Cubit Computer Science Kit Curriculum Guide
Welcome to Cubit

Every aspect of Cubit is designed with education in mind because our curriculum is developed and used by teachers. Each unit is created, reviewed and revised based on real world classroom experience and best practices in STEAM education. We do the heavy lifting of creating formal lesson plans so you can focus your time on doing what you love - teaching.

Each Cubit Curriculum Unit consists of a cohesive framework that allows students to unpack big ideas through a series of highly engaging lessons that are aligned to content standards. The projects create the opportunity for students to explore key concepts through hands-on experiences, grapple with misconceptions, and develop a STEAM growth mindset through an infusion of design thinking elements that rewards creative risk taking.

Cubit Curriculum is designed as a flexible, modular system so educators can choose which format works for their needs. Cubit Curriculum Unit Plans include a PDF format for easy downloading and sharing, creating a compact layout to reduce printing demands and ensuring clear pictures when handouts are photocopied in black and white. Unit plans are comprised of simple setup guides, detailed lessons along with at a glance guides, student worksheets, and specific Cubit Workshop plan files that can be used collectively or separately.

cubit

STEAM Education

Computer Science Kit

Middle School Unit Plan

Musical Coders

Table of Contents

cubit

STEAM Education

Curriculum Standards Mapping

Unit Title

Musical Coders

Theme:

How Can We Design An Interactive Musical Light Show?

Tags: Computer Science, Middle School, Programming, K-12 Computer Science Framework

Cubit Kit: Horn Kit (Color LED Strip, Buzzer, Light Sensor, Origami Template)

Grade Level: Middle School

K-12 Computer Science Framework Standards

Practices	Core Concepts	Crosscutting Concepts
Collaborating Around Computing	Algorithms	Abstraction
Recognizing and Defining Computational Problems	Control	Communication and Coordination
Developing and Using Abstractions	Devices	Human–Computer Interaction
Creating Computational Artifacts	Hardware and Software	System Relationships
Testing and Refining Computational Artifacts	Modularity	
	Program Development	
	Troubleshooting	
	Variables	

cubit

STEAM Education

Curriculum Concept Connections

Unit Title

Musical Coders

Theme:

How Can We Design An Interactive Musical Light Show?

Big Ideas:

1. An Algorithm is a Set of Specific Instructions
2. Computers Read Algorithms Literally
3. Loops Repeat Steps Over and Over
4. For Loops Repeat an Algorithm For A Number of Times
5. One Algorithm Can Be Coded In Multiple Ways
6. Blocks (Parts of an Algorithm) can Receive Inputs and Send Outputs
7. Algorithms Use Data as Inputs and Outputs
8. Conditionals Execute Different Algorithms In Different Conditions
9. Variables Are Data with Names
10. Variables Have A Value At Any Given Time
11. Variables Can Be Used to Change Several Parts of an Algorithms at One Time
12. Designers Plan, Build, and Test Prototypes
13. Designers and Engineers Use Feedback To Improve Designs

Essential Questions:

- How do we use software to achieve a goal?
- Why do programs sometimes not work as expected?
- How do we make programs repeat?
- How can different parts of a program communicate with each other?
- How can we write programs so the output depends on the input?
- What steps do designers and engineers use to plan designs?
- How can a program respond to changing input?
- How can a programmer change many parts of an algorithm at the same time?
- How can a user control a program while it is running?
- Why do designers show their designs to others?

STEAM Components

- **Science** (Sound Frequency, Waves)
- **Technology** (Programming, Hardware and Software, Sensors and Outputs)
- **Art** (Music, Color and Light, Design)

- **Engineering** (Form and Function, Prototyping)
- **Math** (Algebraic Thinking, Functions, Variables and Values)

Design Thinking Focus

- **Explore**: Define a Challenge, Identify Solution Criteria, Gather Information
- **Imagine**: Brainstorm Solutions, Evaluate and Select Ideas, Create Design Plans
- **Prototype**: Build A Prototype, Plan and Conduct Testing, Troubleshoot Issues
- **Iterate**: Provide and Receive Feedback, Identify Revisions, Replan, Rebuild, Retest
- **Communicate**: Create and Give Presentations, Create Explanatory Diagrams, Get New Perspectives, Document and Incorporate Insights

Extensions:

1. Creating Chords: Exploring Music Concepts through Engineering A Three-Horn Device
    - Students use three buzzers with origami cones to play sounds simultaneously, forming chords. This extension presents the opportunity for students to learn music concepts having to do with the interaction of sounds, including harmony, consonance and dissonance, intervals and more. Students can focus on practices and principles of computer science, such as creating loops, Gotos, subprograms, and by designing code that is easy to read.
2. Making a Theremin: Turning Movement into Music
    - Students use Ultrasonic Distance smartware to create a version of a theremin, a device which changes the pitch or tempo of the sounds produced by the theremin as their hand gets closer and further from the sensor.
3. Real-World Applications: “Seeing” Music for Hearing-Impaired Populations
    - Students design a device that communicates sound to people with hearing impairments. Using one or more LED strips, or motor smartware, students design a system to represent different sounds for use by people who differ in their ability to hear. Students learn principles of Human-Centered Design to create a user-friendly device, and integrate arts concepts and practices to provide aesthetics in a modality other than sound.

cubit

STEAM Education

Curriculum Thematic Framework

Unit Title

Musical Coders

Theme:

How Can We Design An Interactive Musical Light Show?

Conceptual Flows

Lesson 1 - Programming a Light Show
Lesson 2 - Coordinating Light and Sound
Lesson 3 - Prototyping The Light and Sound Device
Lesson 4 - Programming the Interactive Show
Lesson 5 - Presenting Design Iterations

Lesson Duration: 60 minutes/lesson	**Learning Objectives** Students will be able to:	**Connections**	**Vocabulary**
Lesson 1 Programming a Light Show	Break down an objective into component tasks Write an algorithm with steps specific enough to successfully achieve an objective Use For Loops and Infinite Loops to makes sets of algorithms repeat Debug and revise a program by identifying the cause of the error or undesired behavior	An Algorithm is a Set of Specific Instructions Computers Read Algorithms Literally Loops Repeat Steps Over and Over For Loops Repeat an Algorithm For A Number of Times One Algorithm Can Be Coded In Multiple Ways	Algorithm Event Handling Sequence Debug Loop For Loop Index
Lesson 2 Coordinating Light and Sound	Use data outputs from one part of an algorithm as an input to control aspects of another part of an algorithm Use a conditional to program different algorithms for a TRUE and a FALSE condition	Blocks (Parts of an Algorithm) can Receive Inputs and Send Outputs Algorithms Use Data as Inputs and Outputs Conditionals Execute Different Algorithms In Different Conditions	Input Output Variable Frequency Conditional

Lesson 3 Designing The Light and Sound Prototype	Set project specifications and translate them into software and hardware design Build a prototype of a device displaying a light and music show	Designers Plan, Build, Test, and Iterate On Prototypes	Design Prototype Users Specifications Iteration
Lesson 4 Programming the Interactive Show	Set a variable and use it to update multiple aspects of an algorithm Use a sensor to create an interactive program Create a software plan demonstrating multiple computer science concepts	Variables Are Data with Names Variables Have A Value At Any Given Time Variables Can Be Used to Change Several Parts of an Algorithm At One Time	Parallel Algorithms Sensor Variables
Lesson 5 Prototype Gallery Walk	Present designs and ask questions to solicit feedback Explain computer science concepts used in designs Evaluate the success of designs based on feedback	Designers and Engineers Use Feedback To Improve Designs	Iteration User Testing

cubit

STEAM Education

Lesson 1: Programming a Light Show
Standards and Conceptual Snapshot

Unit Theme:
How Can We Design An Interactive Musical Light Show?

Big Ideas:
- An Algorithm is a Set of Specific Instructions
- Computers Read Algorithms Literally
- Loops Repeat Steps Over and Over
- For Loops Repeat an Algorithm For A Number of Times
- One Algorithm Can Be Coded In Multiple Ways

Essential Questions:
- How do we use software to achieve a goal?
- Why do programs sometimes not work as expected?
- How do we make programs repeat?

