Seeker®

BY PITSCO EDUCATION

TEACHER'S GUIDE

UNIT 3

Unit Overview

In this Unit, Sophia introduces students to Tronic Theater Company, a local stage production group who wants to incorporate Seeker into their plays and musicals to provide light and sound effects. The tasks students will complete involve creating human-computer interfaces to control Seeker's pixels and produce sound and music. Students are introduced to using conditionals to run a specific algorithm when a certain condition is true or when a certain event occurs. Students also learn about using procedures and functions to decompose code into smaller algorithms.

In Activity 1, students meet Hannah O'Connor, an audio and light engineer for Tronic Theater Company. She wants to use Seeker to create mobile lighting effects on stage. Students will learn about using conditionals to create human-computer interaction. To convince Hannah that Seeker is up to the job, students will program Seeker to perform different light effects using Seeker's pixels when Seeker's buttons are pressed. They'll use nested conditionals to program more advanced light controls using button combinations. The activity ends with students programming Seeker to create lighting effects for a thunderstorm scene in the upcoming musical that Tronic will be producing.

Activity 2 centers around Seeker's ability to produce sound and music. They meet Tarik, a mechatronics technician for Tronic who is interested in using Seeker to act out certain parts in their productions. But, before Seeker can act, students learn how to create sound effects and music using Seeker. They learn the basics of how sound travels as well as some basic music theory. Using procedures and functions, students learn how to make code more efficient, clean, and readable as they decompose programs into subroutines. Using Seeker's buttons, they program Seeker to serve as a musical instrument; each button plays a different note in the musical scale, allowing them to play some simple songs. The activity concludes with a mechatronics challenge where students either write a scene or recreate one from another play or movie and program Seeker to act it out on a stage.

Careers

- Sound and Light Engineer
- Mechatronics Technician

Context and Setting

A local theater company wanting to use Seeker to provide sound and light effects as well as mechatronic movement for stage productions.

Computer Science and Technology Concepts

- Conditionals
- Nested Conditionals
- Forever Loop
- Functions and Procedures
- Events
- Controlling Lights and Sound on Seeker
- Human-Computer Interaction
- Frequency of Light and Sound

Unit Prep Instructions

- Before starting this Unit, make sure the Seekers are assembled in robot mode as they were in Units 1 and 2. Students should not need to do any robot modifications for this Unit.
- This Unit focuses on controlling Seeker's pixels and making sound and music. Because of this, there is much less movement than in Unit 2. Most tasks can be completed right at the students' desks.

However, there are a few tasks such as Mechatronics Movement and the challenge in Activity 2 that will require a space for Seeker to move around. For these activities, create a testing area in the classroom that represents a stage. This can be as simple as a square taped off on the floor. For more information, see the Actor Avatar and Mechatronics Monologue challenge brief documents.

- Activity 2 focuses on sound and music. With multiple Seekers making noise at the same time, it can get loud in the classroom. Seeker does not have a volume control; however, its volume can be reduced by placing a piece of tape over the speaker. However, doing this can make it difficult to hear some of the lower frequencies.
- Consider giving student pairs different roles. For example, for Activity 1, have one student oversee hardware while the other student oversees software. Then, switch roles for Activity 2.

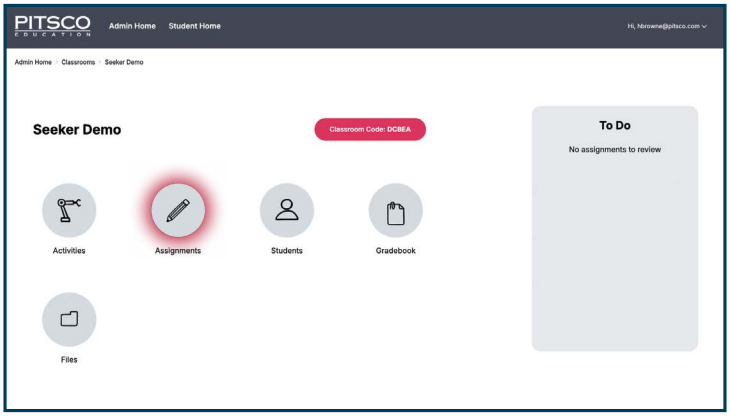
Unit Discussion Questions

- Should robots be used for entertainment? Why or why not? If so, in what areas?
- If robots were used in performing arts such as music, dance, and theater, in what aspects would robots be better than people? In what aspects would people be better than robots?
- How are robots changing the way we experience live performances right now? What about in the future?
- Can robots be creative? Why or why not?
- Can robots be expressive? Why or why not?
- Would you go watch a play if robots were the lead actors instead of human actors? Why or why not?
- How can robots help human performers rather than replace them?

Unit Assignments

The following assignments are available to be used with Unit 3 content. When you set up your classroom, the points possible default to zero points. If you choose to use an assignment, you will need to set the number of points the assignment is worth. This can be done in the Assignment section when managing your specific classroom.

When a student submits an assignment, it should show up as a To Do item for you to review and assign points.



Activity 1

- Page 3 – Submit Blockly code for controlling the color of Seeker's pixels: U3A1T1-LightControl.
- Page 7 – Submit Blockly code for controlling the brightness of Seeker's pixels: U3A1T2-BrightnessControl.
- Page 8 – Submit Blockly code for controlling the brightness and color of Seeker's pixels: U3A1T4-TheaterLights.
- Page 11 – Submit Blockly code for creating different theater lighting effects: U3A1T5-LightEffects.
- Page 11 – Submit Blockly code for creating the storm scene: U3A1T6-StormScene.

Activity 2

- Page 5 – Submit Blockly code for the "Charge" song: U3A2T2-Charge.
- Page 7 – Submit Blockly code for the "Für Elise" song: U3A2T3-FurElise.
- Page 9 – Submit Blockly code for playing different sound effects: U3A2T4-SoundEffects.
- Page 10 – Submit the updated Blockly code for playing different sound effects: U3A2T4-SoundEffects.
- Page 14 – Submit Blockly code for the mechatronics program: U3A2T6-Mechatronics.
- Page 15 – Submit Blockly code for completing either the Actor Avatar or Mechatronics Monologue challenge.

Global Activities

- Career Activity 1: Career Research – Research a career of their choice and submit information about the career in text field.
  - Consider having students research the careers mentioned in this Unit to make a stronger connection between real careers and how robotics is impacting those careers.
    - Sound Engineer
    - Light Engineer
    - Mechatronics Technician
- Career Activity 2: Career Priorities – In a text field, list priorities for choosing a career, three possible career choices, advantages and disadvantages of each career, and how the career priorities are met with each career.
- Career Activity 3: Interest Profile – Complete an interest profile and record results from the profile in a text field.

Icon Key

- **Code Saved to Student Profile**
- **Code Testing Task**
- ***Innovation Notebook* Task**
- **Engineering Task**
- **Guided Programming Task**
- **Exploration Task**
- **Sequencing Task**
- **Debugging/ Troubleshooting Task**
- **Challenge Task**

Guide to Activity 1: Lighting the Stage

Page 1 – Hannah the Light and Sound Engineer

Students meet Hannah O'Connor who works as an audio and light engineer for Tronic Theater Company. After explaining about her career, Hannah expresses her desire to use Seeker on stage to create light and sound effects for their stage productions.

