How can you make those ideas better?*
Can you combine any of the ideas?
Can you add to an idea to make it more interesting?

Step 4: What is the BEST idea you generated? *This will be your project.*

____

Name/Number: ______________________

Persuasive Writing

What is **persuasive** writing?	Writing about your OPINION on something and supporting your point of view with EVIDENCE (information, facts, reasons).	
What are the parts of a **persuasive** piece?	**Introduction:**	Introduce the topic and state an opinion.
	Body Text:	Group related ideas together and support your opinion with facts and details. Use phrases like "for instance", "in order to", or "in addition".
	Conclusion:	Summarize your argument and state why the evidence you provided supported your opinion.

Engineers, Scientists, and many others have to PERSUADE people that their idea or opinion is good and should be used to create a product, solve a problem, or make decisions. To do this they must make a good argument with EVIDENCE that will convince people to share their view.

Writing a persuasive piece about your digital story:

What was the idea you picked to use for your digital story? ______________________

____

Why is this a good idea for this project? *(1 sentence only)* ______________________

____

What is your evidence that this is a good project idea? *(How does it fit what the problem and constraints were? Will you be able to program it using LaPlaya? What else makes it good?)* ______________________

____

____

____

____

Summarize your whole argument in 1 sentence. ______________________

Now you have some ideas to start WRITING your **persuasive** piece online!

8: Storyboarding Digital Stories

Teacher Lesson Plan

Lesson Rationale-

In this lesson students will be elaborating on the idea for their digital story and creating a storyboard that will not only help them determine the structure of their story (including an introduction, memorable moments, and a conclusion), but will also help them communicate their planned story with others. Students will also learn the basics of sketching and communicating ideas efficiently by finding the balance between providing detail without wasting too much time.

Objectives-

Students will be able to create a basic storyboard of their proposed digital story that includes basic sketches of main events and short written descriptions. Students will also learn that sketching does not include excessive detail and should not take too much time.

Standards Emphasized-

Common Core State Standards:
CCSS: ELA-LITERACY.W 4.3: Write narratives to develop real or imagined experiences or events using effective technique, descriptive details, and clear event sequences.

Materials-

Teacher	Student
Computer and projector to show example(s) of storyboards (optional)	"Storyboards" worksheet "Sketching Basics" worksheet Storyboard Template(s) Pen/pencil Markers or colored pencils (optional) Design Notebook (optional)

Learning Tasks -

(Total Approx. Time: 45-60 minutes)

Approx. Time	Learning Tasks	Purpose/Instructional Strategies
3-5 min.	Briefly go over what storyboards are (using the "storyboarding worksheet) and how they are used in comic books, movies, etc. (optional to show a storyboard students may be familiar with- can find some examples online).	It is important to go over the new terminology with students and providing an example gives them a familiar context to connect this new thing they are learning about to. If you show an example of a storyboard you may want to find one for a children's movie that students will most likely

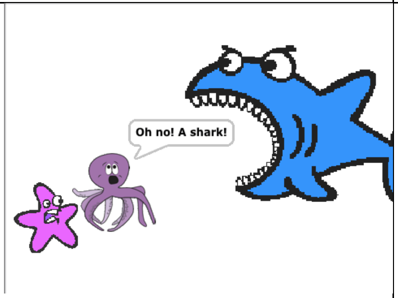
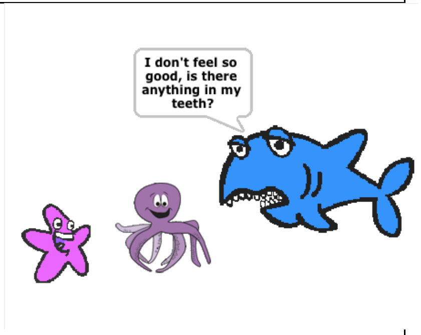
		have seen before (Toy story,
3-5 min.	Discuss with students the main parts of a story (introduction, memorable moments, conclusion) on "Storyboards" worksheet and how to relate that to your digital story's storyboard.	Emphasize how the different parts are essential to creating a good story that an audience will follow and enjoy. This process is also important in programming where you need to think about introducing a new user to your program, making it memorable throughout, and understanding how to wrap things up when the user is finished with the program.
3-5 min.	Go over the different parts of a storyboard with students and use to example storyboard provided for reference	The example storyboard using screenshots from LaPlaya gives students a good context for how their storyboard should look.
5-10 min.	Go over the basics of sketching using the "Sketching Basics" worksheet and emphasize to students that detailed drawings are not necessary, it is more important to get your ideas across efficiently.	The worksheet provides examples of different types of things that students may wish to sketch in their storyboards (people, things, emotions, actions). Making sure that students know that their sketches don't need to be perfect will help them understand that idea sketches should be less detailed as opposed to technical drawings in engineering, which need more detail. This should also make sure that students don't waste too much time on this part of the design process.
3-5 min.	Have students do a basic sketch on their worksheet that includes a person doing something and another object and that shows an emotion.	By having students practicing basic sketching with a generic prompt like this students will have an idea of how detailed they need to be in their storyboard sketches and will have practice showing actions and emotions in their sketching.
15-30 min.	Have students complete a storyboard for their own digital story. Their story will be about the best idea they came up with and wrote the persuasive piece on. After 5-10 minutes students should be moving on from one frame of their storyboard to the next (ex. from the introduction to the first memorable moment).	Creating a storyboard for their digital story is a huge part of the "design" portion of the Engineering Design Process and will be critical in later programming lessons when students learn about scene changes. You should monitor student progress throughout this storyboarding process to ensure that the students don't spend too much time on one part of their storyboard and end up not having time to finish. Students will be using these storyboards to create flowcharts later.

Name/Number: ______________________

Storyboards

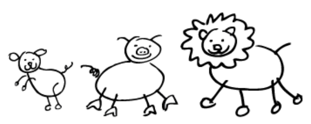
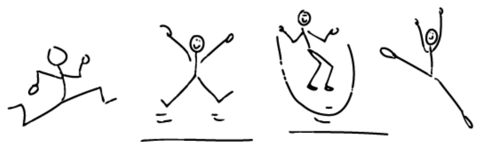
What are **storyboards**?	A series of sketches with written descriptions that represent the important parts of a planned program. They are often used for movies, television, and even comics.	
What are the parts of a **storyboard**?	**Introduction:**	Set the scene for your story. Introduce your characters and establish the setting.
	Memorable Moments:	Something important that happens to your character (they go somewhere, do something, or meet someone). *You can have more than one memorable moment.*
	Conclusion:	Resolve any problems, conflicts, or events that took place in your story. May include a message for users (ex. "the moral of the story is…").