Learning Objectives - Students will be able to...
- Break down an objective into component tasks
- Write an algorithm with steps specific enough to successfully achieve an objective
- Debug and revise a program by identifying the cause of the error or undesired behavior

Design Thinking Connection:
- **Explore:** Identify Solution Criteria
- **Imagine:** Brainstorm Solutions
- **Prototype:** Plan and Conduct Testing, Troubleshoot Issues
- **Iterate**: Identify Revisions, Replan, Retest

K-12 Computer Science Framework Standards:

Practices	Core Concepts	Crosscutting Concepts
Recognizing and Defining Computational Problems	Algorithms	Communication and Coordination
Creating Computational Artifacts	Control	Human–Computer Interaction
Testing and Refining Computational Artifacts	Devices	System Relationships
	Hardware and Software	

	Modularity	
	Troubleshooting	

Important Terms:

1. Algorithm
2. Event Handling
3. Sequence
4. Debug
5. Loop
6. For Loop
7. Index

cubit

STEAM Education

Lesson 1: Programming a Light Show

Setup, Preparation and Clean Up

Materials:

- 1 - Cubit Computer Science Kit per student group
  - 1 - Cubit Controller
  - 1 - Light Sensor
  - 1- Color LED Strip
  - 1 - Buzzer
- 1 - USB Power Cable for each Cubit Controller or 4 - AA Batteries
- 1 - Origami Horn template copymaster
- 2 - sheets of cardstock of different colors to print enough templates for one origami cone
- 1 - Flashlight or other handheld light source

Resources:

Cubit Student Quick Guide
Cubit Computer Science Skills Guide
Rubrics (Notebook, Collaboration) and Brainstorming Mindset Guide
Lesson 1 Student Challenge Handout

Preparation before class:

1. Print enough origami horn templates on colored cardstock for one horn.
2. Build the origami horn and attach an LED strip.
3. Display two Cubit kits: one with the origami horn and one without
4. Load the Warm-up plan in Workshop.
5. Launch the plan and display the kit in a place where students can see it
6. Brainstorm ways students can set up the maze for the algorithm activity.
   a. If you have access to open space, consider marking “walls” on the floor using tape. To save time, you may wish to set this up ahead of time.
   b. If you are making a maze in the classroom, consider using chairs for barriers.

Agenda

1. Warm-up: Thinking about the Horns show (5 min)
2. Theme and Essential Questions (5 min)
3. Acting Out Algorithms (15 min)
4. Introduction to Cubit (5 min)
5. Programming an LED light show (25 min)
6. Reflection on Big Ideas (5 min)

Clean up

1. Halt Program, Clear from Cubit, and Save Cubit Workshop Plan File to Drive (if desired)
2. Disconnect from Cubit Controller and Exit Workshop
3. Turn off or disconnect power

4. Put away Kit
5. If necessary, return 4 AA Batteries to the designated space for storage and charging

cubit

STEAM Education

Lesson 1: Programming a Light Show

Lesson Detail

Duration: 60 minutes

Essential Questions:

- How do we use software to achieve a goal?
- Why do programs sometimes not work as expected?

Learning Objective:

- Break down an objective into component tasks
- Write an algorithm with steps specific enough to successfully achieve an objective
- Use For Loops and Infinite Loops to makes sets of algorithms repeat
- Debug and revise a program by identifying the cause of the error or undesired behavior

Differentiation

- Stretch - Custom Code Blocks
- Scaffold - Vocabulary Cards, Student Guides, Provide Start Files

Hook/Warm-up (5 min):

1. Ensure the two kits are displayed and working as students arrive. The kit with the origami horn should be playing, and the other kit should be plugged in but not doing anything.
2. Distribute brainstorm sheets to students.
3. Shine a light over the light sensor.
4. Prompt students to brainstorm in groups how this light show was made, and write down ideas on their brainstorm sheets

Discuss Theme and Essential Questions (5 min)

1. Introduce the Theme: "How Can We Design An Interactive Musical Light Show?". Tell students they will make their own version of the light and music show they have just seen.
2. Say, "Each day we will be answering specific questions to learn concepts that will help us create the interactive musical light show. In the next few lessons we will learn about concepts in computer science and try them out to create our shows.
3. Introduce the Essential Questions:
    - How do we use software to achieve a goal?
    - Why do programs sometimes not work as expected?
    - How do we make programs repeat?

Acting Out Algorithms (15 min)

1. Introduce the maze algorithm challenge

- Tell students they will write a “real life” computer program by giving instructions to help the teacher to navigate the maze. (Note: This activity can be adapted to have a student volunteer navigate the maze.)
- Explain the challenge: “We are going to build a classroom maze and I will navigate it. You all will take turns giving me instructions to tell me how to get from one end of the maze to the other. Your instructions have to make sure I will not run into any of the walls of our maze.

2. Introduce and contextualize the terms hardware and software in the model.
    - Ask, “We are modeling something robots do. Who has heard of the terms hardware and software? What do those mean? In our model, which is the hardware and the software, and why do you think so?
3. Create the maze. Guide students so their a maze includes at least one path across the classroom.
    - In the classroom, have students work together using their chairs or similar objects.
    - In an open space, have students tape off a maze on the floor.
4. Gather students on the outside of the maze, and stand at the “beginning” of the maze.
5. Call on student volunteers to begin providing instructions acting as the “software”. Each student should provide one instruction without correcting. Model interpreting the instructions “too” literally. Try to run into the barriers if students’ instructions are not very specific. Some examples:
    - Keep walking if students do not specify to stop.
    - Begin by taking very small or very large steps.
    - Move very quickly or very slowly.
    - If students do not specify a direction to walk, move in a diagonal line or sideways.
    - When asked to turn, turn too far or keep turning in a circle.
6. When you reach the end of the maze, debrief by congratulating students on creating a successful algorithm and explain the terms “algorithm” and “debugging”.
    - Explain that computers follow instructions literally and these are called “algorithms”. Algorithms need to be specific enough and provide enough information to the hardware to achieve the programmer’s goal.
    - Explain that the process they used to figure out how to give you correct instructions is called “debugging”: figuring out ways to fix problems with a program that prevent it from achieving a goal. Promote a growth mindset by congratulating students on persisting in debugging and emphasizing that debugging is a part of programming in the real world, and that it can be valuable for learning how to code.
7. If time remains, call on a student volunteer to try acting as the navigator “hardware”.

Questions for Understanding to Pose to Students During Activity and Debrief

1. Why don’t computers understand instructions the way that humans do?
2. What are the differences between hardware and software?
3. What can you learn when your instructions do not work as planned?

Welcome to Cubit Introduction (Guided) (5 min)

1. Distribute Cubit Computer Science Kits
2. If necessary, introduce students to the basics of using the Cubit Controller and LED.
    - The Cubit Controller is the “brain” and communicates with the software (Workshop) and hardware (Cubit smartware)
    - We create (program) the instructions.
    - The smartware we will use today is the LED light strip, which has ten LEDs capable of showing different colors.
    - The cable connects each motor and sends instructions from the Cubit controller to the motor
    - When we change the code, the LED lights change (remind students that the Launch button sends code changes to the Cubit Controller)
3. Guide students through connecting to their Cubit controller to their computers and connecting the LED smartware.
    - Students Open Cubit Workshop
    - Connect the LED strip to the Cubit Controller with a cable connector
    - Connect a USB or batteries to the Cubit Controller
    - From Workshop, select Your Cubit Name from the Cubit Selector menu and connect your Cubit with the Connect button. If necessary, check Bluetooth Setting on student devices
    - The LED Symbol will appear on the Port in the Block Menu and the connection on the Cubit Controller will light up blue.
    - Select a Block from this Menu
    - Send your plan file to your Cubit Controller by selecting Launch

Programming an LED Light Shows (25 min)

1. Distribute “Light Show Algorithm Task Card” task card. Say, “Now we will begin programming algorithms for our Cubits to control a light show using the LED smartware (a type of hardware).
2. Project Brainstorming Mindset Guide and discuss each guideline.
    - Say, “These guidelines help us to work together in teams when coming up with ideas.” Briefly discuss each guideline.
3. Direct students to create their light shows. Circulate and pose Questions for Understanding to groups as they work.
4. If time remains, use the last few minutes to debrief the terms “Loop” and “Index”.

Questions for Understanding to Pose to Students During Collaborative Group work

1. Did anything happen when you launched your code that you did not expect? What happened?
2. Describe how and ***why*** you changed your plan file. How did you figure out what was wrong?
3. What does a Loop mean? Why is a loop useful for a light show?

4. Explain what you found out about the term index while you were programming.

Reflection on Big Ideas

1. Ask 1-2 students what they learned about algorithms. Summarize the discussion with the first two Big Ideas:
    a. An Algorithm is a Set of Specific Instructions
    b. Computers Read Algorithms Literally
2. Ask 1-2 students what they learned about the word loop. Summarize the discussion with the second two Big Ideas:
    a. Loops Repeat Steps Over and Over
    b. For Loops Repeat an Algorithm For A Number of Times
3. Remind students that they replaced the “Do N Times” block with a “For Loop” block, and that they made these two blocks do the same thing. Summarize with the last Big Idea: One Algorithm Can Be Coded In Multiple Ways
4. If time remains, ask for examples of debugging. Ask if any group could not figure out how to debug their program. Congratulate all students for debugging efforts. Emphasize that programmers in the real world collaborate to debug, because it can be difficult, and new perspectives can bring new ideas

Common Misconceptions to Address in this Lesson

- Computers understand instructions the same way humans do
- The functionality of hardware does not depend on software, and vice versa
- There is no way to fix some programming problems
- The only way to make an algorithm repeat is to write it over and over.

cubit

STEAM Education

Light Show Algorithm Task Card

Designing your Light Show

1. You will include at least one algorithm in your light show, and make the light show repeat over and over. Pick an algorithm below and follow the instructions to create a light effect on the LED strip.
2. Once your first algorithm is complete, you will write a second algorithm and connect your two algorithms together.
3. When you finish both algorithms, go to the last section to make code to end your light show.
4. If you have time, you can add more algorithms, or create your own algorithms using what you've learned.

