Lesson 4: Input Unplugged

Overview

In preparation for delving deeper into programming with App Lab, students will explore how a handful of different programs written in both Game Lab and App Lab handle taking input from the user. After comparing and contrasting the approaches they saw in the example apps, students group up to act out the two different models for input (conditionals in an infinite loop and asynchronous events) to gain a better understanding of how they work.

Purpose

This lesson is intended to help students transition from the *draw loop* and conditional model of input used in Game Lab to the event-driven model used in App Lab. While students have experienced and learned a bit about event-driven programming in Unit 4, a deeper understanding of how the events work will help when it comes to responding to events on the Circuit Playground later in this unit.

Assessment Opportunities

1. **Compare and contrast multiple ways to take input**

   Wrap Up: Students should distinguish between the draw loop model of Game Lab and the event model of App Lab.

2. **Model different methods of taking user input**

   As students participate in the activity, circulate the room and ensure that they are following the instructions.

Standards

Full Course Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Agenda

Warm Up (5 minutes)

Comparing Input Methods

Activity (35 minutes)

Objectives

Students will be able to:

- Compare and contrast multiple ways to take input
- Model different methods of taking user input

Preparation

- Prepare to display example programs for the whole class.
- A half deck of cards for each group of three students or **deck-of-cards.js.org**.
- Print one copy of the activity guide for each group of four students.

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the teachers

- **CSD Unit 6 - Physical Computing** - Slides

For the students

- **Input and Events** - Activity Guide

Input Unplugged
Activity Debrief

Wrap Up (5 minutes)
Reflection

Teaching Guide

Warm Up (5 minutes)

Comparing Input Methods

Discuss: Yesterday we wrote some apps that took input from a user, but how did the program know *when* to take input?

Discussion Goal

Goal: This is intended to be a very broad question, and as such could go in a number of different directions. If your students are struggling, encourage them to think about specific programs that they wrote in units 3 and 4.

Display: Open up the Code Studio stage for this level. For each of the example apps, ask the class to identify:

1. Where is the input coming from? (e.g. keyboard, mouse, etc)
2. What input value is the program looking for?
3. How will the program respond to input?

1-2 **Input Examples**

1 2

Prompt: Ask the class the following questions, keeping track of answers on the board. The goal at this point isn't to answer questions that come up, but to record them so we can see how the class feels again after the lesson.

- How did the different apps we looked at approach taking input similarly or differently?
- Which way made the most sense to you?
- Which way made the least sense?
- What questions to you have about how these programs take input?

Activity (35 minutes)

Input Unplugged

Remarks

> In the previous activity we explored several different kinds of input, but how do our programs actually process that input? As a class, and in small groups, we're going to model how input is taken both in

> Game Lab and App Lab.

Group: Place students in groups of 4

Distribute: Hand out a copy of the activity guide and half of a deck of cards to each group (the deck doesn't need to be perfectly split, but should include both red and black cards in each half). The activity guide includes four different role pages, printed front and back.

Model: Choose one group to model the activity for the class.

Input and Events

The goal of this activity is to help students better understand how events work, and how using events differs from the input model that they used in Game Lab. In groups of four, students will model the two different approaches to taking input. By the end of the activity students should understand that the `onEvent` block sets up a input to watch for events in the background, allowing input handling without explicitly asking each input for a value constantly. The roles for this activity are:

Program: The Program reads the code aloud and performs any actions dictated by the inputs. **Inputs 1-3:** The Inputs draw cards from the deck, which represent their changing input values.

Version A: Asking for Input

This first version models the Game Lab approach to taking input. In this scenario the **Program** explicitly asks each input for a value every time the draw loop is run.

Teaching Tip

The programs for both versions of this activity are purposely very simple to allow students to focus on the processes involved in each input method. If your students pick up on the ideas quickly, consider providing some more detailed example programs to push on their understanding. In particular, placing the inputs or events in a different order and nesting conditionals can be useful in revealing misconceptions.

Rules for Version A:

- The **Program** reads each line of code aloud.
- At the beginning of each draw loop, each **Input** draws a new card from the deck. This represents the current state of that input.
- Whenever the **Program** reaches an input checking command (`inputOne()`, `inputTwo()`, or `inputThree()`), the **Program** asks to see that inputs card.
- If the value of the card shown matches the associated conditional, the **Program** performs the action(s) inside the conditional.

After groups have run the program a few times and seem to understand how it works, have them switch up roles so that each student gets a chance to be the **Program**.

Version B: Input Events

This time around, instead of constantly asking for input in a loop, the **Program** will assign each **Input** an Event to watch for. In the context of this activity, an Event is either a red or black card, but in a program that Event could be a click on a button, movement of the mouse, press of a key, or more. This is a simplified model for the way input is handled in App Lab.

Rules for Version B:

- Each **Input** draws a card from the deck at a rate of roughly one per second.
- The **Program** reads each line of code aloud.

- When the **Program** reaches an `onEvent` command, they tell the specified **Input** which Event to watch for, and what the Response should be.
- When assigned an Event, the **Input** writes down the details in their *Events to Watch* table.
- Every time an **Input** draws a card, they check to see if it matches one of the Events in their table. If it does, they tell the **Program** to perform the Response.

There are a few important simplifications that are made in this model to minimize the number of roles and rules:

- This model implies that Inputs are actually watching for Events on their own. In reality, an Event Handler running in the background watching for the specified event.
- In this model the **Input** tells the **Program** what actions to take. In reality, when an Event Handler is triggered, the Program goes to that portion of their code and runs everything in the function.

Activity Debrief

Discuss: Looking back at the comments gathered from the warm up activity, what things are more clear after having run the activity? Do you have any new questions?

Wrap Up (5 minutes)

Reflection

Journal: What's the difference between the way that Game Lab and App Lab handle inputs?

Assessment Opportunity

Students may have different ways of explaining the differences, but Game Lab's draw loop model works sequentially, with each line of code run after the other, and the draw loop code running over and over in sequence. Inputs are only checked when that line of code is run. App Lab's event model is continuously checking for inputs once an event handler is created.