A storyboard has 2 main parts, the SKETCH of the main events in the story and the written DESCRIPTION of what is happening.
See the example below before creating a storyboard for your own story.

Introduction	Memorable Moment #1	Conclusion
		
Alga the Octopus and Marty the Starfish are friends with a nice life in the ocean.	A big shark shows up and scares the two friends with his mouth wide open.	The shark has a toothache and just wanted their help; he didn't want to eat them.

Name/Number: ______________________

Sketching Basics

What are **sketches**?	Quick drawings that are designed to show others what you mean without wasting time on too much detail. *(At this phase in the design process too much detail may be distracting).* They can include some writing as well.
What are the <u>parts</u> of a **sketch**?	**People:** **Animals:**  **Actions:**  **Objects:**

Do a simple SKETCH of either a person or an animal doing an ACTION. Include at least one OBJECT also→	

Figure adapted from: Greenberg, S., Carpendale, S., Marquardt, N., Buxton, B. Sketching User Experiences: The Workbook. Morgan Kaufmann, 2012.

Name/Number: ____________________________ Page # ____ of ____

Storyboard for the story about ____.

Write the part of the story (Introduction, Memorable Moment #__, Conclusion) that each part is describing in the box above it-

______________________________	______________________________	______________________________

Description: ____________________
____
____
____
____
____

Description: ____________________
____
____
____
____
____

Description: ____________________
____
____
____
____
____

9: Event-Driven Programming: When Key Pressed

Teacher Lesson Plan

Lesson Rationale-

In this lesson, students continue creating programs that run on events triggered by a user. In the last lesson, students created scripts that ran when a user clicked on a sprite (using the event block “On Sprite Clicked”). Here, students create scripts with a new event block, “When Key Pressed.” Keys are buttons located on the keyboard such as numbers, letters, or arrow keys. Students will complete multiple tasks by where they need to program a script to run on different computer keys.

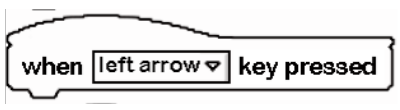
Objectives-

Students will be able to:
- Change the absolute direction of a sprite (point up, down, left, or right).
- Write scripts to run when the four arrow keys are pressed.
- Correlate sounds to play on different keys.
- Engage a user in their programs.
- Create scripts that run when a user interacts with the keyboard.

Vocabulary-

User: The person playing the game, running the program, etc.
Control Blocks: Blocks that determine when something should happen.
Event: Something that the user does (click sprite, press button, etc.)
Key: Buttons located on a computer keyboard such numbers, letters, or arrow keys.
Interactive: A program that responds to things the user does.

New Blocks -

Block	**Block Name**	**Block Description**
 when left arrow key pressed	When [Key] Pressed	Event block that runs a script when a user presses keys on the keyboard.
	When [Reset Button] Pressed	Event block that runs a script with the green reset button is pressed. Typically the start of an initialization script.
	Point towards [ ]	Motion block that orients a sprite in the direction of another sprite.
	Play note [C] for __ beats	Sound block that creates a sound for a variable among of time. Sounds are pre-programmed to

		certain notes.

Standards Emphasized-

Materials-

Teacher	Student

*Learning Tasks (*Total Time: 45 minutes)

Approx. Time	**Learning Tasks**	**Purpose**	**Student Directions**
? min.	*Task 1* Rocket	Use this task to demonstrate how to create a script with the "When [up arrow] is pressed" that moves a rocket ship upwards. Following the teacher demonstration, students complete the task independently on their computer.	Help the astronaut blast off into space to explore the universe! Program a script that, when you press the up arrow, makes the rocket go up. A picture of the script is given to you in the rocket. Make your own copy of that script. Don't forget to click the "get ready" button and then the green flag to start!
	Task 2 California Geography	Students program a car sprite to move in four directions (up, down, left, and right) on the four arrow keys. Then they move the car with the arrow keys to travel to multiple California cities.	Program the car sprite to move in four directions (up, down, left and right) on the four the arrow keys. Then, use the arrow keys to move the car sprite to the different California Cities! Hint: You should have four scripts, one for each arrow key!
	Task 3 Catching Fruit	Students program a basket sprite to move left on the left arrow key and right on the right arrow key to catch apples as they fall. The apples fall when the student clicks on them.	Program the basket sprite to move left and right on the left and right arrow keys. Click on each apple and catch them in the basket as they fall!
	Task 4 Bonus	Students turn their keyboard into a piano by programming different	Turn your keyboard into a piano! For each note, write a

	Piano, Part 2	keys for different sounds. The selected keys match proper hand position for typing.	script that will make it play the correct sound. Press the Green Flag to start! Hint: Use the script on Key 1 as a guide.

Suggestions for Students who Finish Early-

Common Student Errors -

Teaching Hints -

10: Thinking About the User

Teacher Lesson Plan

Lesson Rationale-

The **user** is what we call the person who plays with the software. Students have been "users" of software throughout their lives; they are currently users of LaPlaya in this module! Computer scientists often want to engage users in their programs through events such as clicking the mouse or pressing the keyboard (like the last activity!). Sometimes, in order to make software easy to use, computer scientists also must run special programs in the background. This is true with LaPlaya! In all the activities for this curriculum, our computer scientists run invisible scripts to help students finish the tasks. In this activity, students begin learning the difference between being the **user** of software to being the **designer** of software. In these tasks, students will analyze finished LaPlaya tasks to determine what actions may be hidden and discuss how they can engage the user in their own programs.

Objectives-

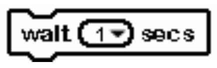
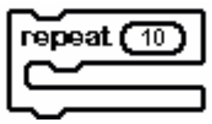
Students will be able to:

- Differentiate between a user and a designer.
- Identify activities when they acted as user and when they acted as a designer.
- Consider what types of scripts are hidden in LaPlaya.
- Discuss the benefits and consequences of hiding scripts in LaPlaya.