Algorithm Menu

1. Fading Colors

1. **Goal: Create an algorithm that makes the light change colors slowly from one color to another, and repeat for certain number of times.**
2. Place two "Change LED colors over time" blocks. Choose your **parameters** -- the color you want, and the number of seconds it takes to change to each color -- and test your code.
    i. Did it work as your expected? Debug your algorithm.
3. Place a "Do N Times" block. Experiment to make your fading color algorithm repeat a number of times.
    i. What is the position of your "Do N Times" block in your algorithm? That is, is it the first step, last step, or a step in the middle?
4. Now, place a For Loop block. Take out the "Do N Times" block and replace it with the "For Loop" block. Experiment to make your fading color algorithm repeat in the same way as it did using the "Do N Times" block.
    i. Did this help you understand what "Index" means? Experiment to find out.
5. Connecting algorithms:

i. If you will be adding a **new** algorithm to your light show, connect the first block of your new algorithm to the “Completed” pin of your “For Loop”. You may want to use Regions to organize your code -- it may get big!
ii. If you are **not** adding new algorithms, you will make your sequence of two (or more) algorithms repeat forever. Go to the last step of this card.

2. Flashing Colors

1. **Goal: Make some of the lights on the LED strip flash colors while other lights stay the same color.**
2. Place a “Set All LEDs to Color” block for your starting color.
3. Place at least two “Set Indexed LED Color” blocks. Experiment to find out how to make at least two different LEDs change from the original color to a new color, at the same time.
    i. Did you see your lights change color? Or did they seem to start out as the color you wanted them to change to? Try looking in the “Flow” menu to make the Cubit wait before moving on to the “Set Indexed LED Color” blocks. Use the Debug button to help you.
    ii. How did you connect the purple wires to make your selected LEDs flash different colors at the same time?
    iii. Did this help you understand what “Index” means? Experiment to find out.
4. Now, place a For Loop block. Experiment to make your flashing color algorithm repeat a number of times. Use what you know about the term “index” to figure out how.
    i. What is the position of your “Do N Times” block in your algorithm? That is, is it the first step, last step, or a step in the middle?
5. Connecting algorithms:
    i. If you will be adding a **new** algorithm to your light show, connect the first block of your new algorithm to the “Completed” pin of your “For Loop”. You may want to use Regions to organize your code -- it may get big!
    ii. If you are **not** adding new algorithms, you will make your sequence of two (or more) algorithms repeat forever. Go to the last step of this card.

3. Fill the Strip

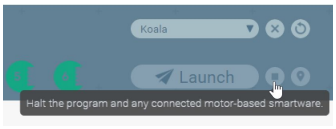
1. **Goal: Change the color of the LED strip from one to another by changing it two LEDs at a time.**
2. Set a starting color using the “Set All LEDs to Color” block.
3. Start changing the individual LEDs on your strip by placing “Set LED Range Color” blocks. Experiment to figure out how to make sets of two LEDs change to a new color until all 10 LEDs have changed.
    i. Did you see your lights change color? Or did they seem to start out as the color you wanted them to change to? Try looking in the “Flow” menu to make the Cubit wait before moving on to the “Set LED Range” blocks. Use the Debug button to help you.
    ii. What is a range?
4. Place a “Do N Times” block. Experiment to make your filling color algorithm repeat a number of times.
    i. What is the position of your “Do N Times” block in your algorithm? That is, is it the first step, last step, or a step in the middle?
5. Now, place a For Loop block. Take out the “Do N Times” block and replace it with the “For Loop” block. Experiment to make your filling color algorithm repeat a number of times. Use what you know about the term “index” to figure out how.
6. Connecting algorithms:
    i. If you will be adding a **new** algorithm to your light show, connect the first block of your new algorithm to the “Completed” pin of your “For Loop”.
    ii. If you are **not** adding new algorithms, go to the last section to figure out how to make it repeat forever until you tell it to stop.

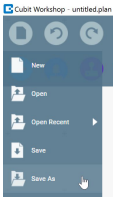
Creating an Infinite Loop that stops on a button press

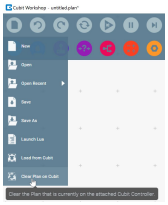
1. Do this only after you have completed your second (or more) algorithm.
2. Place a “Cubit Button Pressed?” block. Connect the “Completed” pin to the “Cubit Button Pressed?” block.

3. Place a “Turn LEDs Off” block and connect it to the “Yes” pin of the “Cubit Button Pressed?” block.
4. Connect the “No” pin of the “Cubit Button Pressed?” block. To the “Start” pin of the first For Loop in your light show sequence.
5. Test it out. All your For Loops will complete -- your algorithm will not stop immediately when you press the button on the Cubit!
6. This is called **Event Handling**: when part of an algorithm checks whether an event happened (like a button press), and then does something new after the event.
7. Congratulations on your light show! If you have time, you can add more light shows to your algorithm, or create your own algorithm using what you’ve learned. Don’t forget to save your plan file!

C. Wrapping Up







1. Stop the Program
2. Save your plan file to Google Drive Folder.
3. Clear your plan from your Cubit Controller
4. Exit Workshop by clicking on the X in the upper right corner
5. Return your Cubit Controller, Cables, Smartware, and Power Source as instructed!

Interactive Light and Music Project Brainstorm Worksheet

Observe how the Light and Music Project works. How do you think it was made? Why does it act the way it does? What strategies do you think you would use to make your own version? Brainstorm your ideas below.

Lesson 2: Coordinating Light and Sound
Standards and Conceptual Snapshot

Unit Theme:
How Can We Design An Interactive Musical Light Show?

Big Ideas:
- Blocks (Parts of an Algorithm) can Receive Inputs and Send Outputs
- Algorithms Use Data as Inputs and Outputs
- Conditionals Execute Different Algorithms In Different Conditions

Essential Questions:
- How can different parts of a program communicate with each other?
- How can we write programs so the output depends on the input?

Learning Objectives - Students will be able to...
- Use data outputs from one part of an algorithm as an input to control aspects of another part of an algorithm
- Use a conditional to program different algorithms for a TRUE and a FALSE condition

Design Thinking Connection:
- **Explore:** Gather Information
- **Imagine:** Brainstorm Solutions, Evaluate and Select Ideas, Create Design Plans
- **Prototype:** Plan and Conduct Testing, Troubleshoot Issues

K-12 Computer Science Framework Standards:

Practices	Core Concepts	Crosscutting Concepts
Collaborating Around Computing	Algorithms	Communication and Coordination
Creating Computational Artifacts	Control	System Relationships
Testing and Refining Computational Artifacts	Devices	
	Hardware and Software	
	Modularity	
	Program Development	

	Troubleshooting	

Important Terms:

1. Input
2. Output
3. Variable
4. Frequency
5. Conditional

Lesson 2: Coordinating Light and Sound

Setup, Preparation and Clean Up

Materials:

- 1 - Cubit Computer Science Kit per student group
  - 1 - Cubit Controller
  - 1 - Light Sensor
  - 1 - Color LED Strip
  - 1 - Buzzer
- 1 - USB Power Cable for each Cubit Controller or 4 - AA Batteries
- 1 - copy of the Linking Lights to Music Task Card per student group

Resources:

Cubit Student Quick Guide
Cubit Computer Science Skills Guide

Preparation before class:

1. Draw Warm-up frequency diagrams on the board:
   a. Draw a waveform of 1 cycle (1 peak and 1 valley) along an X-axis. Label the X-axis to show that the cycle completes over 1 second. Label it “1 Hz”.
   b. Draw a waveform of many cycles along another 1 second X-axis such that all the cycles complete over the 1 second, and label accordingly. For example, draw 8 cycles (8 peaks and 8 valleys) over 1 second, and title the waveform “8 Hz”.
   c. Draw another 1 second X-axis, but do not draw a waveform. Title it “262 Hz (C note)”. Write a question mark where the waveform would be.
2. Make copies of the Linking Lights to Music Task Card, one per student group.