Tasks to Complete

- Discover how robotics relates to theater productions, sound, and audio engineering.

Teaching Tips

- Before starting this Unit, make sure Seeker robots are in robot mode. There shouldn't need to be any changes made from how the robots were configured in Units 1 and 2.
- This Unit will involve Seeker performing light and sound functions on a simulated theater stage. Seeker's movement should be fairly limited. A table or desk should be able to work as the performance stage.
- Consider having students research the careers of a sound or light engineer. Career activities are provided as assignments that can be assigned through the teacher portal.

Notes:

Page 2 – Conditionals

Lucas explains how working with Tronic Theater Company might not be a big moneymaker for Adaptibots but that there is other value in partnerships besides turning a profit, such as supporting the arts, company exposure, and working on projects that you're passionate about. Guided by Lucas, students then begin a program to control Seeker's pixels using buttons presses. Lucas explains what a conditional is in programming and how the **if block** is used to create a conditional for when a button is pressed. Lucas also explains how the **set pixel color block** is used to change a pixel's color and brightness.

Tasks to Complete

- Begin a program to control Seeker's pixel LEDs using button presses.
- Determine what a conditional is and how to create a basic conditional using the **if block**.
- Investigate how to change Seeker's pixel colors and brightness.
- Test the program using the simulator and with a real Seeker robot.

Teaching Tips

- Discuss conditionals and how we use them in our everyday lives. Have students give examples of conditionals from their lives.
- While it shouldn't be a major issue, Seeker's pixels are one of the things that drains the battery faster. Battery life can be extended by using smaller brightness values and not leaving the pixels on for extended periods of time. The default brightness is 50%.
- When using the simulator to test their code, students can use the "Board View" option to zoom in on the controller.

Notes:

Page 3 – Changing Colors

Students add to the program so that Seeker displays a different pixel color when each button (A, B, C, and D) is pressed. This allows Hannah to use Seeker to set different themes or moods for different scenes in a play. To program this behavior, the program requires creating four conditional statements in the code. Students test their program in the simulator and with their Seeker robot.

Tasks to Complete

- Program Seeker to light up its pixels a different color when each button is pressed.
- Test the program in the simulator and with Seeker.
- Save the program as U3A1T1-LightControl.

Teaching Tips

- If time permits, discuss how colors are connected to our emotions and what colors would be appropriate for different kinds of scenes in a play or musical performance (examples: conflict, love/emotional, suspense, dialogue, action, peaceful, and so on).
    - Consider bringing in a theater teacher to help lead the conversation.

Notes:

Page 4 – Human-Computer Interaction

Lucas explains that human-computer interaction is the study of how humans interact with computing devices and robots. Students use their *Innovation Notebook* to brainstorm how the light control program could be improved to give more lighting options.

Tasks to Complete

- Brainstorm ways to improve the light control program to give more lighting options.

Teaching Tips

- See page 13 for the *Innovation Notebook* Answer Key for this task.

Notes:

Page 5 – Controlling Pixel Brightness

Lucas discusses how flowcharts can be used to represent the logic of a program. Students load an example program that uses nested conditionals to not only affect the color of the pixels but their brightness as well. Buttons A, B, C, and D change the pixel color. However, when the Start button is held down, these same buttons affect the pixel brightness. Students test this program to identify bugs and make improvements to the program.

Tasks to Complete

- Test and identify bugs in a given program that allows a user to change pixel color and brightness using the buttons on the controller.
- Record bugs and ideas for improvement in the *Innovation Notebook*.

Teaching Tips

- This is the first time nested conditionals have been discussed. A nested conditional is a conditional within another conditional. A real-world example of nested conditionals could be if it is a weekday, then if it is not a holiday or summer break, then go to school . . . otherwise, stay home.
- Sometimes, it is easy to get nested conditionals and compound conditionals mixed up. A nested conditional allows for multiple levels or a hierarchy of decision-making. A compound conditional evaluates multiple conditions at once using and/or to execute a command or algorithm.
- See page 13 for the *Innovation Notebook* Answer Key for this task.

Notes:

Page 6 – Fixing Bugs

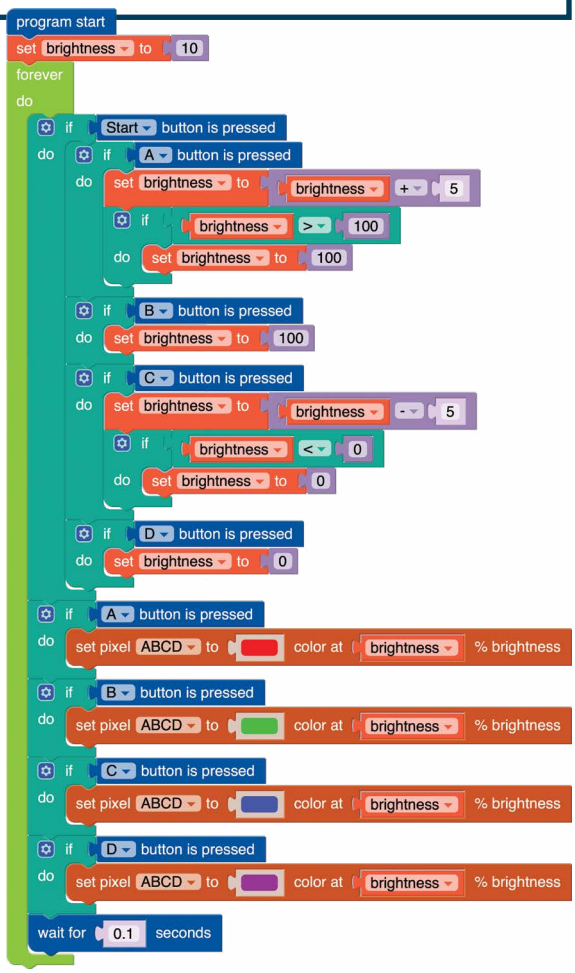
Lucas guides students in fixing some of the bugs in the light control program. This includes slowing down the rate that the brightness changes and limiting the brightness to a max of 100%. Using what they learn from Lucas's guidance, students attempt to add similar functionality on their own that limits the minimum brightness to 0%. After addressing the bugs in the program, students test their fixes in the simulator and with Seeker.

Tasks to Complete

- Fix three bugs in the light control program.
- Test the fixes to the program to better allow a user to control the brightness of Seeker's pixels.

Teaching Tips

- Limiting the brightness variable between 0 and 100 requires three levels of nested conditionals. The first is for when the Start button is pressed, and the second is for when Button A or C is pressed to increase or decrease the brightness variable. The final nested conditional is for when the brightness variable is greater than 100% or less than 0%. It is easy for students to get lost in the logic of these nested conditionals. Remind them to evaluate each nested conditional from outside to inside.