Vocabulary-

User: The person playing the game, running the program, etc.
Designer: Computer scientist or programmer than makes software.

New Blocks -

Block	**Block Name**	**Block Description**
	Place at [ ]	Motion block that immediately places sprite at a particular location.
	Wait [ ] secs	Control block to adds a variable amount of time between actions.
	Repeat __	Control block that repeats scripts in bracket. Repetitions by vary by any amount.

Standards Emphasized-

Materials-

<u>Teacher</u>	<u>Student</u>

*Learning Tasks (*Total Time: 45 minutes)

Approx. Time	**Learning Tasks**	**Purpose**	**Student Directions**
? min.	*Task 1* Finding Hidden Scripts	In this task, students look through a completed project like last activity. They are supposed to play it and identify what hidden things are happening for which they do not see scripts. The solution should be a list in this format: Sprite X does What when This Happens Bush says "" when bear hits it (if Bear hasn't hit bush): Honey pot says "Good Job" when bear hits it Honey pot tips over and bear goes inside when bear hits it	Your job in this task is to find all the hidden scripts! Some of the sprites do things, and you are not able to see the scripts. This is because we, the designers, do not want to confuse you, the user, with complex scripts. You need to produce a very short list that looks like this: Sprite ______ does _____ when _________ happens
	Task 2 Analyzing the User Interface	In this task, students are asked to figure out all of the things that cause actions in the program. They should check every key on the keyboard as well as click everywhere on the screen. Their solution list should look something like this: Key X causes sprite Y to do Something If I click sprite Z, then sprite L does Something	Here is a tortoise and the hare project. The tortoise and the hare are supposed to race but designer did not make it very clear. Your job is to figure how a user causes actions to happen in the program. Make sure you check all of the keys on the keyboard and click things on the screen. Your solution should look something like this: When I press key _____, sprite ______ does _______ When I click on sprite ________, sprite ______ does ________
	Task 3 Designing the User Interface	This task starts with the same file as "Analyzing the User Interface." This time, students will fix the interface. We want them to choose an interface that makes sense. For example, clicking the two animals	Your job now is to fix the project you just explored. It was not a good interface to have one animal start when it was clicked but the other animal start when a key was

		should do something similar (like make them both go). Or they could both go on the green flag (even better!). Likewise, h and b should do similar things (perhaps those would change the colors). Whatever they choose, they should be consistent in having the same type of command do the same thing.	pressed. Here is what you need to do: Make the animals start their race on the green flag. Make them change get bigger then smaller when you click on them. Don't do anything when you press keys.

Suggestions for Students who Finish Early-

Common Student Errors -

Teaching Hints -

11: Flow Charts

Teacher Lesson Plan

Lesson Rationale-

This lesson introduces flowcharts to students, which are used by computer programmer (among others) to help plan out their program and visually display all of the different parts of the program. This lesson bridges the conceptual divide between the storyboards students created previously and the sophisticated flowcharts that students will be creating once they start programming and revising their digital stories.

Objectives-

Students will be able to identity the different parts of a flowchart and understand what each means. They will also interpret a basic flowchart them create their own using their storyboards as a guide.

Standards Emphasized-

Materials-

Teacher	Student
	"Flowcharts" worksheet Flowchart template Storyboard template Design Notebook (optional)

Learning Tasks -

(Total Approx. Time: 30-45 minutes)

Approx. Time	**Learning Tasks**	**Purpose/Instructional Strategies**
3-5 min.	Go over what flowcharts are and how they are used in programming. Emphasize that all software developers (programmers) go through this process to layout what the parts of their program are and how everything works together before even getting on the computer.	Going over what flowcharts are and how they are used in computer science is important for students to understand the importance of this process in the design cycle before getting on a computer to program their digital stories.
5-10 min.	Go through what the different parts of a flowchart are (lines, dotted lines, rectangles, ovals,	It is important to spend some time going though the different parts of a flowchart with students because they can be

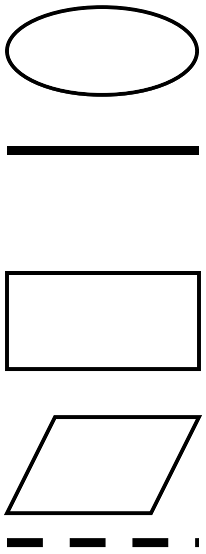
	and parallelograms) and go through the example flowchart given on the worksheet.	confusing and might not make sense out of context (not on the computer). As students complete the questions related to the example flowchart on their worksheets you may want to help them answer the questions given.
3-5 min.	Guide students through creating the basic flowchart on the left of their Flowchart Template worksheet. Students should use their storyboards (the description of each frame) as a guideline to help them fill out the three boxes provided in this flowchart.	This step of creating a basic flowchart using their storyboard as a guide is a way to let students transition more seamlessly from thinking of their digital story as a visual story to thinking of it more like a computer program.
5-10 min.	Allow students to create a fuller version of their flowchart in the blank one on the right of the Flowchart Template worksheet. Students should look at the different shapes that denote different parts of a program and think about how they can incorporate those into this new flowchart of their program.	Students should start thinking of their digital story more like a program and begin figuring out what kinds of commands will need to be used to start the main events that will occur in their program. You may want to check in with students throughout this process and relate the parts of the flowchart to the things they want to happen in their digital story.
2-3 min.	Discuss how students can add different parts (new shapes and lines and even new flowcharts for each sprite) to their own flowcharts for their digital story when they learn more programming skills and elaborate on their program.	It is good to let students know that the flowchart they will be creating now is not their final one and that they will be learning more programming skills and adding to their digital story and flowchart throughout the design process.

Name/Number: ______________________

Flowcharts

What are flowcharts? A way to organize and show the "flow" of your program. Programmers use flowcharts to work out the steps in a program they are designing before coding anything on the computer. You will create one for each sprite later on.

What are the parts of a flowchart?



The START or END of a program.