Agenda

1. Warm-up: The Science of Music (10 min)
2. Theme and Essential Questions (5 min)
3. Writing Algorithms to Coordinate Sound and Light (40 min)
4. Reflection on Big Ideas (5 min)

Clean up

1. Halt Program, Clear from Cubit, and Save Cubit Workshop Plan File to Drive (if desired)
2. Disconnect from Cubit Controller and Exit Workshop
3. Turn off or disconnect power
4. Put away Kit
5. If necessary, return 4 AA Batteries to the designated space for storage and charging

cubit

STEAM Education

Lesson 2: Coordinating Light and Sound

Lesson Detail

Duration: 60 minutes

Essential Questions:

- How can different parts of a program communicate with each other?
- How can we write programs so the output depends on the input?

Learning Objective:

- Write an algorithm that uses data from one type of smartware to control aspects of another type of smartware

Differentiation

- Stretch - Custom Code Blocks, Create additional light pattern algorithms
- Scaffold - Vocabulary Cards, Student Guides, Provide Start Files

Hook/Warm-up: The Science of Music (10 min):

1. Explain that today students will be working with the buzzer smartware to create the music for their projects. Tell them they will be using sound data, and need to learn about this data before using it in their programs.
2. Direct students' attention to the waveforms on the board. Ask, "What do you think these represent?" Accept all responses.
    - Tell students that the drawings are called waveforms and represent sound waves. Give examples of other waves that can be represented by waveforms such as light waves, a jump rope, or the movement of water waves in the ocean.
3. Discuss the phenomenon that sound waves represent. Ask,
    - "What real-life phenomenon does a waveform of sound represent?"
    - "Think about ocean waves. We can draw a waveform of their movement that would look like the motion of the water. How are sound waves the same? How are they different?"
    - Guide students to an understanding that sound waves represent the movement of air molecules. Things that make sounds move air molecules and create a wave, just like splashing water creates waves by moving water molecules
    - Tell students that we hear sounds when sound molecules hit our eardrums and our brains interpret the motion of our eardrum. Say,
        - Say, "The peaks of a sound waveform represent the air molecules compressing, or exerting a very small push. The valleys of a sound waveform represent the air molecules becoming more spaced out, exerting a very small pull. Our eardrums move back and forth from these pushes and pulls, which we call vibration.
4. Introduce the idea of frequency. Ask,

- "What do you think Hertz means? What does it measure?"
- Guide students to an understanding that a sound's frequency is a measure of cycles over time. Connect the term frequency to the more familiar term frequent.
- Say, "The frequency of a sound measures how frequent the wave cycles are. Hertz (Hz) completes a full cycle over some period of time. Hz is a measurement of the

5. Connect to music by discussing the 262 Hz (C note) waveform.
    - Ask, "What would this 262 Hz waveform look like? Why do you think I did not draw it?"
    - Say, "If you played this frequency on an instrument, you would hear a C note in the 4th octave."
    - If desired, give additional examples:
        - "If you played a sound with a frequency of 392 Hz, you would hear a G note in the 4th octave."
        - "If you played a sound with a frequency of 131 Hz, you would hear a C note in the 3th octave, so it sounds lower than a C in the 4th octave."
        - "If you played a sound with a frequency of 4186 Hz, you would hear a C note in the 8st octave. That's very high pitched!"
    - Say, "When we hear different sounds in music, we hear changes to the waves that are produced by whatever object is vibrating to create the sound."

Discuss Theme and Big Idea (5 min)

1. Remind students of the Theme: "How Can We Design An Interactive Musical Light Show?". Tell students they will be adding to their Loop algorithm from the previous lesson to create the program of their Interactive Musical Light show.
2. Introduce the Essential Questions:
    - How can different parts of a program communicate with each other?
    - How can we write programs so the output depends on the input?
3. Ask students how they might define the terms input and output. Ask students to interpret the meaning of the second question.

Writing Algorithms to Coordinate Sound and Light (40 min)

1. Pass out the Linking Lights to Music Task Cards, one per student group.
2. Explain that students will complete two activities.
    - In Task 1, they will follow instructions to coordinate the LED lights with the song, Twinkle Twinkle Little Star.
    - In Task 2, they will add their own coordinated music and light algorithm to the loop algorithm they wrote in the previous lesson. They will use these algorithms for their Interactive Musical Horn prototype, which they will design in the next lesson.
3. Prompt students to begin working through the tasks. Circulate to provide support and ask Questions for Understanding.

Questions for Understanding to Pose to Students During Tasks

1. What is happening to the frequency data as the song plays?
2. When one LED lights up after another, what is happening to the LED index? What is its value?
3. What would happen if you connected the frequency data pin directly to the LED index pin?

Reflecting on Big Ideas (5 min)

1. Introduce each Big Idea, and ask volunteers to explain what they learned about each idea during their task activities:
2. Big Idea: Blocks (Parts of an Algorithm) can Receive Inputs and Send Outputs
    - “What data was an input in your algorithm? What data was your output?”
    - “How did you know whether a block in your algorithm could receive input or send input? How is that represented on the block?”
3. Big Idea: Algorithms Use Data as Inputs and Outputs
    - “How did your input change what your algorithm did? What was the output?”
    - If students need support, ask them to think back to the Convert Value block.
4. Big Idea: Conditionals Execute Different Algorithms Under Different Conditions
    - “What was the conditional in your algorithm?”
    - “Does anyone have an example of a question that is a conditional in English? What makes it a conditional?”

Common Misconceptions to Address

- Sounds are an object; sounds are not the movement of air molecules
- A waveform does not represent the movement of a substance.
- Every part of an algorithm works independently of every other part of an algorithm.
- Smartware (or hardware) cannot work together or communicate with each other.

Lesson 2: Coordinating Light and Sound

Task Card: Linking Lights to Music

Task 1

Goal:

Coordinate lights and music by making different LEDs light up on different notes of the song, and turn the LEDs off when no sound plays.

Coordinating Lights With the notes "Twinkle Twinkle Little Star"

1. Create a **Play Sequence** block from the Buzzer Menu and connect the Start block to the Play pin.
2. In this example we will use Twinkle Twinkle Little Star. The first 16 notes are:
   CCGGAAG(Rest)
   FFEEDDC(Rest)
   (Rest is the word for a pause in music)
4. Click on the Edit Button on the Play Sequence block and enter the notes above.
5. Hit Launch and the Buzzer will play Twinkle Twinkle Little Star.
6. To add lights to our song, we will start by using the Logging block. We need to find the Frequency Range for our notes so we can use that information to drive the LED strip.
7. Create a **Log Values Over Time** Block from the Logging menu.
8. Connect the Purple On Note pin to the Purple Log Values pin.
9. Connect the Blue Freq pin to the Blue Y-Value pin.
   - Freq stands for Frequency, which is a number representing the pitch of the note (Pitch is the word in music for how high or low a note is).
10. Hit launch again to see what frequencies are playing with each note.
    - Notice that during the rests, the value of the frequency is 0.
    - Look at the highest Y-Values and the lowest Y-Value, ignoring the zeros. This is the range you will use to have different lights turn on with different frequencies.
    - **Hint**: If you are having trouble seeing the exact values on the graph, you can use a Display Value block and watch it closely over time, looking for the highest and lowest numbers.
    - For this song, excluding the rests (0's), the range is 262 to 440.
11. You can now delete the **Log Values Over Time Block.**
12. We are going to make the LEDs turn off on the rests (0's). To do this, we need to tell the Cubit to do different things when no note plays (a frequency of 0) versus when a note plays (a frequency greater than 0).
13. Create a Compare > Comparators > **Are Value Equal?** Block. Connect the A==B pin to the On Note pin of the Play Sequence block.
    - Notice that there are two output pins (the purple pins on the right side of the block), **Equal** and **Not Equal**. These are two **conditions**: the TRUE condition (Equal) and the FALSE condition (Not Equal). You will use this block to run

different algorithms under different conditions -- in the condition when the value is equal to 0, and the condition when the value is not equal to 0.
    - That is why this block is called a **Conditional**: it gives instructions to do different things in the TRUE and FALSE conditions.
14. Now we will have the Play Sequence block send the frequency to the Are Values Equal? Block, to check whether or not it is 0.
    - Connect the Freq(uency) pin to the A pin.
15. Set the B value to 0. The block will compare the value of A (the frequency) to the number 0.
16. Create a LED Strip > **Turn LEDs Off** Block and connect it to the Equal pin. We can say this in English: “If the current frequency is equal to 0, turn the LEDs off”.
17. Now we will have different LEDs light up when different notes play. We will use the frequency data as the **input** and the LED light index as the **output.**
    - However, the frequency data and LED index data are different ranges: the LEDs are numbered 1 through 10, and the frequency range is 262 to 440.
    - The goal is to change our frequency from the range 262 to 400 to the range 1 to 10 to drive the LED lights.
18. Create a Math > **Convert Value** block connect the Convert Value pin to the Not Equal pin.
19. Connect the Blue Freq(uency) pin to the Blue Value pin. This is the value we will be converting. The goal is to change our frequency from the range 262 to 400 to the range 1 to 10 to drive the LED lights.
20. Set the Input Range of the Convert Value block to be 262 for Input Start and 440 for Input End.
21. Set the Output Range of the Convert Value block to be 1 for the Output Start and 10 for the Output Range.
22. **Explanation**: The Convert Value block is dividing the large frequency range (262-440) into 10 smaller ranges (1-10). The length of these smaller ranges is 17.8 (do the math!). This means that frequencies between 262 and 279.8 (which is 262 + 17.8) will light up the first LED light (index 1), because the Convert Value block converts any frequency input in this range into a result of 1.
23. Create a LED Strip > **Turn LEDs Off** Block and connect it to the output pin of the Convert Value Block.
24. Create a LED Strip > **Set Indexed LED Color** Block and connect it to the output pin of the Turn LEDs Off Block.
25. Pick a color by clicking on the white color selector button of the Set Indexed LED block.
26. Connect the Result pin of the Convert Value Block to the Index pin of the Set Indexed LED Color Block.
27. Hit Launch and watch the amazing results of your coordinated music and light algorithm.