Notes:

Page 7 – Light Exploration

After saving the light control program, Lucas and Nova divide up responsibilities. Lucas works on solving one last nagging bug in the light control program while the students start exploring other lighting effects. A series of exploration objectives are presented, and students spend about 20 minutes trying to accomplish as many of the objectives as possible. If students finish and time remains, students can experiment with their own lighting effects. Students check objectives off in their *Innovation Notebook* as they complete them.

Tasks to Complete

- Save the program to their profile as U3A1T2-BrightnessControl.
- Explore other pixel blocks and lighting effects.

Teaching Tips

- Sophia tells the students to spend about 20 minutes exploring other lighting effects and pixel blocks; however, depending on your students' abilities, you might want to give more time for them to do some exploring on their own after accomplishing the objectives in the *Innovation Notebook*.
- Here are the exploration objectives that are listed for students to try:
  - Explore the **blink block**. Change the program so the pixels blink three times when a color is selected rather than remain solid.
  - Explore the **show animation block**. Instead of changing pixel colors, set a different animation to play for five seconds when Buttons A, B, C, and D are pressed.
  - Set the color to a random color when Button A is pressed.
  - Instead of having every pixel display the same color, change the program so different color combinations shine when the different buttons are pressed. For example, you could have pixels one and three shine blue while pixels two and four shine red. Create a different color combination for each button.
  - In the current program, the brightness of the pixels changes by five percent when the Start button is held down and Button A or C is pressed. Create a variable to represent this increment. Set the value of the increment to two percent to give the user more brightness options. Don't forget to use **variable blocks** when the brightness is increased or decreased.
- It isn't necessary for students to complete every objective listed in the *Innovation Notebook*.

Notes:

Page 8 – Comparing Programs

While students have been exploring other light effects and pixel blocks, Lucas has solved the remaining bug in the light control program. Students use the Load Code button to see both the original program and Lucas's new program. Students compare the two programs and record differences in their *Innovation Notebook*. After recording their observations, they delete the original program. They save the new bug-free program and then test it in the simulation and with Seeker.

Tasks to Complete

- Compare the original program and the new bug-free light control program to identify differences.
- Record differences in the *Innovation Notebook*.
- Save the new version of the program as U3A1T4-TheaterLights.
- Test the new program in the simulator and with Seeker.

Teaching Tips

- Lucas's new program is similar except for two main differences. These will be explained on page 9.
- See page 14 for the *Innovation Notebook* Answer Key for this task.

Notes:

Page 9 – Conditional Structures

Students use the Load Code button to reload both the old and new light control program. Lucas explains the two main differences between his new program and the old. The first is that the color is tracked using a variable and updates at the end of the loop, no matter whether a button is pressed or not. The other is that he used if/else-if conditional structures. Lucas goes on to explain if/else and if/else-if structures and how to create them in the app. Lucas explains that, with if/else-if conditionals, the order matters and asks students to record what would happen if the conditions were rearranged so that checking to see if the Start button was pressed was the last condition rather than the first in the program.

Tasks to Complete

- Understand how to create if/else and if/else-if conditionals.
- Record in the *Innovation Notebook* what would happen if checking whether the Start button was pressed was the last condition rather than the first in the if/else-if conditional structure.

Teaching Tips

- To change the structure of an **if block**, students click the gear icon to bring up the conditional builder tool. As they drag an **else-if block** or **else block** under the **if block**, the conditional structure automatically updates.
- If students have trouble answering the question in their *Innovation Notebook*, they can try to rearrange the order of the conditions in the **if block** and test it in the simulator or with Seeker to see what happens. They can always use the Load Code button to get Lucas's fixed program back again.

Notes:

Page 10 – Puzzle: Light Effects

Sophia wants to create one more demonstration for the theater that shows different lighting effects when Seeker's buttons are pressed. She provides students with pseudocode to work from as well as all the blocks for the program. Students need to create the logic structure and then put the given blocks in the correct order to complete the program according to the pseudocode. After putting all the blocks together, they test their code in the simulation and with Seeker.

Tasks to Complete

- Create the correct conditional structure for the program according to the provided pseudocode.
- Place the given blocks and algorithms in the correct sequence and locations in the program.
- Test the light effects program in the simulator and on Seeker.

Teaching Tips

- When the Load Code button is clicked to bring up the blocks for this task, some of the blocks are already put together into the correct sequence or algorithm. To move a series of connected blocks, grab the top block and move it to where it needs to go. The other blocks should stay connected to it allowing you to move the entire sequence.
- If students mess up the provided sequences or algorithms, they can click the Load Code button again to start over.
- The logic for this task could be done in multiple ways. But, according to the provided pseudocode, an if/else-if conditional structure should be used. Discuss with students why this is the best option for this program. An if/else-if logic structure evaluates each conditional in order until a true condition is found. Then, it skips the rest of the conditionals because only one condition can be true in the structure. This makes the program more efficient.

Notes:

Page 11 – Storm Scene

Students are given a final task for this activity that is a mini challenge. The challenge is to program Seeker's pixels to display a light show for a storm scene in an upcoming musical at the theater. Pressing different buttons on Seeker should cause Seeker's pixels to show different parts of a storm. As students solve the mini challenge, they test their code in the simulator and with Seeker. Students are also told they will need to make a short presentation about their program and demonstrate. This presentation should mimic a pitch to the theater company for them to purchase and include Seeker in their productions.

Tasks to Complete

- Save the previous program as U3A1T5-LightEffects.
- Create a new program for a storm scene light show that is controlled by Seeker's controller buttons.
- Test the storm scene program in the simulator and with Seeker.
- Save the storm scene program as U3A1T6-StormScene.
- Develop a short presentation that explains how the program works. Record talking points in the *Innovation Notebook*.

Teaching Tips

- There are multiple ways this mini challenge could be solved. However, student solutions should follow these design criteria:
  - The parts of the scene include a bright sunny day, the storm rolling in, lightning flashes, the storm fading and returning to blue sky, and then the scene ending by fading out the lights.
  - The scene should be controlled by button presses.
- Students are given these topics to discuss when presenting their storm scene program:
  - Explain how the program works using Seeker's buttons.
  - Explain what kind of conditional structure you choose to use and why.
  - Explain your process for developing the storm scene program.

- Discuss the most difficult part of programming the storm scene.
- Discuss what other actions Seeker could do to make the storm scene even more realistic.

- At this point, presentations do not need to be highly polished; they should be two to three minutes at most. However, it is a chance for students to practice and improve their public speaking. Encourage every student in the group to have a role in sharing something.
- Students can use their *Innovation Notebook* to jot down talking points.

Notes:

Page 12 – Storm Scene Presentation

The Adaptibots team gathers at the theater to present their work to Hannah and the Tronic Theater Company. After reviewing the Seeker programs developed throughout the activity, students demo their Storm Scene program and then deliver their presentations on it.

Tasks to Complete

- Demonstrate the storm scene program on Seeker.
- Deliver a short two- to three-minute presentation about the program, how it works, and how it was developed.