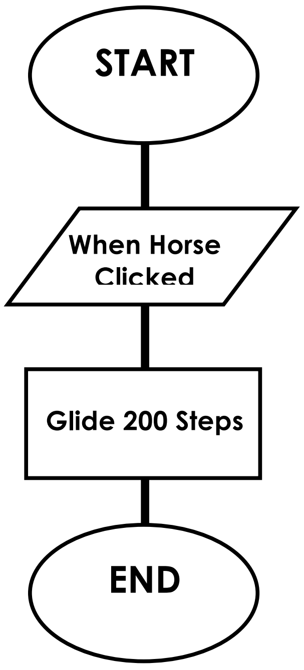
Shows the RELATIONSHIP between the parts of a program.

A PROCESS that will be performed. (Ex. "Glide 50 Steps")

An INPUT that starts a process. (Ex. "When Sprite Clicked")

The RELATIONSHIP between different sprites (Ex. "When Sun Clicked" - - "Boy says It's hot!")

Interpreting a flowchart:



1. What does this program do?

____

____

2. What is the INPUT that starts the process "Glide 200 steps"?

____

____

3. What kind of a program do you think this flowchart is for?

____

____

4. How could you use this to help write a flowchart about your digital story?

____

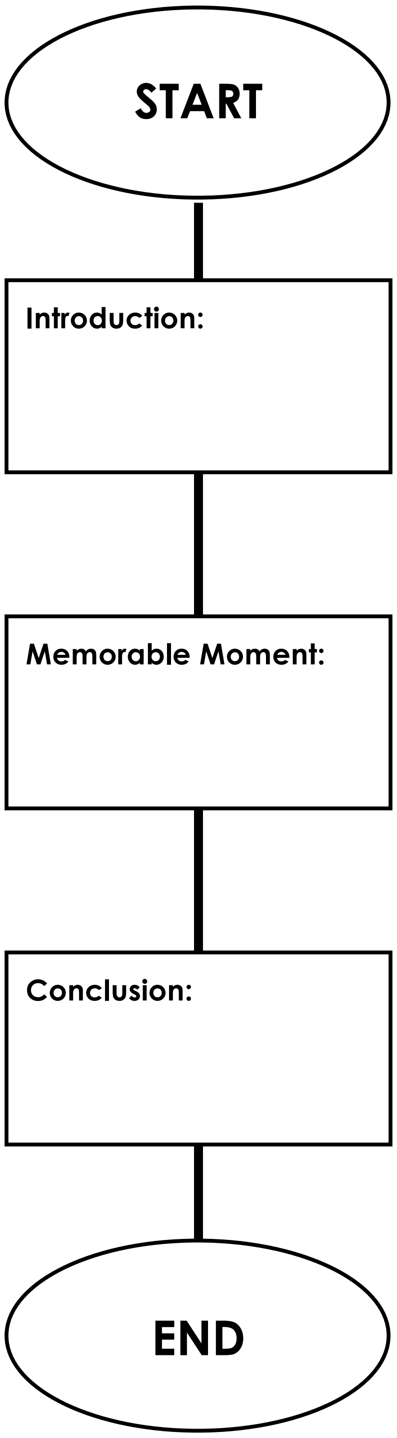
____

Name/Number: ______________________________ Page #___ of ___

Flowchart for the story about ____.

Step 1: Fill out the Basic Flowchart

Step 2: Add to it!
(Try using other shapes too)



Step 3: Keep adding to your flowchart as you learn new programming skills.
(Use a new piece of paper)

12: The Application Cycle

Teacher Lesson Plan

Lesson Rationale-

This lesson elaborates on the application cycle (program, test, evaluate, redesign, etc.) of the Engineering Design process and gives students a framework for how to go about developing their digital story program in LaPlaya and how to keep track of their work and elaborate on their program flowchart(s).

Objectives-

Students will be able to understand the iterative process of the application cycle within the Engineering Design Process and will be able to distinguish between each part of the cycle in the context of creating a digital story using LaPlaya (Program, test and evaluate, and redesign). Students will also be able to record these aspects of the design process and understand that they will need to elaborate on their flowchart as they learn more programming skills and create a more complex digital story.

Standards Emphasized-

CSTA Computer Science Standards:
L1:6:CL- Identify ways that teamwork and collaboration can support problem solving and innovation.

Materials-

Teacher	Student
"Engineering Design Thinking" worksheet with projector (optional)	"The Application Cycle" worksheet "Engineering Design Thinking" Worksheet (optional) Piece(s) of paper & pen/pencil for notes and new flowchart iterations Design Notebook (optional)

Learning Tasks and Instructional Strategies-

(Total Approx. Time: 10-15 minutes)

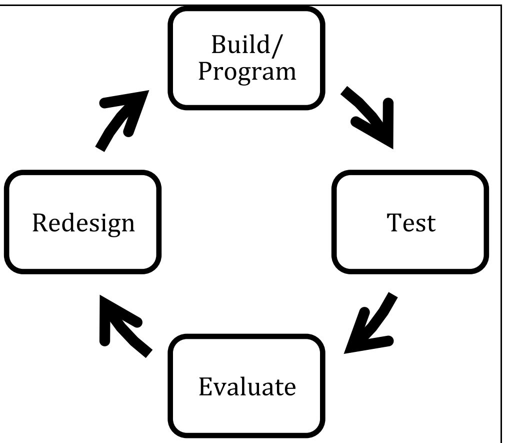
Approx. Time	Learning Tasks	Purpose/Instructional Strategies
3-5 min.	Go over with the class what the application cycle is (related to scientific process for science, design process for engineering & program development process for computer science), where it fits with the other step	Discussing how this application cycle in engineering design is similar to the scientific process as well as the program development process in computer science is important to allow students to see the relationships between these different fields and the importance of this process overall.