Lesson 2: Coordinating Light and Sound

Task Card: Linking Lights to Music

Task 2

Goal:

Add your own Coordinated Music and Light Algorithm to your Loop algorithm for the final Interactive Show project.

Write your own Coordinated Music and Light Algorithm

1. Now you can make your own coordinated music and light algorithm! You will add one to the plan you made in the previous lesson, for your Interactive Show. The Music and Light algorithm can go before or after your loop algorithm.
2. Load your plan and place the first block of your music and light algorithm like you did in this example by placing a Play Sequence block.
3. If you want your Music and Light algorithm to happen **before** your loop algorithm:
    - Connect the Play pin of the Play Sequence block to the Start block.
    - Then, connect the Done pin of the Play Sequence block to the Start pin of your For Loop block.
4. If you want your Music and Light algorithm to happen **after** your loop algorithm:
    - Connect the Play pin of the Play Sequence block to the Start block.
    - Then, connect the Done pin of the Play Sequence block to the Start pin of your For Loop block.
5. Plan and write an algorithm for a light show in the same way you did with Twinkle Twinkle Little Star. Test and revise until it works as you planned.
6. Make an Infinite Loop
    - If your **Music and Light algorithm is first** and your Loop algorithm is **second,** connect the Completed pin of your for loop block to the Play pin of the Play Sequence block
    - If your **Loop algorithm is first** and your Music and Light algorithm is second, connect the Done pin of your Play Sequence block to the Start pin of your For Loop block.
7. Tip: You can “clean up” your code in Workshop by making “shortcuts” called GoTos. Professional programmers keep their code simple so it is easier to read. You can too!
8. Create a GoTo by hovering your cursor over the wire making your two algorithms into an infinite loop. A +GoTo icon will appear.
    a. Click the icon and give your GoTo a label. Press OK.
    b. Notice that the wire will turn into two GoTo arrows. When the Cubit gets to the GoTo arrow at the end of your algorithms, it will “go to” the matching arrow with the same name.

Lesson 3: Prototyping The Light and Sound Show
Standards and Conceptual Snapshot

Unit Theme:
How Can We Design An Interactive Musical Light Show?

Big Ideas:
- Designers Plan, Build, Test, and Iterate On Prototypes

Essential Questions:
- What steps do designers and engineers use to create prototypes?

Learning Objectives - Students will be able to...
- Set project specifications and translate them into software and hardware design
- Build a prototype of a device displaying a light and music show

Design Thinking Connection:
- **Explore**: Define a Challenge, Identify Solution Criteria
- **Imagine**: Brainstorm Solutions, Evaluate and Select Ideas, Create Design Plans
- **Prototype**: Build A Prototype, Troubleshoot Issues
- **Iterate**: Provide and Receive Feedback, Identify Revisions, Replan, Rebuild
- **Communicate**: Create Explanatory Diagrams, Document and Incorporate Insights

K-12 Computer Science Framework Standards:

Practices	Core Concepts	Crosscutting Concepts
Recognizing and Defining Computational Problems	Hardware and Software	Human–Computer Interaction
	Devices	System Relationships
	Troubleshooting	

Important Terms:
1. Design
2. Prototype
3. Users
4. Specifications
5. Iteration

cubit

STEAM Education

Lesson 3: Prototyping The Light and Sound Show

Setup, Preparation and Clean Up

Materials:

- 1 - Cubit Computer Science Kit per student group
  - 1 - Cubit Controller
  - 1 - Light Sensor
  - 1 - Color LED Strip
  - 1 - Buzzer
- 1 - USB Power Cable for each Cubit Controller or 4 - AA Batteries
- 1 - Origami Horn template copymaster
- 2 - sheets of cardstock of different colors per student group (or more), to print enough templates for one origami cone per group. Choose at least 3 colors of cardstock.
- Glue or gluesticks, different varieties of tape, or other adhesive materials
- 1 - pair of scissors per group; 2 pairs if possible to make prototyping faster
- Any materials that can be used to build the projects. Suggestions: cardboard, plastic soda bottles, egg cartons, shoeboxes, pipe cleaners, straws, popsicle sticks, string, etc.
- 1 - permanent markers for labeling designs
- 1 - folder (or more) to keep Design Planning and Revisions worksheets for future lesson
- 1 - box paperclips to keep Design Planning and Revisions worksheets together for future lesson

Resources:

Cubit Student Quick Guide
Cubit Computer Science Skills Guide
Rubrics (Notebook, Collaboration) and Brainstorming Mindset Guide
Lesson 1 Student Challenge Handout

Preparation before class:

1. Write the Warm-Up discussion question on the board: “If you were a designer, what would you do to create a new product?”
2. Gather materials for students to build their designs. Consider asking students to bring materials from home several days in advance.
3. Print 2 origami horn templates on colored cardstock per group. Distribute across different colors enough for each group to have a choice of colors.
4. Have origami horn from Lesson 1 on hand. It does not need to be connected to a Cubit.
5. Ensure there is a space for students to keep their prototypes to revise and present in future lessons.
6. Identify a folder or other place to keep student design planning worksheets for the next lesson.

Agenda

1. Warm-up: What do Designers Do? (5 min)

2. Theme and Essential Question (5 min)
3. Setting Design Specifications (15 min)
4. Prototyping Designs (35 min)

Clean up

1. Turn off or disconnect power
2. Put away Kit
3. Store prototypes
4. Store design planning worksheets for students to use in the next lesson.
5. Return and clean up building materials
6. If necessary, return 4 AA Batteries to the designated space for storage and charging

cubit

STEAM Education

Lesson 3: Prototyping the Light and Sound Show

Lesson Detail

Duration: 60 minutes

Essential Question:

- What steps do engineers use to plan designs?

Learning Objective:

- Set design specifications and translate them into a physical and software design
- Build a prototype of a device displaying a light and music show

Differentiation

- Stretch - Custom Code Blocks
- Scaffold - Vocabulary Cards, Student Guides, Provide Start Files

Hook/Warm-up: What do Designers Do? (5 min):

1. Prompt students to discuss the Warm-Up Question: “If you were a designer, what would you do to create a new product?” Provide 3-4 minutes for partners or groups to discuss.
2. In the last 1-2 minutes, ask volunteers to share their ideas.

Discuss Theme and Essential Question (5 min)

1. Remind students of the Theme: “How Can We Design An Interactive Musical Light Show?”. Explain that today they will be creating their plans, and building the device that will play their musical light show.
2. Introduce the Essential Question: “What steps do designers and engineers use to plan designs?”

Setting Design Specifications (15 min)

1. Hand out the Design Planning Worksheets to each group. Explain that they will use this to guide their design of their Interactive Musical Light Show. Assign one student per group to write on the worksheet.
2. Project Brainstorming Mindset Guide and discuss each guideline.
    - Say, “These guidelines help us to work together in teams when coming up with ideas.” Briefly discuss each guideline.
3. Allow students to complete the worksheets and their designs. Set up the example Circulate and ask questions for understanding.

Questions for Understanding to Pose to Students During Planning

1. Why not just start building your design right away?
2. Why is it important to think about users?
3. Why is it important to think about the needs of users?
4. How will it help to write down your specifications?

5. How will it help to draw a diagram?

Prototyping Designs (35 min)

1. Tell students they will now build their designs.
2. Distribute the Design Revisions worksheet and read the introduction on the worksheet aloud.
    - Tell students that it is important to record revisions so they can see their progress in the design process.
    - Emphasize that there may not be enough time to make all revisions, and that they will have time in future lessons to revise their prototypes.
3. Encourage students to prototype quickly by helping each other and distributing the work. Everyone should have something to do at all times.
4. Distribute Cubit Computer Science Kits and printed Origami templates to each group.
5. Make scissors, tape, glue, permanent markers, and other materials available.
    - Tell students not to glue anything directly to smartware or the Cubit. Tell students their designs must be able to come off the device.
6. Walk students through how you built the the origami horn from Lesson 1.
7. Prompt groups to begin creating their designs, starting with the origami horn. Circulate to provide assistance, remind students to record revisions, and ask Questions for Understanding.
8. When 5 minutes remain, debrief the activity.
    - Congratulate students on successfully engaging in designing prototypes. Emphasize that running into issues is part of the design process. Mention the Big Idea: “Designers Plan, Build, Test, and Iterate On Prototypes”
9. Have students paperclip their Design Planning and Revisions Worksheets together. Collect them and store them for the next lesson.
10. Direct students to label the pieces of their prototypes and begin cleaning up.