Teaching Tips

- Presentations should be two to three minutes in length.
- Before beginning student presentations, remind students to use good presentation skills such as eye contact, body language, speaking rate and clarity, engagement, avoid filler words, and so on.
- Provide constructive feedback on their presentations to help students build their presentation skills. As students' progress through other Seeker Units, they will have more opportunities to present solutions to challenges, some of which will require them to have more polished presentations. The goal with this task is not necessarily to assess their current presentation abilities but to help them improve their skills for the future.
- If time permits, allow a few questions to be asked by the audience.

Notes:

Page 13 – Beyond Lights

The activity ends with the Adaptibots team reaching a deal with Hannah and the Tronic Theater Company to provide Seeker robots for lighting. They also make plans to continue discussions on how Seeker can be used for audio and mechatronics to enhance their productions.

Tasks to Complete

- Complete the activity.

Teaching Tips

- Here are a few discussion topics to consider if time permits:
  - What are the pros and cons of using robots in performing arts such as theater, dance, and music?
  - Are there limits as to how robots should be used in performing arts?
  - Can a performance still be considered "art" if it is performed by a robot?
  - How is other technology impacting performing arts?

Notes:

Activity 1 *Innovation Notebook* Answer Key

Human-Computer Interaction

What features could be added to the program to give Hannah more lighting control and options? Spend some time brainstorming ideas and recommending improvements to the light control interface. Record your ideas in the following space.

Ideas could include:

- Turning the pixels off.
- Turning the pixels on only while a button is pressed. When the button is released, the pixels turn off.
- Changing the brightness of the pixels.
- Flashing or blinking the pixels.
- Allowing for more color options by using button combinations.

Controlling Pixel Brightness

Identify and list bugs or things that can be improved in the program for changing pixel brightness.

Bugs and improvements could include:

- The brightness changes too fast when holding down Button A or Button C.
- Changing the brightness also changes the color of the pixels.
- The brightness can be increased beyond 100%.
- The brightness variable can be decreased below 0%.

Light Exploration

Check off these exploration objectives as you complete them:

- ❏ Explore the **blink block**. Change the program so the pixels blink three times when a color is selected rather than remain solid.
- ❏ Explore the **show animation block**. Instead of changing pixel colors, set a different animation to play for five seconds when Buttons A, B, C, and D are pressed.
- ❏ Set the color to a random color when Button A is pressed.
- ❏ Instead of having every pixel display the same color, change the program so that different color combinations shine when the different buttons are pressed. For example, you could have pixels 1 and 3 shine blue while pixels 2 and 4 shine red. Create a different color combination for each button.
- ❏ In the current program, the brightness of the pixels changes by five percent when the Start button is held down and Button A or C is pressed. Create a variable to represent this increment. Set the value of the increment to two percent to give the user more brightness options. Don't forget to use **variable blocks** when the brightness is increased or decreased.

Comparing Programs

Compare the differences between the original light control program and Lucas's new program. What specific differences do you see? Record your observations in the following space.

Differences can include:

- There is a color variable that stores a value of 1, 2, 3, or 4 depending on what button is pressed.
- The pixel color is set at the end of the loop based on the color variable even when a button isn't pressed.
- Many of the conditionals have an "else-if" structure rather than just an "if" structure.

Conditional Structures

What do you think would happen if we rearranged the conditions in the Theater Lights program and moved the Start button being pressed to the last else-if condition rather than the first condition? Record your thoughts in the following space.

In else-if conditional structures, only one set of commands is executed. The program looks for the first condition that is true and executes the commands associated with that condition. Then, it stops checking the rest of the conditions. The order of the conditions matters because of this. So, if the condition for the Start button being pressed were moved to the last condition, it would never be executed because another button, either A, B, C, or D, is always pressed with it. You could change the pixel color, but you would never be able to change the pixel brightness.

Storm Scene

Use the following space provided to make a few notes on talking points for your storm scene presentation.

Notes can include talking points around these topics that were provided to students or their own talking points:

- Explain how the program works using Seeker's buttons.
- Explain what kind of conditional structure you chose to use and why.
- Explain your process for developing the storm scene program.
- Discuss the most difficult part of programming the storm scene.
- Discuss what other actions Seeker could do to make the storm scene even more realistic.

Guide to Activity 2: Sounding Off

Page 1 – Sound Exploration

Students begin this activity by exploring Seeker's sound and musical capabilities. They spend 20 minutes creating their own programs using **play tone blocks** and **play musical note blocks**. Students can investigate how the frequency of a tone affects the sound that is heard.

Tasks to Complete

- Explore how to use the blocks in the Sound category to play different pitches, musical notes, and sound effects.

Teaching Tips

- Students are given ideas to explore using the blocks from the Sound category:
  - Try some different sound effects to see what they sound like.
  - Use **play tone blocks** to play different pitches. The default pitch is 1,000 hertz, but humans can hear pitches between 20 hertz and 20,000 hertz.
  - Experiment by playing different musical notes. Try putting multiple **play musical note blocks** together to see if you can make a song or jingle.
- Seeker's speaker has limits to the range of sound it can produce. Frequencies below 40 hertz or above 15,000 hertz might sound distorted.

Notes:

Icon Key

- **Code Saved to Student Profile**
- **Code Testing Task**
- ***Innovation Notebook* Task**
- **Engineering Task**
- **Guided Programming Task**
- **Exploration Task**
- **Sequencing Task**
- **Debugging/ Troubleshooting Task**
- **Challenge Task**

Page 2 – Mechatronics

Students meet Tarik Amad, a mechatronics technician for Tronic Theater Company, and learn about how mechatronics is used in theater productions. Tarik and Hannah set the stage for how students will be programming Seeker to play music and sound effects as well as act out scenes.

Tasks to Complete

- Understand the goals and context of the programming work that will be done in this activity.

Teaching Tips

- Make sure students have completed Unit 3 Activity 1 before starting this activity.
- This activity has a heavy emphasis on the science of sound and music including some basic music theory.

Notes:

Page 3 – Sound and Frequency

Sophia explains some of the science of sound including how it travels as longitudinal waves and is affected by frequency. Students then load a program to explore the relationship between frequency and pitch. Students modify the program three different times to help them explore this relationship. For each program, students record their observations in their *Innovation Notebook*.

Tasks to Complete

- Investigate the relationship between a sound's frequency and its pitch and record observations in the *Innovation Notebook*.

Teaching Tips

- The volume of Seeker's speaker can be reduced by placing a piece of tape over the speaker. This can help reduce the noise level in the classroom if multiple Seekers are being used. However, it also makes it harder to hear some lower frequencies.
- Here is a brief description of the four programs students use to investigate the relationship between frequency and pitch.
  - Program 1 – the frequency doubles every two seconds
  - Program 2 – the frequency halves every two seconds
  - Program 3 – the frequency increases by 10 every 0.1 second
  - Program 4 – five different frequencies of the student's choosing are played for 1 second each
- See page 27 for the *Innovation Notebook* Answer Key for this task.