	in the Engineering Design process (may want to reference the "Engineering Design Thinking" worksheet), and what each part of it is (Program, Test, Evaluate, and Redesign).	Referencing where the application cycle is in the Engineering Design process as a whole is good for allowing students to visualize which part of the process they are currently engaging in. By going through what each part of the design process is in more detail it should help students distinguish between each part and will help them while ultimately programming their digital story.
3-5 min.	Go over how from this point forward students will be using LaPlaya to create their digital story. There is a button titled "Design Thinking Project" that students can click on any time they want to work on their digital story. Let students know the logistics of when they can and will be working on this program during class time (there will be a few dedicated sessions to work on this program at the end of the module and you may consider allowing students to work on this project if they finish the computer-based lesson early).	Giving students the logistics about when and how students will be working on this program is good so students know what they have to do, when they can do it, and what your expectations are for them. Make sure to tell students that they should be using their storyboard and flowcharts to help them in the initial stages of creating their program. They should first decide on the basics including which sprites they will be using for their story and what background(s) they will need and then they can begin adding blocks to program these elements. You should also remind students that they will need to be thinking about the user and their experience when using their program.
3-5 min.	Go through the bottom portion of "The Application Cycle" worksheet with students, which gives prompts for the kinds of things that they should be recording throughout this design process and when creating their program. They can record these things as much as you wish (after every work session, at the end of the module, etc.). Make sure that they understand that they should be going back and creating more complex flowcharts for their programs and that they may want to create a new flowchart for each sprite (or the	Recording problems you encounter, changes you make, and possible solutions to those problems is a huge part of not just engineering, but also computer science and the other sciences. Often times these things are recorded in a science notebook or a design notebook and these are used so you can go back and look at the changes you have made and the process you have gone through, which is good practice in case a problem arises (ex. If a program you write has a glitch you can trace which part of it may be messing up the whole thing). It is also important that students clearly understand that they should be adding to the flowcharts for their program and making them more complex and encompassing of everything they have included.

	background) that they are using.	
During work sessions	Throughout the work sessions that students will have to program their digital story you may consider having students think about the following things: • Do I have all the sprites I need? • Do they all do what I want them to do? • Is my background how I want? • Does everything do what it is supposed to when clicked, when a key is pressed, when something happens? • Is there a way I can make my project better, more fun, more user friendly? • Can I use fewer blocks to do the same thing? • What else do I need to add to, take away from, change on my flowchart?	These questions should help guide students when programming their digital stories by asking that they think more deeply about the program as a whole, each specific aspect of the program, as well as the different people who may be using their program. This should help them evaluate their own work and create a high quality program.

Name/Number: ____________________

The Application Cycle

What is the **application cycle**?	The part of the Engineering Design Process where you are actually creating something (by building, making, or programming it). You also test your design and see what could be improved so you can redesign and create a better program.	
What are the parts of the **application cycle**?	**Build/ Program:**	You are creating something based on the design you created. This is where you move to the computer to actually program your digital story.
	Test:	Try out your program. Make sure to think about how others may use your program also.
	Evaluate:	What did and didn't work?
	Redesign:	What should you change about your program?

From this point on you will be working on the computer using LaPlaya to create your digital story! *Keep coming back to your work to keep track of the **constraints** you must follow and draw new, better, and more detailed **flow charts**.*

Keep track of the changes you make throughout the application cycle:

In your Design Notebook or on a separate sheet of paper write the following information down when you make a change to your project:
1. What change(s) did you make to your digital story? *(How is it different from your previous version?)*   2. Why did you make those changes? *(Did it make your project better or worse than you had originally hoped?)*   3. How did you change your flowchart to show the changes you made? *(Did you create new commands/shapes? Did you add more detail? Did you delete part of it? Did you add a new flowchart for a different sprite?)*   4. Is there any way to improve your project? *(Could you make it better? How? What would you add/take away?)*

13: Event-Driven Programming: When Other Sprite Clicked

Teacher Lesson Plan

Lesson Rationale-

In this lesson, students continue creating programs that run on events triggered by a user. In the last two lessons, students programmed the action of a sprite to correspond with a key press or when that same sprite was clicked on the stage. Here, we add one more level of complexity by coordinating action between sprites. In some cases, we may want one sprite to do something based on the action of another sprite. In these tasks, students will program the action of a sprite to occur when a **second sprite** is clicked using the event block, "When Other Sprite Clicked."

Objectives-

Students will be able to:

- Create scripts using the event block, "When Other Sprite Clicked."
- Practice keeping track of multiple sprites doing multiple actions.
- Engage a user in their programs.
- Make a sprite respond to a user clicking a different sprite

Vocabulary-

User: The person playing the game, running the program, etc.
Control Blocks: Blocks that determine when something should happen.
Event: Something that the user does (click sprite, press button, etc.).

New Blocks -

Block	Block Name	Block Description
	When [Other Sprite] Clicked	Event block that runs a script when a second sprite is clicked.

Standards Emphasized-

Materials-

Teacher	Student

*Learning Tasks (*Total Time: 45 minutes)

Approx. Time	Learning Tasks	Purpose	Student Directions

? min.	*Task 1* Introduction: Intersection 1	Use this task to demonstrate how to create a script with "When Other Sprite Clicked" event block. Create a script that moves the car to the parking spot. Following the teacher demonstration, allow students to complete the task independently on their computer.	Program the Car sprite to move to the Parking Spot when the Traffic Light sprite is clicked.
	Task 2 Intersections 2	Students program two car sprites to move to their corresponding parking spaces when the corresponding traffic light is clicked. Note: The color of each car matches the house and parking spot.	Program the White car to move to its parking space when the White traffic light is clicked. Program the Blue car to move to its parking space when the Blue traffic light is clicked. Hint: The color of the parking space matches the color of the car!
	Task 3 Intersections 3	Students program two car sprites to move to their parking spaces when the corresponding traffic light is clicked. Each car must turn turn 90 degrees to the right. Note: The color of each car matches the house and parking spot.	Program the White car to move to its parking space when the White traffic light is clicked. Program the Blue car to move to its parking space when the Blue traffic light is clicked. Hint: Both cars need to turn at the intersection!
	Task 4 Piñata Candy Falling	Students program three candy sprites to move to the ground when the piñata sprite is clicked.	Program the three pieces of candy to glide downwards when you click the piñata. Bonus! Try to make the candies fall so you can see all three of them!
	Task 5 Shaking the Tree	Students program four apple sprites to glide downwards when the tree sprite is clicked.	It's such a windy day! The wind is shaking the fruit from the tree. When the tree sprite is clicked, the tree shakes. Program each apple sprite to glade downwards when the Tree sprite is clicked. Hint: Each apple may need to fall a different distance to reach the ground.

Suggestions for Students who Finish Early-

Common Student Errors -

Teaching Hints -

- Students can copy/paste scripts from one sprite to another (useful in the planets task!). To copy a script, drag the script over to the other sprite in the sprite list (bottom right corner).