Questions for Understanding to Pose to Students During Collaborative Prototyping

1. Now that you are building your design, do you still think it will meet your specifications?
2. Explain ***why*** your proposed revisions are important.

Common Misconceptions to Address

- Designing is usually a quick, smooth process.
- Planning is not necessary for most designs.
- Encountering problems or issues is a “bad thing”.
- Designers usually do not need to revise their designs; iteration is not necessary.
- Building things has to take a very long time; there is no such things as “quick prototyping”.

cubit

STEAM Education

Design Revisions Worksheet

Revising your Interactive Musical Light Show Prototype

Making Revisions

You may discover issues or problems with your design. This is valuable information! It can be used to make revisions.

Whenever you discover a problem with your original design, make a Revision Record. One Revision Record has been written out for you. Create a new Revision Record for each revision your team would like to make.

Revision Record 1

1. The problem:
2. Why it is a problem:
3. Our new idea:
4. We think this will work because:
5. We will revise it: **Now** **Later**

Write additional Revision Records below:

cubit

STEAM Education

Design Planning Worksheet

Designing your Interactive Musical Light Show

Materials

1. Cubit controller
2. Light sensor
3. LED strip
4. Buzzer
5. Origami Horn materials
6. Other materials provided by your teacher

Design Planning

Brainstorm all of the following in your group. Write down your ideas.

1. Who are the **users** use your Interactive Musical Light Show? What kinds of people are you designing this device for? Brainstorm ideas together.

2. What are the **needs** of your **users**? What is its **purpose** for them? Why would these people want to use your device? Brainstorm ideas together.

3. Write down **specifications** for your device. What must your device do, or what parts must it include, to achieve its **purpose** for the **users**?

Brainstorm ideas together. Start with number 1 in each section, and write as many **specifications** as you need.

Device and hardware specifications: 1.
Software specifications: 1.

4. Brainstorm a design for your device to meet the **specifications**. Remember that every group will create the Origami Horn and attach it to the Buzzer. How will your design

include the Light Sensor, the LED strip, and any other materials to meet your **specifications**?

Choose a design, then draw a diagram. Be sure to label the parts.

cubit

STEAM Education

Lesson 4: Programming the Interactive Show

Standards and Conceptual Snapshot

Unit Theme:

How Can We Design An Interactive Musical Light Show?

Big Ideas:

- Variables Are Data with Names
- Variables Have A Value At Any Given Time
- Variables Can Be Used to Change Several Parts of an Algorithm At One Time

Essential Questions:

- How can a program respond to changing input?
- How can a programmer change many parts of an algorithm at the same time?
- How can a user control a program while it is running?

Learning Objectives - Students will be able to...

- Set a variable and use it to update multiple aspects of an algorithm
- Use a sensor to create an interactive program
- Create a software plan demonstrating multiple computer science concepts

Design Thinking Connection:

- **Explore**: Identify Solution Criteria
- **Imagine**: Brainstorm Solutions, Evaluate and Select Ideas, Create Design Plans
- **Prototype**: Plan and Conduct Testing, Troubleshoot Issues
- **Iterate**: Identify Revisions, Rebuild, Retest

K-12 Computer Science Framework Standards:

Practices	Core Concepts	Crosscutting Concepts
Collaborating Around Computing	Algorithms	Communication and Coordination
Creating Computational Artifacts	Control	System Relationships
Testing and Refining Computational Artifacts	Devices	
	Hardware and Software	
	Modularity	

	Program Development	
	Troubleshooting	
	Variables	

Important Terms:

1. Parallel Algorithms
2. Sensor
3. Variables

Lesson 4: Programming the Interactive Show

Setup, Preparation and Clean Up

Materials:

- 1 - Cubit Computer Science Kit per student group
  - 1 - Cubit Controller
  - 1 - Light Sensor
  - 1 - Color LED Strip
  - 1 - Buzzer
- 1 - USB Power Cable for each Cubit Controller or 4 - AA Batteries
- 1 - copy of the Programming the Light-Controlled Ping Pong Worksheet & Task Card per student group
- 1 - Flashlight or other light source per group.

Resources:

Light-Controlled Ping Pong plan file
Light-Controlled Ping Pong Worksheet & Task Card copymaster
Cubit Student Quick Guide
Cubit Computer Science Skills Guide

Preparation before class:

1. Gather enough flashlights or other light sources for each group to use one. Consider asking students to bring a flashlight from home. If students have smartphones with flashlights, these may also be used.
   a. Test light sources to make sure they are sufficiently bright by creating a Workshop plan with a Read Light Sensor Data block, and checking the values. The best light sources will provide enough light for the Sensor to read 1000.
2. Make sure students have access to the Light-Controlled Ping Pong plan file.
3. Make copies of the Light-Controlled Ping Pong Worksheet & Task Card, one per student group.
4. Place a Cubit with all smartware attached in a place where the class can see it.
5. Project Workshop and load the Light-Controlled Ping Pong plan file onto the Cubit at the front of the class.

Agenda

1. Warm-up: Understanding Parallelism (5 min)
2. Theme and Essential Questions (5 min)
3. Understanding Variables and Finishing Interactive Light Show Plans (45 min)
4. Reflection on Big Ideas (5 min)

Clean up

1. Halt Program, Clear from Cubit, and Save Cubit Workshop Plan File to Drive (if desired)
2. Disconnect from Cubit Controller and Exit Workshop

3. Turn off or disconnect power
4. Put away Kit
5. If necessary, return 4 AA Batteries to the designated space for storage and charging

cubit

STEAM Education

Lesson 4: Programming the Interactive Show

Lesson Detail

Duration: 60 minutes

Essential Questions:

- How can a program respond to changing input?
- How can a programmer change many parts of an algorithm at the same time?
- How can a user control a program while it is running?

Learning Objective:

- Set a variable and use it to update multiple aspects of an algorithm
- Use a sensor to create an interactive program
- Create a software plan demonstrating multiple computer science concepts

Differentiation

- Stretch - Custom Code Blocks, Extend Light Show algorithms, Add algorithm for event handling by Cubit button press
- Scaffold - Vocabulary Cards, Student Guides, Provide Start Files

Hook/Warm-up: Understanding Parallelism (10 min):

1. Project Workshop and set up a Cubit with all smartware attached.
2. Turn on Debugging and launch the Ping Pong Light Interactive plan.
3. Tell students to watch which blocks are active in the top and bottom algorithms (in the green Ping Pong Effect region and the orange Wait Time Variable.
    - Ask, “Which blocks are active at any given time?”
    - “Are they running in order, or at the same time?”
4. Explain parallelism. Say,
    - “When more than one algorithm runs through a single processor at one time, this is called **parallelism**. The two algorithms in our plan run in parallel.” If students are not comfortable with the word parallel, explain using parallel lines on the board. Connect this to the way the different algorithms in the plan are arranged.
5. Explain how processors like the Cubit handle algorithms running in parallal.
    - Say, “We see the algorithms running simultaneously. Actually, the processor inside the Cubit processes each algorithm in sequence, that is, one block after another. However, processors like the Cubit work so fast that the two algorithms run at about the same time -- there are only tiny fractions of a second between the times the Cubit processes each part of an algorithm.
    - “Many computers have more than one processor, and when a different algorithm runs in each processor at the same time, this is true parallelism.”

Discuss Theme and Big Idea (5 min)

1. Remind students of the Theme: “How Can We Design An Interactive Musical Light Show?”. Tell students they will be learning a final computer science concept today, and

then completing the plans for the Interactive Musical Light show they will run on their prototypes.

2. Introduce the Essential Questions:
    - How can a program respond to changing input?
    - How can a programmer change many parts of an algorithm at the same time?
    - How can a user control a program while it is running?

Understanding Variables and Finishing Interactive Light Show Plans (45 min)

1. Pass out the Linking Lights to Music Task Cards, one per student group.
2. Explain that students will complete two activities.
    - In Task 1, they will follow instructions to coordinate the LED lights with the song, Twinkle Twinkle Little Star.
    - In Task 2, they will add their own coordinated music and light algorithm to the loop algorithm they wrote in the previous lesson. They will use these algorithms for their Interactive Musical Horn prototype, which they will design in the next lesson.
3. Prompt students to begin working through the tasks. Circulate to provide support and ask Questions for Understanding.