Notes:

Page 4 – Musical Notes

Sophia explains how music is produced through Seeker's speaker by a vibrating cone that is controlled by an electromagnet. She also explains some basic music theory including how musical notes are named, organized into octaves, and are tied to specific frequencies. Students are then guided to build a program that plays a simple song. Students add **play musical note blocks** and set the notes and durations according to a table provided in the content.

Tasks to Complete

- Understand some basics of how musical notes are named and organized.
- Create a program that plays a simple song using Seeker.
- Test the program and try to identify the name of the song.

Teaching Tips

- The program students create should play the song "Charge," which is often heard at sporting events.
- Notes in the **play musical note block** are named according to their note letter followed by the octave number.

Notes:

Page 5 – Für Elise

After saving the "Charge" song program, Sophia continues to discuss some basic music theory including note durations and the number of beats each note gets. Students use the Load Code button to load a program that plays the song "Für Elise" by Beethoven. Sophia explains how variables are used in the program to set the length of different note durations such as whole notes, half notes, quarter notes, eighth notes, and so on. She also explains how changing the tempo variable changes the length of each type of note affecting the speed of song. Students upload the program to Seeker and run the program.

Tasks to Complete

- Save the "Charge" song program to their profile as U3A2T2-Charge.
- Investigate note durations and how they can be controlled in programming using variables.
- Load and play the "Für Elise" song program.

Teaching Tips

- The "Für Elise" song program has hundreds of blocks and can look overwhelming at first. Remind students that it is just an algorithm that plays one note after another.
- Because the "Für Elise" song program has so many blocks, it can take quite a while to load through Bluetooth. If possible, have students load this program using the USB cable.

Notes:

Page 6 – Decomposing the Song

Sophia discusses the importance of making programs efficient and decomposing them into parts allowing the programmer to focus on one part at a time. She also introduces how programmers use procedures and functions to break programs down into specific subroutines or algorithms. Students study the "Für Elise" song program looking for ways to decompose it into parts. They record their ideas for decomposing the song in their *Innovation Notebook*.

Tasks to Complete

- Understand how procedures and functions can be used to break a program down into smaller parts and to make code more efficient.
- Record ideas for decomposing the song in the *Innovation Notebook*.

Teaching Tips

- Many people use the words *procedure* and *function* interchangeably in programming. For many programming languages, a procedure executes a command or algorithm whereas a function returns a value. You can see these distinctions with the two types of blocks in the Functions category of the programming app. However, in Python® syntax programming, there is no distinction between these two terms and the term *function* is most common. Because of this, in the Python programming Units of Seeker content, the term *function* will be mostly used.
- See page 28 for the *Innovation Notebook* Answer Key for this task.

Notes:

Page 7 – Procedures

Students use the Load Code button to load a decomposed version of the "Für Elise" song program that uses procedures to play the different parts of the song. Sophia talks students through how it is broken down using procedures. Students save this version of the song and then upload and test the new program on Seeker.

Tasks to Complete

- Load and play the more efficient version of the "Für Elise" song program.
- Save the "Für Elise" song program as U3A2T3-FurElise.

Teaching Tips

- There should be a significant difference in the load time from the first "Für Elise" song program to the decomposed version. Point out that this is because of much more efficient code that uses fewer blocks.

Notes:

Page 8 – Events

Sophia explains the concept of events in programming and how events are actions that happen that the program can respond to. She guides students through creating a new program that uses events and procedures to give users an interface to play various sound and light effects. At this point, students focus on the logic (**if block** structure) of the program and record predictions for different button events in their *Innovation Notebook*.

Tasks to Complete

- Understand what an event is in programming.
- Use events and procedures to begin a sound/light effect program.
- Record predictions in the *Innovation Notebook* for what actions will occur based on different events in the program.

Teaching Tips

- Remind students that programming events are actions that the program can respond to. Simply touching Seeker on the head is not an action Seeker can detect with its current hardware; therefore, it is not considered an event in programming. On the other hand, pressing Seeker's Start button is an action Seeker can respond to, therefore it can be considered an event. Events with Seeker often require some kind of sensor to detect the action.
- In the if/else-if/else conditional structure, the order of events matters. Whichever event occurs first (or condition that is true), the associated action that will occur.
- See page 28 for the *Innovation Notebook* Answer Key for this task.

Notes:

Page 9 – Sound Effect Procedures

Sophia continues to guide students through creating the Sound Effect Interface program. Now, students will focus on building procedures for each event. Students are guided in building the first sound effect using procedures and procedure call blocks.

Tasks to Complete

- Build a procedure for the first button press event.
- Save the partially completed program to their profile as U3A2T4-SoundEffects.
- Test and debug the partially completed program.

Teaching Tips

- A common mistake in programming is to forget to include the procedure call block that tells robot or device when to run the procedure. Make sure students have added the function call block to the do section for if Button A is pressed.
- Students will define procedures for Buttons B, C, and D on the next page. For testing on this page, only Button A should play a sound effect.

Notes:

Page 10 – Sound Effects Interface

Continuing from the previous page, students create events and procedures to play three more sound effects, one for each button. Some example procedures are provided, and students can either recreate these sound effects or create their own sound effect procedures for each event. The one stipulation is that each procedure should have at least three block commands in it for controlling pixels and sounds. They also need to define Seeker's behavior so that, when no buttons are pressed, Seeker returns to its starting position.

Tasks to Complete

- Program three more effects for the sound effect interface program using events and procedures.
- Save over the previous program as U3A2T4-SoundEffects.
- Test and debug the completed program.

Teaching Tips

- Encourage students to be creative and to come up with their own effects rather than just using effects from the **play sound effects block** or copying the effects that are shown in the video.
- Remind students that they can include light effects to go with their sound effects.

Notes:

Page 11 – Keyboard Troubleshooting

Sophia provides students with a program that uses Seeker's buttons as a musical instrument to play notes in the manner of a piano or keyboard. Students load the program and, after Sophia explains how it is supposed to work, they test it on Seeker to determine that there are bugs in the program. Students work to identify the bugs using Seeker's debugging process. After the bugs are identified, students describe the bugs they find in their *Innovation Notebook*. After identifying at least three bugs, students then work to debug the program and record their fixes in the *Innovation Notebook*. To help with debugging, Sophia provides a flowchart of how the program is supposed to work.

Tasks to Complete

- Test the provided musical keyboard program and identify the bugs in the program.
- Record program bugs and fixes in the *Innovation Notebook*.
- Debug and fix the musical keyboard program.

Teaching Tips

- Students who are less musical might have a hard time identifying and fixing the bugs in the program. If this is the case, provide hints and clues as to where to look in the program.
- Remind students to refer to the provided flowchart to help them in debugging the program.
- See page 30 for the *Innovation Notebook* Answer Key for this task.

Notes:

Page 12 – Keyboard Songs

If they need it, students can use the Load Code button to load the fixed musical keyboard program. Students are then provided a PDF that shows the button presses for a couple common songs. Students can use this page to try to play one or both of these songs using Seeker. If time allows, they can try to figure out and play another song and see if another student can recognize it. Or, they can make up their own song.