14: Initializing Programs

Teacher Lesson Plan

Lesson Rationale-

Computer scientists often want their programs to run in the exact same way every time. In order to do this, any items that changed over the course of the program need to be reset to the starting conditions before the program runs again. This is initialization. In LaPlaya, sprites can move, change color, or point in new directions during a program. In order to initialize, students need to reset these attributes to their starting conditions without a user being able to notice. If a program is initialized correctly, the user will not see how the sprites changed, only that they did change. Sprites will instantly appear in their correct location with the starting color and orientation! In these tasks, students begin to understand the importance of initializing in computer science by playing games that were not reset correctly. Then, students will learn how to initialize in LaPlaya by creating special scripts with the event block, "When Get Ready Button Pressed".

Objectives-

Students will be able to:

- Identify attributes of a sprite that need to reset when a program runs such as position, size, color and orientation.
- Create a program that moves a sprite immediately to the starting position with the Go To blocks.
- Create initialization scripts using the event block, "On Reset Button clicked"

Vocabulary-

Initialize: To reset any changes in a program to the starting values.
Starting Conditions: The many ways a sprite should begin when a program is reset (color, location, orientation, etc.).
Orientation: The direction something is pointed

New Blocks -

Block	Block Name	Block Description
	When Get Ready Button Clicked	Event block that runs a script when the Get Ready button is pressed. Used to create an initialization script.

Standards Emphasized-

Materials-

Teacher	Student

I	

*Learning Tasks (*Total Time: 45 minutes)

Approx. Time	Learning Tasks	Purpose	Student Directions
? min.	*Task 1* Fired Up	Teacher leads an in-classroom activity to reinforce why initializing is important in daily life (*Initialization Fired Up*).	
	Task 2 Demo: Watch the Tortoise and the Hare	Students watch a video in LaPlaya of Tortoise and the Hare. Video begins after pressing the "Get Ready" button followed by the "Green Flag" button.	Click the "Get Ready" button and then the "Green Flag" button. Watch the story of the Tortoise and the Hare.
	Task 3 Demo: Tortoise and the Hare Version 2	Students will learn the importance of initialization through a common childhood story. Part 2 involves a fully functioning story for the students to watch after pressing the "Ready Set" button followed by the "Go" button. However, when the student watches the story a second time everything goes wrong because nothing has been initialized.	Watch the story of The Tortoise and the Hare once. Then watch it one more time and explain the difference between the two.
	Task 4 Piñata Part 2	Students create initialization scripts to reset a piñata. They program each candy sprite to start in the piñata.	Oh, no! Someone forgot to start the candies in the piñata! The kids will be so disappointed! Program the "get ready" scripts for the candies so that they start in the piñata. The first candy has a picture of the script that you can see how to do it. Program the script in all three candies.
	Task 5 Animal Sprint: Horse	Students program the "get ready" initialization script that to make a horse sprite start in the proper place for a sprint. The position is marked by a sprite that is the outline of the horse.	Oh, no! The horse is not at the starting line for the race! Program the "get ready" initialization script that will make the horse start in the proper place (marked by a sprite that is the outline of the horse).
	Task 6	Students program the "get ready"	Oh, no! The cat is not ready

	Animal Sprint: Cat	initialization script that to make a cat sprite start in the proper place with the correct size for a sprint. The position and size are marked by a sprite that is the outline of the cat.	for the race! Program the "get ready" initialization script for the cat. Then run the program (press "get ready" button, then "green flag" button). What changed about the cat? Make sure your "get ready" initialization script initializes everything about the cat.
	Task 7 Animal Sprint: Rooster	Students program the "get ready" initialization script that to make a rooster sprite start in the proper place with the correct orientation for a sprint. The rooster needs to point left to start. The position and orientation are marked by a sprite that is the outline of the rooster. Students may need to run the program more than once to realize they need to initialize the orientation.	Oh, no, what is that rooster doing! It needs to get ready for the race! Program the "get ready" initialization script for the rooster. Then run the program and watch the rooster. Make sure your get ready script initialized everything about the rooster.

Suggestions for Students who Finish Early-

Common Student Errors -

Teaching Hints -

Initialization Fired Up

Teacher Lesson Plan

Lesson Rationale-

Computer scientists often want their programs to run in the exact same way every time. In order to do this, items need to be set up properly when the program begins.
The process of properly setting up conditions for an activity is very common in daily life. For example, whenever you play a game, you must set it up properly to begin. Before a soccer game begins, all of the players get on their own side of the field, and the ball is placed in the middle. When you play a board game like Monopoly, each player receives the same amount of money to begin. Players do not hold on to hotels or property game to game!

Initialization is equally important in programming. In traditional programming, initialization involves setting starting values of variables. Initialization in LaPlaya differs from traditional programming in two ways. First, initialization involves placing sprites in the right location, size and orientation. Second, LaPlaya holds onto values between when you close it and open it again. Only items that changed during the course of the program need to be initialized. Thus, it's in LaPlaya to relate initialization to "resetting" instead of "setting up." Please focus on the wording "setting up."

In this activity, the teacher introduces students to importance of initializing. The teacher intentionally "forgets" to place items in their starting position for an activity. You can take this as far as you want – doing this in the activity we suggest or throughout the day. We have further suggestions at the end if you want to do it for the entire day.

Objectives-

The purpose of this activity is to reinforce the idea of initialization – setting multiple conditions properly before restarting. There are many ways one needs to set something up, so we do encourage you to be creative and find additional ways to reinforce this concept.

Standards Emphasized-

Common Core State Standards:
CCSS. Math.Content.4.G.A.1 - Draw points, lines, line segments, rays, angles (right, acute, obtuse), and perpendicular and parallel lines. Identify these in two-dimensional figures.