Questions for Understanding to Pose to Students During Tasks

1. Explain to me what a variable is.
2. What is useful about variables?
3. Have you used variables in math before? What about like this: if you have an equation like x + x + 1, and the value of x is 2, what would the answer be? Does this word mean the same thing in math and in computer science?
4. Have you reached the step where you work with the two Wait Time regions? (if yes…) Why can you replace the three steps in the top Wait Time region with just one block? How could this kind of thing help a professional programmer?

Reflecting on Big Ideas (5 min)

1. Introduce each Big Idea, and ask volunteers to explain what they learned about each idea during their task activities:
    - Big Idea: Variables Are Data with Names
    - Big Idea: Variables Have A Value At Any Given Time
    - Big Idea: Variables Can Be Used to Change Several Parts of an Algorithm At One Time
2. Congratulate students on completing their Interactive Light Shows. Tell them they will present their plans on their prototypes in the next lesson.

Common Misconceptions to Address

- Different instances of a variable with the same name have different values.
- Variables only have names like X or Y; they cannot have names in words.
- Computers can process any amount of data simultaneously.
- A sensor has an input and an output.

Lesson 4: Programming the Interactive Show

Worksheet & Task Card: Programming the Interactive Musical Light Show

Worksheet Activity 1

Goal:
Understand a program that changes the speed of a ping-pong light effect when a light is shined on the sensor.

Interpreting the Light- Controlled Ping Pong Light Interactive plan

1. Launch the plan file and watch what happens to the LED strip. Change the amount of light on the sensor by shining a light on it, and by covering the sensor.
2. Notice that there are regions in this plan. Regions look like light-colored boxes that surround parts of the code. When you zoom out, you can see the name of each region.
    - Regions are a good way to keep code organized and label what an algorithm does.
3. Turn on Debugging in your plan file while the effect is running. Watch which blocks light up and when. Notice what happens to the blocks in each region.
4. With your group, study the plan file and complete the “algorithm story” below for each region:

Algorithm story of the region titled “**Ping Pong Effect**”:

1. When the Launch button is pressed, the Set Variable block creates a variable and gives it the name _____. This Variable starts off with a value of 0.
2. The Set Variable block then starts a For Loop block. An algorithm connects to the Loop Body pin. The first block in this algorithm is the Turn LEDs Off block.
3. All the blocks in that algorithm run until the Wait block is done. The Done wire of the Wait block connects back to the Next Index pin on the For Loop block. What this does is _____.
4. Each time the For Loop runs, the Current Index pin sends _____ to the Index Pin of the Set Indexed LED block. This tells the Set Indexed LED block to _____.

5. The first time this For Loop runs, the value of the Current Index is _____, and the value of the Index of the Set Indexed LED Color is _____. This makes the LED smartware _____.
6. The Set Indexed LED block then connects to three blocks in the “Wait Time” region. The Read Light Sensor Data uses the Light Sensor smartware to send an output of _____ to the Value pin of the Convert Value block.
7. Think back to what you learned about how Convert Value blocks work. If the light sensor measures light with a Lux value of 1000, the Convert Value block sends the Basic Math block an output with the value _____.
8. The Basic Math block has two inputs. The input titled A takes its value (1) and _____ the value that is the input titled B (this is the type of math the Basic Math block does). If the value of the input sent to the B pin is the amount that you answered in step 7 above, then the output of the Basic Math block is _____.
9. This makes the Wait Block that takes input from the Basic Math block wait for _____ seconds (in our example where the Light Sensor measures light with a Lux value 1000.)
10. The Wait Block has a wait time of 0.2 seconds, so it waits that long before it loops back to the Get Light Sensor Data block. This makes this part of the algorithm repeat more slowly, every 0.2 seconds.
11. Then the wire connecting back to the For Loop block makes the algorithm _____, but this time the value sent to the Set Indexed Color

block is _____. On the LED strip, this causes

_____.

12. The For Loop completes when

    _____.

    Then it connects to the second For Loop block.

13. This For Loop block connects to a Turn LEDs Off block and a Set Indexed LED block, just like the other part of the algorithm. But when this part of the algorithm runs for the first time, the difference is that _____

    _____.

14. The second time the For Loop runs the Loop Body, the Set Indexed LED block

    _____ on

    the LED Strip.

15. The Set Indexed LED block connects to the Get Variable Block. We will think about the value of this block later, after you complete the next Algorithm Story.

16. On the last time the For Loop runs, the Completed pin connects to the

    _____ pin of the first For Loop in the algorithm. This makes

    the LED Strip _____

    _____.

Algorithm story of the region titled "**Wait Time Variable**":

1. When the Launch button is pressed, it runs the algorithm in the blue Wait Time region. This region has the same title as the blue Wait Time region of the Ping Pong Effect algorithm. The Basic Math block in both of these regions output the SAME VALUE / DIFFERENT VALUES (circle one).

2. The output of the Basic Math block sends a value to the input pin of the Set Variable block. If the value of the Basic Math block output is 0.5, then the value of the WaitTime variable is _____.
3. Then the Wait block waits for 0.2 seconds before the Get Light Sensor Data block fires again and the algorithm starts over.

There is a Get Variable block in the Ping Pong Effect region. It leads to a wait block that gets an input of the value output of the Get Variable block. What does this block have to do with the Set Variable block in the Wait Time Variable region? _____

_____

_____

Worksheet Activity 2

Goal:
Add a Variable to the Ping Pong Effect Region and understand how it works.

Writing algorithms with variables

1. Zoom into the Wait Time region that is part of the Ping Pong Effect region.
2. We are going to delete all the blocks in this region. Click the tab at the top right corner of the region (it says “Wait Time”). Hit the backspace or delete key on your computer. All the blocks in that region should disappear.
3. We will replace those blocks with the variable titled WaitTime. Add a Get Variable block from the Utilities -> Variables menu.
4. In the drop down, select the title WaitTime.
5. Connect the Get Variable block to the Set Indexed LED Color block and the Wait block using purple wires.
6. Connect the output pin of the Get Variable block to the input pin of the Wait block.

7. Add a Display Value block and connect it to the Get Variable block with a purple wire. Connect the output pin of the Get Variable block to the input pin of the Display Value block. This lets you see the current value of the Get Variable block.
8. Launch this algorithm to your Cubit. Test the plan by shining the light on the light sensor and covering it with your hand. Does the plan work in the same way as it did before you changed the plan? Circle one: YES / NO, it works differently in this way:

   _____

9. While the plan is running on the Cubit, compare the Display Value blocks that are now in your plan. Their value is THE SAME / DIFFERENT.
10. With your group, describe in your own words how the WaitTime variable works in this plan, and why it is useful:

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

Save your plan. Congratulations!

Lesson 4: Programming the Interactive Show

Worksheet & Task Card: Programming the Interactive Musical Light Show

Task 2

Goal:

Add your own Interactive Light algorithm to the Interactive Musical Light Show you will play on your prototype that you built in the previous lesson. This will finish your Interactive Musical Light Show!

Write your own musical light algorithm using variables

1. Load the plan you have been working on and place the first block of your music and light algorithm like you did in this example by placing a Play Sequence block.
2. Plan and write your own algorithm that uses variables to respond to the data from the light sensor smartware, similar to what you saw in the Light-Controlled Ping-Pong plan. Use what you learned about the buzzer smartware to add sounds or music to this algorithm. Test and revise until it works as you planned.
3. Use what you have learned about loops to make your light show repeat in an infinite loop.
4. Congratulations, you have now completed your Interactive Musical Light show! **BE SURE TO SAVE YOUR PLAN FILE!**
5. If you have more time, add more algorithms to your show. Make sure your show still repeats in an infinite loop.
6. Tip: Your code may have a lot of wires that look confusing. You can “clean up” your code in Workshop by making “shortcuts” called GoTos. Professional programmers keep their code simple so it is easier to read. You can too!
7. Create a GoTo by hovering your cursor over the wire making your two algorithms into an infinite loop. A +GoTo icon will appear.
    a. Click the icon and give your GoTo a label. Press OK.
    b. Notice that the wire will turn into two GoTo arrows. When the Cubit gets to the GoTo arrow at the end of your algorithms, it will “go to” the matching arrow with the same name.

cubit

STEAM Education

Lesson 5: Prototype Gallery Walk

Standards and Conceptual Snapshot

Unit Theme:

How Can We Design An Interactive Musical Light Show?

Big Idea:

- Designers and Engineers Use Feedback To Improve Designs

Essential Question:

- Why do designers show their designs to others?

Learning Objectives - Students will be able to...