Tasks to Complete

- Use the provided music notation to play common songs using Seeker's keyboard program.

Teaching Tips

- You might need to limit how much time students spend on this task making up their own songs or trying to figure out other songs.
- The PDF for playing the songs can be printed, if that is preferred. However, it will need to be printed in color so students can tell the difference between the red and blue notes.
- In the music notation, red notes are lower and are played without holding down the Seeker's Start button. Blue notes are higher and should be played while holding down Seeker's Start button.
- This task might be difficult for students who are not musically inclined. The PDF includes the songs "Camptown Races" and "Row Row Row Your Boat." It might be helpful to play the songs for them and have them play along using their Seeker. To do this though, you'll need to find the songs in the key of C major. If you can't find the songs in the right key, you can at least play the songs for them to get familiar with them and then have them play it using Seeker.

Notes:

Page 13 – Mechatronics Predictions

Students are given a mechatronics program using the Load Code button. They examine the program and make predictions about what Seeker will do when each of its buttons are pressed. Predictions are recorded in their *Innovation Notebook*.

Tasks to Complete

- Record predictions in the *Innovation Notebook* on what Seeker will do when each of its buttons are pressed while running the given mechatronics program.

Teaching Tips

- This program uses string variables to track the position of Seeker's arms (either "up" or "down"). This will be a new concept to students and will be explained on Page 14.

Notes:

Page 14 – Mechatronics Movement

Sophia walks students through the mechatronics program and especially how string variables are used to track the position of Seekers arms as either "up" or "down." After saving the program, students test and run the program in the simulator and on Seeker. Students compare Seeker's behavior to the predictions they made in their *Innovation Notebook*. If time permits, students can create a short obstacle course and then navigate Seeker through the course using its buttons.

Tasks to Complete

- Understand how string variables are used to track Seeker's arm position.
- Upload and run the program to compare Seeker's actual behavior to the predictions made in the *Innovation Notebook*.
- Save the program as U3A2T6-Mechatronics.

Teaching Tips

- Notify students whether or not they have time to create a short obstacle course or maze. If there is time, they can create it and then direct Seeker through it using the mechatronics program.

Notes:

Page 15 – Challenge: Robot Actor

Students are given their final challenge for Unit 3. They can choose which challenge they want to complete – either the Mechatronics Monologue Challenge or the Actor Avatar Challenge. The challenges are detailed on the challenge brief documents. The main difference between the two challenges is that the Mechatronics Monologue requires students to write their own monologue script for Seeker to perform where Actor Avatar allows to them use an already existing scene from a play, musical, or movie. Both challenges require Seeker to use its motors to move around the stage; use its arms and head to create gestures; incorporate lights and pixel animations to set the mood, convey emotions, or communicate signals; and incorporate sounds and music. To complete the challenge, students must demonstrate their working program on Seeker as well as make a three-minute presentation on their solution and the process for solving the challenge. Portions of the challenge such as planning their scene, writing pseudocode, making observations while testing, and creating their script should be completed in the *Innovation Notebook*. The challenge concludes with Seeker performing their three to five-minute monologue or scene.

Tasks to Complete

- Create an outline for the scene or monologue.
- Design and create the scene using the provided stage area and any stage props.
- Develop the program for Seeker to act out the scene.
- Test the program in the simulator and with Seeker.
- Complete all the Challenge sections of the *Innovation Notebook*.
- Perform the scene by running the program and narrating the scene as needed.

Teaching Tips

- Decide how much time to give students to complete the challenge. Base your decision on the ability and maturity level of your students, the amount of time you have available for completing this Unit, how much detail is put into Seeker's program to act out the scene, and the quality of the presentation you want them to perform. We suggest three to five days. Whatever you decide, make sure students know their deadline.
- Because there are two different challenges, you can either let students decide which challenge to complete or you can make the choice for them. The Mechatronics Monologue may take a little longer to complete due to the need to write their own story and script. However, this challenge is a great way to include some cross-curricular ELA standards into the activity as well.
- Refer to the Actor Avatar and Mechatronics Monologue challenge brief documents for specific information about the challenge. You can find these documents in the Teacher Resources section of your classroom in the Learning Portal.
- Students can view the challenge brief document digitally on their device. However, for some students, it might be beneficial to provide printed copies of the document for them to refer to as well.
- The challenge brief includes a section on creating a physical test course that resembles a stage. Before starting the challenge, determine if you want each student group to create their own stage, whether student groups work together to create a shared stage, or whether you create a single classroom stage for all groups to use. When making this decision, consider the space available in your classroom.
- Consider using a real theater stage for the final challenge performances. You can even invite administrators, teachers, other classes, and even parents to join the audience.
- As students demonstrate their challenge solution, keep the criteria and constraints of the challenge in mind for assessment and evaluation.
- See page 31 for the *Innovation Notebook* Answer Key for the challenge.
- Use the following evaluation rubrics for assessing the Actor Avatar Challenge and the Mechatronics Monologue Challenge.

Notes:

Actor Avatar Evaluation Rubric

Category	0 Points	1 Point	2 Points	3 Points	4 Points	Points Earned
3-5 Minute Time Limit	The performance was not delivered.	The performance was more than 1 minute too short or long.	The performance was 30 seconds to 1 minute too short or long.	The performance was up to 30 seconds too short or long.	The performance was presented in the 3- to 5-minute time limit.	
Seeker Features	The performance included none of the following Seeker features: • Drive motor movement • Servo movement of arms and head • Lighting effects using pixel LEDs • Music or sound effects	The performance included 1 of the following Seeker features: • Drive motor movement • Servo movement of arms and head • Lighting effects using pixel LEDs • Music or sound effects	The performance included 2 of the following Seeker features: • Drive motor movement • Servo movement of arms and head • Lighting effects using pixel LEDs • Music or sound effects	The performance included 3 of the following Seeker features: • Drive motor movement • Servo movement of arms and head • Lighting effects using pixel LEDs • Music or sound effects	The performance included all the following Seeker features: • Drive motor movement • Servo movement of arms and head • Lighting effects using pixel LEDs • Music or sound effects	
Stage Boundaries	While performing, Seeker did not stay within the stage boundaries.	N/A	N/A	N/A	While performing, Seeker stayed within the stage boundaries.	
Button Press Events	Button press events were not used in the performance.	N/A	N/A	N/A	Button press events were used to perform different parts of the scene.	
Procedures	The program incorporated 0 procedures to break the program into sections.	The program incorporated 1 or more procedures that were incorrectly used to break the program into sections.	The program incorporated 1 correctly used procedure to break the program into sections.	The program incorporated 2 correctly used procedures to break the program into sections.	The program incorporated 3 or more correctly used procedures to break the program into sections.	
Variables	The program did not incorporate any variables.	The program incorporated 1 variable, but it was not well named, and it wasn't clear what it was used for.	The program incorporated 1 well-named variable, and it was clear what it was used for.	The program incorporated at least 2 variables, but they were not well named, and it wasn't clear what they were used for.	The program incorporated at least 2 well-named variables, and it was clear what they were used for.	
Conditions	The program did not include conditions.	The program included conditions, but the logic of how the conditions were used did not make sense.	The program included 1 condition, and the logic of how the condition was used made sense.	The program included 2 conditions, and the logic of how the conditions were used made sense.	The program included at least 3 conditions, and the logic of how the conditions were used made sense.	
Narration	Narration met none of these 4 aspects of good public speaking: • What was spoken matched what was being acted out using the robot. • Narration was clear and at an appropriate volume. • Narration tone varied to convey emotion – was not monotone. • Body language was appropriate.	Narration met 1 of these 4 aspects of good public speaking: • What was spoken matched what was being acted out using the robot. • Narration was clear and at an appropriate volume. • Narration tone varied to convey emotion – was not monotone. • Body language was appropriate.	Narration met 2 of these 4 aspects of good public speaking: • What was spoken matched what was being acted out using the robot. • Narration was clear and at an appropriate volume. • Narration tone varied to convey emotion – was not monotone. • Body language was appropriate.	Narration met 3 of these 4 aspects of good public speaking: • What was spoken matched what was being acted out using the robot. • Narration was clear and at an appropriate volume. • Narration tone varied to convey emotion – was not monotone. • Body language was appropriate.	Narration met all 4 aspects of good public speaking: • What was spoken matched what was being acted out using the robot. • Narration was clear and at an appropriate volume. • Narration tone varied to convey emotion – was not monotone. • Body language was appropriate.	
					Total Score:	

