Lesson 8: The Program Design Process

Overview

This lesson introduces students to the process they will use to design programs of their own throughout this unit. This process is centered around a project guide which asks students to sketch out their screens, identify elements of the Circuit Playground to be used, define variables, and describe events before they begin programming. This process is similar to the Game Design Process that we used in Unit 3. In this lesson students begin by playing a tug o' war style game where the code is hidden. They discuss what they think the board components, events, and variables would need to be to make the program. They are then given a completed project guide which shows one way to implement the project. Students are then walked through this process through a series of levels. At the end of the lesson students have an opportunity to make improvements to the program to make it their own.

Purpose

This lesson gives students to practice developing a larger scale program in order to prepare them to do so independently. While previous lessons have focused on building skills around using specific elements of the Circuit Playground and related programming concepts, this lesson focuses on combining everything learned so far into a more complex program. The lesson heavily scaffolds the software development process by providing students a completed project guide, providing starter code, and walking students through its implementation. In the subsequent lessons students will need to complete a greater portion of this guide independently, and for the final project they will follow this process largely independently.

Assessment Opportunities

1. **Implement different features of a program by following a structured project guide**

   The final program should reflect the functions, events, and screen design as shown in the project guide.

2. **Develop a program that responds to events from a hardware input**

   Code Studio: See rubric on bubble 12

Objectives

Students will be able to:

- Create a function that uses parameters to generalize behavior
- Develop a program that responds to events from a hardware input
- Implement different features of a program by following a structured project guide

Preparation

- Provide students with copies of the project guide

Links

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

For the teachers

- **Booleans and Comparison Operators** - Resource
- **CSD Unit 6 - Physical Computing** - Slides
- **Functions** - Resource
- **If Statements** - Resource

For the students

- **Emoji Race** - Project Guide

3. **Create a function that uses parameters to generalize behavior**

   Code Studio: See rubric on bubble 12

Standards

Full Course Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming
- ▶ **CS** - Computing Systems

Agenda

Warm Up (5 minutes)
Play Emoji Race
Stop: Review Project Guide

Activity (80 minutes)
Implement Project Guide

Wrap Up (5 minutes)
Make It Your Own

Teaching Guide

Warm Up (5 minutes)

Play Emoji Race

Group: Place students in pairs

Demo: Show students the Emoji Race demo, available at the beginning of this lesson.

Transition: Send students online to try out the game on their own. Each pair should play the game and follow the instructions which ask them to list the board components, events, and variables they think are necessary to create this game.

Stop: Review Project Guide

Discuss: Students should have individually created a list of board components, events, and functions they would need to make the game they played. Ask students to share their lists with a neighbor before discussing as a class.

Discussion Goal

Running the Conversation: You can write "Board", "Events", and "Functions" on the board and record their ideas below each. Ask students to justify their decisions but don't feel the need to settle on one right answer.

Teaching Tip

Keeping Focus: Students can easily get distracted by the fun of playing the game. Let them play for a while but eventually encourage them to follow the on-screen instructions and make a list of the board components, events, and variables that would be necessary to create the game.

Distribute: Give each student or pair of students a copy of the project guide.

Prompt: Compare the list of things you thought would be needed to the ones on this project guide. Do you notice any differences?

Discuss: As a class compare the list you had on the board to the list of board components, events, and functions on the project guide. Note the similarities. Where there are differences try to understand why. Don't approach one set as "right" vs. "wrong" but just confirm both would be able to make the game students played.

Teaching Tip

Sharing the Project Guide: Students are not actually writing on the project guide in this lesson. You can give each student their own copy for reference but you might also choose to print one copy per pair, share digital copies, or just display the guide on the projector. So long as it is available for reference any approach will work fine.

Remarks

> There's usually lots of ways you can structure a program to get it to work the way you want. The important thing when writing complex or large programs is that you start with a plan. Today we're going to look at how we could implement this plan to build our own emoji race game. By the end of the lesson you'll not only have built your game, but you'll know how to change it and make it your own. Let's get going!

Activity (80 minutes)

Implement Project Guide

Students are given starter code to establish the functions and event handlers needed for this project. These levels guide students through how to use the provided variables and what ought to occur within the event handlers. As students move through the levels point out how the project guide is being used. Though this project is highly scaffolded, the next lesson will ask students to develop a hardware-based program of their own design, so make sure to reinforce the connection between the planning that was done in the project guide and the programming levels.

1 **Sample Program**

2 **Plan Your Project**

3-5 **Screen Design**

3 4 5

6-8 **Finishing Functions**

6 7 8

Teaching Tip

Exposure to Functions with Parameters

Parameters are a powerful way to make functions more useful by giving them specific input. We are introducing this here primarily as a way to expose students to this concept, but functions with parameters won't be taught thoroughly until chapter 2. For students who are interested in experimenting more with this technique now, direct them to the **documentation.**

9-12 **Checking for a Winner**

9 10 11 12

Assessment Opportunity

Level 12 You can use this level as a formative assessment for students. Click inside the level to view a rubric and leave feedback to your students

13 **Make it Your Own**

Wrap Up (5 minutes)

Make It Your Own

This last level encourages students to make the game their own. If students have made their way to this point they have all the skills they need to progress through the curriculum, so there is no pressure to complete any of the modifications suggested in this level. If you have time, however, getting practice planning and implementing new features will be a useful skill. Even just modifying the look of the screens is an easy way students can make the game their own.

Share: Once students have completed the project they can share their work with their classmates. Encourage students to showcase the additional code they wrote and explain how it has changed the way the game works.