- Present designs and ask questions to solicit feedback
- Explain computer science concepts used in designs
- Evaluate the success of designs based on feedback

Design Thinking Connection:

- **Explore:** Gather Information
- **Imagine:** Brainstorm Solutions
- **Prototype:** Troubleshoot Issues
- **Iterate**: Provide and Receive Feedback, Identify Revisions, Replan
- **Communicate**: Create and Give Presentations, Create Explanatory Diagrams, Get New Perspectives, Document and Incorporate Insights

K-12 Computer Science Framework Standards:

Practices	Core Concepts	Crosscutting Concepts
Testing and Refining Computational Artifacts	Algorithms	Communication and Coordination
Creating Computational Artifacts	Control	Human–Computer Interaction
	Devices	System Relationships
	Hardware and Software	
	Program Development	
	Troubleshooting	

Important Terms:

1. Iteration
2. User Testing

cubit

STEAM Education

Lesson 5: Prototype Gallery Walk

Setup, Preparation and Clean Up

Materials:

- 1 - Cubit Computer Science Kit per student group
  - 1 - Cubit Controller
  - 1 - Light Sensor
  - 1 - Color LED Strip
  - 1 - Buzzer
- 1 - USB Power Cable for each Cubit Controller or 4 - AA Batteries
- Glue or gluesticks, different varieties of tape, or other adhesive materials
- Student Prototypes built in previous lessons
- 1 - Flashlight or other handheld light source per group
- 1 - copy Progress as Programmers Worksheet per student group
- 1 - copy Prototype Feedback Worksheet per student group
- Students' completed Design Planning Worksheets from previous lesson
- 1 - (optional) timer or online/phone app,to track presentation time for different groups

Resources:

Copymasters:

- Progress as Programmers Worksheet
- Prototype Feedback Worksheet

Cubit Student Quick Guide
Cubit Computer Science Skills Guide
Rubrics (Notebook, Collaboration) and Brainstorming Mindset Guide
Lesson 1 Student Challenge Handout

Preparation before class:

1. Make copies of the Progress as Programmers and Prototype Feedback Worksheet, one per group.
2. Ensure there is sufficient space for students to display their designs and for students to travel to view different prototypes.
3. Make sure student prototypes are available to rebuild.
4. Make sure students' completed Design Planning Worksheets are available to distribute.
5. Write the following Gallery Walk Roles on the board:
   - Presenter Role
     1. Say who your target users are, the purpose of your project, and the specifications.
     2. Launch the show and show how it works.
     3. Explain how your code contains each computer science idea.
     4. Write down revision feedback.

- Viewer Role
    1. Act as if you are the target users.
    2. Ask questions about each computer science idea.
    3. Give feedback on the design.
    4. See as many prototypes as possible.

6. Consider what to do with student prototypes after class, such as dispose or save. If desired, extend this unit by saving prototypes and adding a final prototype revision day based on feedback from this lesson, or have students add new algorithms to their programs.
7. If desired, prepare for Assessment Opportunities (see end of Lesson Detail). If documenting student explanations in the Gallery Walk, consider preparing a clipboard with a class roster and space to write evaluations.

Agenda

1. Warm-up: Progress as Programmers (5 min)
2. Theme and Essential Questions (5 min)
3. Preparing to Present Using Regions (10 min)
4. Prototype Gallery Walk (40 min)

Clean up

1. Halt Program, Clear from Cubit, and Save Cubit Workshop Plan File to Drive (if desired)
2. Disconnect from Cubit Controller and Exit Workshop
3. Turn off or disconnect power
4. Put away Kit
5. If necessary, return 4 AA Batteries to the designated space for storage and charging
6. Save or dispose of prototypes.

cubit

STEAM Education

Lesson 5: Prototype Gallery Walk

Lesson Detail

Duration: 60 minutes

Essential Question:

- Why do designers show their designs to others?

Learning Objective:

- Present designs and ask questions to solicit feedback
- Explain computer science concepts used in designs
- Evaluate the success of designs based on feedback

Differentiation

- Stretch - Custom Code Blocks, Extend with another Revision Cycle, Give Presentations to Younger Students
- Scaffold - Vocabulary Cards, Student Guides, Provide Start Files

Hook/Warm-up: Progress as Programmers (5 min):

1. Pass out a Progress as Programmers Worksheet to each group.
2. Prompt students to reflect on what they have learned by writing their ideas about each of the computer science concepts in the first column of the worksheet. Instruct them NOT to move on to the second column.

Discuss Theme and Essential Question (5 min)

1. Remind students of the Theme: "How Can We Design An Interactive Musical Light Show?". Tell students they will now present their own designs and prototypes of the light and music show to each other, and get feedback from their peers on how well it meets the specifications.
2. Introduce the Essential Question: "Why do designers show their designs to others?"

Preparing to Present using Regions (10 min)

1. Prepare students for the presentation task. Say,
    - "You will be presenting your prototypes to other people. You will also be showing other people your plan in Workshop, and explaining how your code shows the concepts you have learned We can make it easier to show people the parts of our program by organizing our code. Professional programmers also organize their code to make it easier to show other people. In Workshop we can organize our code visually using regions. For example, to show that your code covers the concept of loops, you can make a region titled "Loop" around the part of your code that contains a loop.

    - “Part of your team will go view the prototypes of other groups, and the other part of your team will stay with your prototype and explain it. This means everyone on your team needs to know how to explain your software. The regions will help you to show others which part of your program shows the concept.
2. Pass out Cubit kits and direct students to open Workshop.
3. Direct students to fill out the second column of the Progress as Programmers worksheet, creating regions as they go.
    - “Work together so everyone knows how to talk people through the program. Use the regions to keep organized. Write down the title and color of your region

Questions for Understanding to Pose to Students During Activity and Debrief

1. Why does the worksheet say you do not need to create a region for the “Algorithm” concept? How many regions could you make for that concept -- Zero? One? Many?

Prototype Gallery Walk (40 min)

1. Read Gallery Walk Roles from the board.
2. Distribute each group’s Design Planning Worksheets. Explain that students will refer to this to remember their user audience and purpose.
3. Hand out the Prototype Feedback Worksheet to each group.
4. Divide groups as evenly as possible into Viewers and Presenters.
5. Tell students they have about 5 minutes to assemble their prototype and prepare to launch their Workshop plan. Consider giving students a 1- or 2- minute warning.
6. Begin the Gallery Walk by asking Viewers to distribute themselves among the prototypes of different groups.
    - Consider setting an 15-minute timer to ensure all students have equal time in each role.
    - Circulate to pose Questions for Understanding. Periodically remind Viewers to switch groups and try and see as many prototypes as possible.
7. After 15 minutes, ask Viewers to return to their own prototypes and switch roles. Circulate to pose Questions for Understanding and remind Viewers to switch groups and try and see as many prototypes as possible. Reset your timer, if using.
8. After about 15 more minutes, ask Viewers to return to their groups, and briefly discuss the revisions they wrote down.
9. Debrief with the Big Idea: “Designers and Engineers Use Feedback To Improve Designs”.
10. Congratulate students on their successful presentations and programming. Encourage growth mindset by giving specific feedback on students’ effort and persistence, and emphasize that trying your best and exploring new ideas is how to gain new skills, like becoming a professional programmer, designer, or engineer.
11. Have students disassemble their prototypes and clean up.

Questions for Understanding to Pose to Students During Prototype Gallery Walk

1. Show me and explain which part of your Workshop plan demonstrates one of these concepts.
2. Explain the concept this part of your code shows as if I were a younger student who didn't know computer science.
3. What knowledge did you have to use to program this?
4. Why did you connect the wires the way you did?

Common Misconceptions to Address in this Lesson

- Presenting prototypes is just for show.
- Code does not need explanation.
- Programmers just write code, they do not add notes or other ways to help others read the code.
- Programmers work by themselves and don't look at other people's code.

Assessment Opportunities (Optional)

- **Groups**: Review students' ideas in the first column of the Progress as Programmers worksheets
- **Individual:** During the Gallery walk, listen to students' explanations of how their code demonstrates computer science concepts. You may wish to document students'

cubit

STEAM Education

Progress as Programmers Worksheet

Warm-up: Reflect on your progress as programmers! Fill out the FIRST column with what you have learned about each computer science concept. Do NOT fill out the second column during the Warm-Up -- wait until the next activity.

Presentation Planning: Identify these concepts in your program to prepare for the prototype presentation.

Concept	**Warm-Up**: What we've learned	**Presentation Planning:** How our prototype program demonstrates this concept. (Create a region in your plan if directed)
Algorithms		No Region Needed!
Loops and For-Loops		Region Name and Color:

Go on to the next page.

Concept	**Warm-Up**: What we've learned	**Presentation Planning:** How our prototype program demonstrates this concept. (Create regions if possible)
Conditionals		Region Name and Color:
Comparators		Region Name and Color:
Variables		Region Name and Color:
Data Inputs and Outputs		Region Name and Color:

Prototype Feedback Worksheet

Write down feedback from the people viewing your presentation.

Feedback on Workshop Plan

Feedback on Prototype Design