Mechatronics Monologue Evaluation Rubric

Category	0 Points	1 Point	2 Points	3 Points	4 Points	Points Earned
3-5 Minute Time Limit	The performance was not delivered.	The performance was more than 1 minute too short or long.	The performance was 30 seconds to 1 minute too short or long.	The performance was up to 30 seconds too short or long.	The performance was presented in the 3- to 5-minute time limit.	
Seeker Features	The performance included none of the following Seeker features: • Drive motor movement • Servo movement of arms and head • Lighting effects using pixel LEDs • Music or sound effects	The performance included 1 of the following Seeker features: • Drive motor movement • Servo movement of arms and head • Lighting effects using pixel LEDs • Music or sound effects	The performance included 2 of the following Seeker features: • Drive motor movement • Servo movement of arms and head • Lighting effects using pixel LEDs • Music or sound effects	The performance included 3 of the following Seeker features: • Drive motor movement • Servo movement of arms and head • Lighting effects using pixel LEDs • Music or sound effects	The performance included all the following Seeker features: • Drive motor movement • Servo movement of arms and head • Lighting effects using pixel LEDs • Music or sound effects	
Stage Boundaries	While performing, Seeker did not stay within the stage boundaries.	N/A	N/A	N/A	While performing, Seeker stayed within the stage boundaries.	
Button Press Events	Button press events were not used in the performance.	N/A	N/A	N/A	Button press events were used to perform different parts of the scene.	
Procedures	The program incorporated 0 procedures to break the program into sections.	The program incorporated 1 or more procedures that were incorrectly used to break the program into sections.	The program incorporated 1 correctly used procedure to break the program into sections.	The program incorporated 2 correctly used procedures to break the program into sections.	The program incorporated 3 or more correctly used procedures to break the program into sections.	
Variables	The program did not incorporate any variables.	The program incorporated 1 variable, but it was not well named, and it wasn't clear what it was used for.	The program incorporated 1 well-named variable, and it was clear what it was used for.	The program incorporated at least 2 variables, but they were not well named, and it wasn't clear what they were used for.	The program incorporated at least 2 well-named variables, and it was clear what they were used for.	
Conditions	The program did not include conditions.	The program included conditions, but the logic of how the conditions were used did not make sense.	The program included 1 condition, and the logic of how the condition was used made sense.	The program included 2 conditions, and the logic of how the conditions were used made sense.	The program included at least 3 conditions, and the logic of how the conditions were used made sense.	
Narration	Narration met none of these 4 aspects of good public speaking: • What was spoken matched what was being acted out using the robot. • Narration was clear and at an appropriate volume. • Narration tone varied to convey emotion – was not monotone. • Body language was appropriate.	Narration met 1 of these 4 aspects of good public speaking: • What was spoken matched what was being acted out using the robot. • Narration was clear and at an appropriate volume. • Narration tone varied to convey emotion – was not monotone. • Body language was appropriate.	Narration met 2 of these 4 aspects of good public speaking: • What was spoken matched what was being acted out using the robot. • Narration was clear and at an appropriate volume. • Narration tone varied to convey emotion – was not monotone. • Body language was appropriate.	Narration met 3 of these 4 aspects of good public speaking: • What was spoken matched what was being acted out using the robot. • Narration was clear and at an appropriate volume. • Narration tone varied to convey emotion – was not monotone. • Body language was appropriate.	Narration met all 4 aspects of good public speaking: • What was spoken matched what was being acted out using the robot. • Narration was clear and at an appropriate volume. • Narration tone varied to convey emotion – was not monotone. • Body language was appropriate.	
					Total Score:	

Notes:

Page 16 – Wrap-Up

The Unit wraps up with the team at Tronic discussing the details of using Seeker in their theater productions. Sophia offers a deal: if their company purchases 14 robots, Adaptibots will donate an extra Seeker and will sponsor their next production if Tronic uses Seeker in the performance. The Unit ends with a purchase order being signed.

Tasks to Complete

- Complete the activity and the Unit.

Teaching Tips

- If time permits, consider asking some of the discussion questions provided near the start of this document.

Notes:

Activity 2 *Innovation Notebook* Answer Key

Sound Exploration

Exploration ideas:

- ❏ Try some different sound effects to see what they sound like.
- ❏ Use **play tone blocks** to play different pitches. The default pitch is 1,000 hertz, but humans can generally hear pitches between 20 hertz and 20,000 hertz.
- ❏ Experiment by playing different musical notes. Try putting multiple **play musical note blocks** together to see if you can make a song or jingle.

Sound and Frequency

Program	How the Frequency Changes	Observations
1	Doubling the frequency	Observations will vary, but students should notice that the pitch of the sound goes up as the frequency goes up. Students who have a musical background might notice that when the frequency doubles, the pitch goes up by a full musical octave.
2	Halving the frequency	Observations will vary, but students should notice that the pitch of the sound decreases as the frequency is cut in half. Students who have a musical background might notice that when the frequency halves, the pitch goes down by a full musical octave.
3	Increasing the frequency by 10 Hz	Observations will vary, but students should notice that the pitch of the sound goes up as the frequency goes up. Because the frequency changes very fast according to the program (every 0.1 second), it sounds like a siren that increases pitch gradually.
4	Choosing your own frequencies	Observations will vary depending on which frequencies are chosen.

Describe the relationship between a sound's frequency and its pitch.

Students should recognize that frequency and pitch are related: the higher the frequency, the higher the pitch of the sound.