Materials-

<u>Teacher</u>	<u>Student</u>
Several erasers, an outdoor play area with two parallel lines that are quite far apart	

Learning Tasks -

(Total Approx. Time: 15-30 minutes)

Approx. Time	Learning Tasks
5 min.	• Begin by dividing the students in two groups. Do not have them stand on the lines the first time. Place the erasers on the ground in the middle of one of the groups. • Explain the rules of the game – when you say go, everyone tries to get the erasers. The group who gets the most erasers wins. • If someone raises a point of fairness, move to the next step – do not run the game. • Run the game once. Presumably, the group in which you placed the erasers will win. Declare them the winners. • Someone should raise an issue with the fairness. If not, ask them whether it was fair.
10 min.	• Discuss with the students what was unfair • Identify the problem as not setting up the game properly. • Now, with the same group assignments, have one group line up on one side, and the other group line up parallel to them with a large gape in the middle. • Put the erasers on a third line equidistant from the students. • You are welcome to use this as a math lesson – parallel lines, equidistant. • Play two more times just for fun. ☺
8 min.	• Return to the classroom (either immediately or after recess or more play time) • Explain the concept of “setting up” or “initializing” • Invite the students to supply ideas as to other times when initialization is required for success. Below is a list we have compiled. For each one, have students tell you o What needs to be initialized? o What would happen if they forgot to initialize it?

Optional Learning Tasks -

(Total Approx. Time: 15-30 minutes) – these are things the teacher can “forget” to set up properly throughout the day.

Approx. Time	Learning Tasks
5 min.	When you write on the chalkboard, forget to erase it before using it again.
5 min.	Xerox a worksheet on top of another worksheet. You can do it once and then use that image to Xerox the rest.

15: Animating Sprites

Teacher Lesson Plan

Lesson Rationale-

Animations are important for telling a digital story. Often, digital writers want to do more than place images on screen, they want images to move. In LaPlaya, animations are created when images of a sprite change in small ways much like a flipbook. On each page of a flipbook, animators drew pictures that changed slightly from page to page. When these pages were turned rapidly, the image on the pictures seemed to move as well. And the faster someone turned the pages, the faster the images seemed to move! (See a YouTube flipbook of the 2014 World Cup as an example).

Unlike flipbooks though, computers can rotate through images so quickly that users will not see the animation. So, computer scientists write special programs for **timing**, code that pauses each image briefly.

In LaPlaya, animating works by rapidly moving through images of a sprite called **costumes**. And, timing is necessary to determine what kind of visual effects are created! When there are long waits between costumes, the animation seems slow; when the waits are short, the animation seems fast. In this activity, students will practice animating sprites in LaPlaya by programming a series of costume changes and wait blocks.

Objectives-

Students will be able to:

- Animate a sprite through costume changes.
- Practice implementing initialization scripts.
- Properly time an animation with waits blocks between costume changes.
- Create new costumes for a sprite.

Vocabulary-

Costume: A picture that represents a sprite or a version of the sprite.
Costume Change: The process of moving between pictures (costumes) of sprites in LaPlaya
Animation: Using multiple and successive drawings of a character to create the illusion of movement.
Timing: Choice of when something should occur.

New Blocks -

Block	Block Name	Block Description
	Switch to costume ___	Look block that changes the appearance of a sprite to another costume that user selects.

	Wait __ secs	Control block to adds a variable amount of time between actions.
	Next Costume	Optional: Look block that changes the appearance of a sprite to subsequent costume on the list.

Standards Emphasized-

Materials-

<u>Teacher</u>	<u>Student</u>

*Learning Tasks (*Total Time: 45 minutes)

Approx. Time	**Learning Tasks**	**Purpose**	**Student Directions**
? min.	*Task 1* Ballerina	Teacher demonstrates how to move between costumes for a single sprite with the example of a dancing ballerina. Students then complete the task independently on their computer.	If you just make the sprite glide, it does not look like it's really moving. Using costumes, we can make the sprites do many fun actions! Program two scripts for the ballerina. 1) Initialize the ballerina's position and starting costume. 2) Make the ballerina dance when she is clicked.
	Task 2 Pick a Dancer	Students create scripts to animate dancer sprites. They may choose from three dancers to program, and need to use at least three costume changes in the routine.	Now make a dance routine yourself! First pick a dancer to program. Then write a script for routine that includes 3 costume changes. Hint: Vary the wait blocks to make the dance slow or fast!
	Task 3 Dance Party!	Students create scripts to animate a dance party. They program each dancer sprite to animate when the disco ball is clicked.	Now, create a group dance! Program each dancer to animate when the disco ball sprite is clicked. Hint: Start with the break-dancer!
	Task 4	Through a tutorial, students learn how to create their own costumes	If you can't find a sprite you like, then you can draw your

	Draw Your Own	and animate a stick figure with those new costumes.	own! Click the "get ready" button, then the green flag button to go through a lesson on how to draw your own sprite costume. Click on each Next button on the stage to learn steps to drawing your own costumes for Stan the Stick Figure.

Suggestions for Students who Finish Early-

Common Student Errors -

Teaching Hints -

- Bring a flipbook to class to emphasize animation.
- Allow students free time during class to create their own flipbooks with paper and pencils.
- Encourage students to use different times in the wait blocks for their scripts. Not every costume change needs to the same timing – wait blocks can differ in time for each action!

16: Changing Scenes

Teacher Lesson Plan

Lesson Rationale-

In this project, students learn how to change scenes in LaPlaya. Often, in a program or game, computer scientists want different aspects of their projects visible at different times. In a game, the scene may change when the player travels to a different location or moves to a higher level. Or, in a digital story, a scene will change as characters travel to new places and meet new characters. In LaPlaya, scene changes require two things 1) Figuring out which sprites are visible, 2) Changing the stage background. In this activity, students practice moving between scenes by determining when the change is going to happen, what the new background will be, which sprites will be visible in each scene.

Objectives-

Students will be able to:

- Use show/hide blocks to determine which sprites are visible in each scene.
- Change the stage background.
- Change to new backgrounds with different events.

Vocabulary-

Scene: A set of sprites and backgrounds that serves as setting to a program.
Background: Representations of the stage and part of the scene.
Show: Make a sprite appear or reappear on the stage.
Hide: Make a sprite disappear from the stage.

New Blocks -

Block	**Block Name**	**Block Description**
	Hide	Look block that makes a sprite immediately disappear.
	Show	Look block that makes a sprite immediately appear or reappear.
	Switch to background __	Look block that changes the background of the stage to a selected background. Only visible when the stage is selected.
	Next background	Optional: Look block that changes the appearance of the stage to the following background on the list. Only visible when the stage is selected.