Decomposing the Song

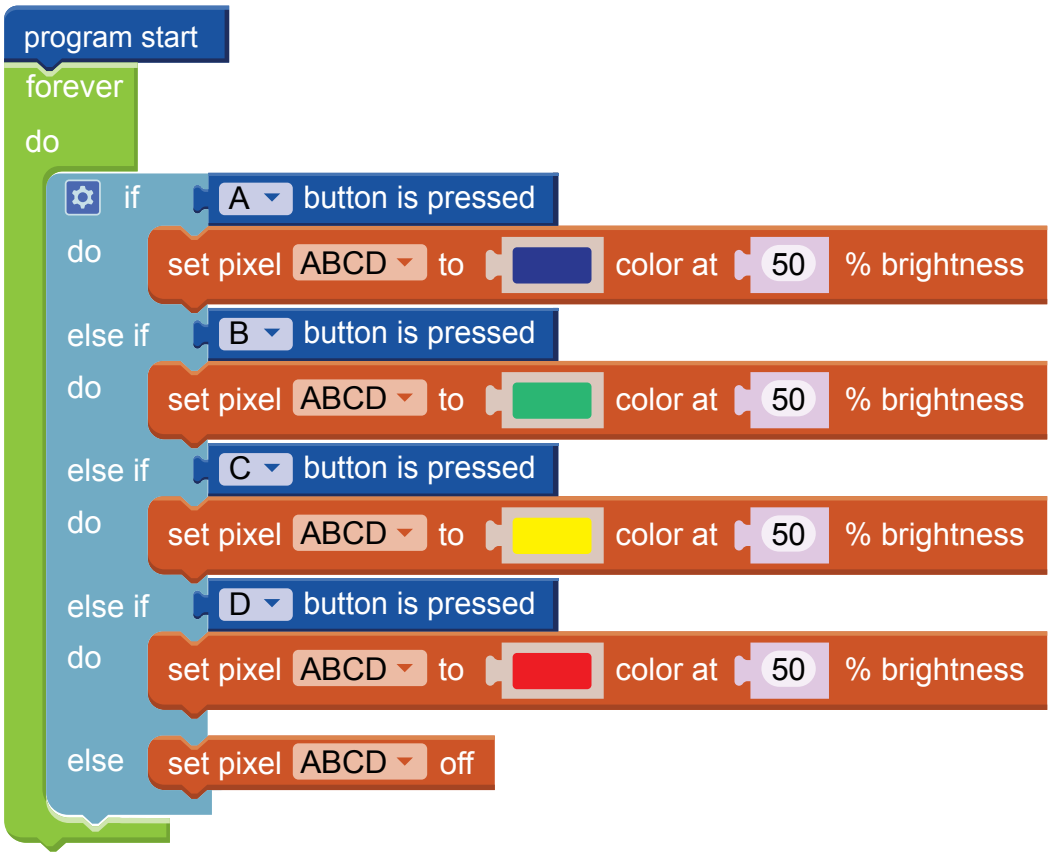
Review the "Für Elise" song program. Look for blocks of code that repeat or that have natural breaks. Think about where we could add procedures in this song to make the code more efficient or easier to read. Jot down your ideas in the space provided.

There are lots of ways the "Für Elise" song could be broken down into procedures. Listed here are some examples:

- Setting the note length variables (W, H, Q, E, S, DH, DQ, and DE) happens multiple times in the program as the tempo variable changes. The program could be shortened by having these blocks in a procedure that is used multiple time in the program as the tempo changes.
- Students might recognize that the opening notes of the song are repeat a few times throughout the code. A procedure for playing these notes could be created and used any time those specific notes need to be played.
- There are natural break points at every **wait block**. Each section of notes that ends with a **wait block** could be a procedure.

Events

When Seeker runs the program shown, explain what you think will happen for the events listed in the following table and explain why this will happen. If you don't know, upload the program to Seeker and run the program to find out what happens.



Event	What Will Happen?	Why?
Button C is pressed	The pixels will turn yellow.	Pressing Button C would be the first true condition in the **if block**, so it's do section is executed.
Button A is pressed	The pixels will turn blue.	Pressing the Button A would be the first true condition in the **if block**, so it's do section is executed.
No buttons are pressed	The pixels will turn off.	Pressing no buttons means there are no true conditions in the **if block**, so the else section is executed.
Buttons B and D are pressed at the same time	The pixels will turn green.	The Button B condition comes before the Button D condition. Because the Button B condition is the first true condition, the pixels would turn green. In an if/else-if structure, the first true condition is executed and the rest are skipped, even if others are true as well.
All the buttons are pressed at the same time	The pixels will turn blue.	The Button A condition comes first, and, because it would be true if all the buttons are pressed, it is the one that is executed. In an if/else-if structure, the first true condition is executed and the rest are skipped, even if others are true as well.

Keyboard Troubleshooting

Bug	Bug Description	Bug Fix
1	The last note keeps playing when the button is released. The sound should stop if no buttons are pressed.	Change the conditional structures to include an else section for the main conditional and for the nested conditional. Inside the else section, add commands to stop sound and to set all the pixels off.
2	When the Start button is pressed, the notes don't change.	In the A note and B note procedures, the **play musical note blocks** need to be changed to play the A (5) note and the B (5) note.
3	When the Start button and Button D are pressed, the wrong procedure is called.	In the nested conditional, delete the C note procedure call in the else-if condition for Button D. Replace it with a High C note procedure call.

Mechatronics Predictions

As you examine Lucas's mechatronics program, use the following table to make predictions about what Seeker will do when each of its buttons are pressed.

Event	Bug Fix
Pressing the Button A	Seeker will play a sound effect, blink the A pixel red, and back up a short distance of 20 cm.
Pressing the Button B	Seeker will play a sound effect, turn its head to the left, blink the B pixel (left pixel) orange, and then makes a 90-degree turn to the left.
Pressing the Button C	Seeker will play a sound effect, blink the C pixel green, and drive forward a short distance of 20 cm.
Pressing the Button D	Seeker will play a sound effect, turn its head to the right, blink the D pixel (right pixel) orange, and then makes a 90-degree turn to the right.
Pressing the Start button	Seeker will alternate its arm position between up and down every time the Start button is pressed.

Challenge

Plan

Use the following space to outline your play, musical, movie scene, or monologue.

Answers will vary depending on which challenge is being completed and the scene Seeker will be acting out.

Draw a sketch of your scene.

Sketches will vary depending on what Seeker is acting out.

Develop pseudocode for Seeker to act out the play, musical, movie scene, or monologue.

Answers will vary depending on which challenge is being completed and the scene Seeker will be acting out.

Put to the Test

Program Observation Notes	Program Modification Notes
Observations will vary.	Modifications will vary.

Present

Include a copy of your script here. If you're using a paper notebook, print a copy of your script and attach it here inside your notebook. If you're using a digital notebook, copy and paste your script in the following space.

Students should include a full script of their narration.

Notes:

PITSCO
EDUCATION

Pitsco Education • P.O. Box 1708, Pittsburg, KS 66762 • 800-835-0686 • Pitsco.com

PE•0325•0000•00 79428