Standards Emphasized-

Materials-

<u>Teacher</u>	<u>Student</u>
Storyboarding worksheets	Pencil/Pen

*Learning Tasks (*Total Time: 45 minutes)

Approx. Time	**Learning Tasks**	**Purpose**	**Student Directions**
? min.	*Task 1* Plant Growing Demo	Teacher demonstrates the complete Plant Growing program, which contains three scenes. Click on the different number sprites to change scenes. Students then complete a storyboard of the three scenes by determining: 1) Visible sprites in each scene. 2) Background used in each scene. 3) Sprites who do something special when clicked. During the next tasks, students will incrementally write scripts for each of these changes.	Let's storyboard this program! Using the worksheet, draw the three scenes in this program. Be sure to draw: 1) Sprites you can see. 2) Where the sprites are in each picture. 3) If a sprite does something special. 4) The background.
	Task 2 Plant Growing Initialization	Students create initialization scripts for all sprites in Plant Growing. These are the starting conditions for each scene. Students need to initialize the location, costume, and visibility (hide/show) for each sprite. Then they need to initialize the starting background.	Look at the storyboard you made for the plant growing project. Write scripts to initialize all of the sprites and the background. Remember to initialize the costume, location, and whether it is visible. The code for the plant has been given to you.
	Task 3 Plant Growing Scene 2	Now, students program scripts to change from scene 1 to 2. Clicking script One and Two on the stage triggers the scene change. Students look at their storyboards to determine how each sprite and background changes in scene 1 and scene 2. They need to how/hide different sprites, pick correct costume for each sprite in	Write the scripts to change from Scene 1 to Scene 2. When the Two sprite is pressed, the scene should change to the scene 2. When the 1 sprite is pressed, the scene will change to to scene 1. Hint: Scripts for the plant have been started for you, and the

		the two scenes, and position the sprites in the correct place.	initialization scripts are already complete!
	Task 4 Plant Growing Scene 3	Students complete the program by creating scripts to change between Scene 2 and Scene 3.	Now you are adding scripts to change to Scene 3. Use everything you learned in the last two tasks and look at your storyboard!

Suggestions for Students who Finish Early-

Common Student Errors -

Teaching Hints -

- Use the storyboards from the earlier design-thinking activities to connect how scenes change.

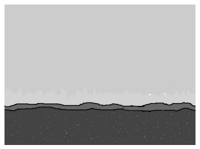
Name/Number: ___________________________

PLANT GROWING STORYBOARD

Create a drawing for each scene.

Scene 1	Scene 2	Scene 3
Description: ______________________	Description: ______________________	Description: ______________________

Circle which version of the sprite or background is correct for each scene.

		Scene 1	**Scene 2**	**Scene 3**
Stage		Scene 1 Scene 2 Scene 3	Scene 1 Scene 2 Scene 3	Scene 1 Scene 2 Scene 3
Plant		Show Hide Seed Sapling Flower	Show Hide Seed Sapling Flower	Show Hide Seed Sapling Flower
Sun		Show Hide ClickMe ShiningSun RegularSun	Show Hide ClickMe ShiningSun RegularSun	Show Hide ClickMe ShiningSun RegularSun
Cloud		Show Hide Raining Not Raining	Show Hide Raining Not Raining	Show Hide Raining Not Raining
Three		Show Hide Selected Normal	Show Hide Selected Normal	Show Hide Selected Normal
Two		Show Hide Selected Normal	Show Hide Selected Normal	Show Hide Selected Normal
One		Show Hide Selected Normal	Show Hid Selected Normal	Show Hide Selected Normal

17: Sharing Your Work

Teacher Lesson Plan

Lesson Rationale-

After students have spent the allotted class time working on their digital stories they will finish them up and get them ready to share with others. This lesson underscores the importance of sharing your work in not just engineering, but also the sciences, programming, design, etc. This also introduces students to the idea of providing others with constructive criticism to help them create a better program.

Objectives-

Students will be able to give constructive feedback on each other's work and will be able to share their digital story with another person or people.

Standards Emphasized-

Materials-

Teacher	Student
**Determine who/where your students will be sharing their digital story programs with and whether or not they will be provided with feedback on their work.	"Engineering Design Thinking" worksheet (optional) Design Notebook (optional) Computer (to share their program)

Learning Tasks-

(Total Approx. Time: 10-15 minutes)

Approx. Time	**Learning Tasks**	**Purpose/Instructional Strategies**
3-5 mins.	Introduce the last part of the Engineering Design process; sharing your work. Discuss the importance of sharing your work with others and how by doing so you can get helpful feedback to help improve your program. Explain that sharing your work is done in many fields- the sciences, engineering, programming, design- and is hugely important in creating the best possible	By showing students where this step in the Engineering Design process is it helps contextualize where they are in the process and helps to solidify the parts of the whole process. It is important to emphasize that the design process is a cycle and once you share your work and get feedback on it you may wish to go back to the beginning and create a new design that is even better. It is important that students understand that feedback and even criticism from other can be a good thing that motivates them to make their program better.

	products, processes, or ideas. Provide options for the teacher for how to share students' work—teacher only, other students in the class, another class, parents, or online community (other students participating in this module).	
3-5 mins.	Discuss with the class the importance of giving constructive criticism when looking at others' work. This means that you are not only making negative remarks about their work, but you are offering suggestions for how they could improve their program and also including things that they did well. Also explain that just because your project isn't perfect and someone gives you feedback they are not trying to be mean.	By teaching students how to give and receive constructive criticism they can help each other develop better ideas that multiple people have worked on. This collaborative working atmosphere is crucial for many fields and will be a critical skill for students to be familiar with in life and school (playing a team sport, being in a club, working on a group project, etc.)
3-5 mins.	Have students share their work online. You may decide that students simply save their project on the LaPlaya website (in that case students are sharing their work with us), or you may chose to have them share their work with others including you, their classmate(s), students in another class, parents, or online. This choice is up to you.	It is important to make it clear to students who will be seeing their work and whether or not they will be receiving feedback from those people (we will not be providing feedback, but you could ask that their classmates or students in another class do). As an optional extension, if students are provided with feedback from others they may go back to a previous step in the design process (either design—storyboarding and flowcharts—or the application cycle) and make improvements to their program based on that feedback. In this case the iterative nature of the Engineering Design process is truly highlighted.