Lesson 18: Project - Interactive Card

90 minutes

Overview

This end-of-chapter assessment is a good place for students to bring together all the pieces they have learned (drawing, variables, sprites, images, conditionals, user input) in one place. In this project, students plan for and develop an interactive greeting card using all of the programming techniques they've learned to this point. Giving students the opportunity to really be creative after learning all these new concepts will help to engage them further as they head into Chapter 2.

Question of the Day: What skills and practices are important when creating an interactive program?

Standards

Full Course Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

Agenda

Warm Up (5 minutes)
- **Journal**

Activity (80 minutes)
- **Demo Project Exemplars (Level 1)**
- **Step 1: Define - Plan Your Card**
- **Step 2: Prepare - Design Your Card**
- **Step 3: Try - Develop Your Card**
- **Peer Review**
- **Iterate - Update Code**
- **Step 4: Reflect**

Wrap Up (5 minutes)
- **Reflection**

Objectives

Students will be able to:

- Apply an iterator pattern to variables or properties in a loop
- Sequence commands to draw in the proper order
- Use conditionals to react to keyboard input or changes in variables / properties

Preparation

- Print out a copy of the project guide for each student
- Check the **"Teacher's Lounge"** forum for verified teachers to find additional strategies or resources shared by fellow teachers
- If you are teaching virtually, consider checking our **Virtual Lesson Modifications**

Links

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

For the teachers

- **Extra Code in Challenge Levels** - Resource ▾ Make a Copy
- **Project - Interactive Card** - Slides ▾ Make a Copy

For the students

- **Computer Science Practices** - Activity Guide ▾ Make a Copy
- **Interactive Card** - Rubric ▾ Make a Copy

- **Interactive Card - Activity Guide** - Activity Guide Make a Copy
- **Interactive Card - Peer Review** - Activity Guide Make a Copy
- **Interactive Card - Student Checklist** - Resource Make a Copy
- **Problem Solving with Programming** - Resource Make a Copy

Teaching Guide

Warm Up (5 minutes)

Journal

Prompt: Think of one time you gave or received a card from someone. Who was that person? What was the purpose of the card? What about the card made it specific to that purpose?

Share: Have students share out their ideas.

Discussion Goal: This discussion introduces the chapter project: creating an interactive card. This is a good chance to tie in the 'card' concept to the problem-solving process, by defining the "problem" as cheering someone up, or letting them know that we are thinking about them.

Remarks

> We're going to start designing our own cards today. Because we have learned how to program, our cards are going to be interactive. At the end of the project, you'll be able to email or text your card to someone.

Question of the Day: What skills and practices are important when creating an interactive program?

Activity (80 minutes)

Demo Project Exemplars (Level 1)

Goal: Students see an example of a final project and discuss the different elements that went into making it.

Display: Show students the sample program.

1 **Demo App: Interactive Card**

Discuss: Have students share their observations and analyses of the program.

Encourage the class to consider that there are multiple approaches to programming anything, but that there may be clues as to how something was created. In particular, when they are sharing their thoughts ask them to specify the following:

- Clues that suggest a sprite was used
- Clues that suggest a conditional was used
- Clues that suggest an iterator pattern was used

Display: Show students the rubric and student checklist. Review the different components of the rubric with them to make sure they understand the components of the project.

Distribute: Pass out copies of the **Interactive Card - Activity Guide**. Student should use this activity guide to plan out what they want to create before they head to the computer so that once they get to the computer they are just executing the plan. Give students some time to brainstorm the type of card they want to create and who the recipient will be.

Also distribute a copy of the **Interactive Card - Student Checklist** to each student. Students can use this to self-assess and reflect as they develop their project.

Teaching Tip

Scoping Student Projects: Students may ideate projects that are beyond the skills they currently have or that would take longer than the allotted time to implement. Rather than asking students to choose a different project, consider asking students to imagine a more scaled-down version of their initial idea. As an analogy, if students initial idea is the "Run" step, imagine a less intense version that represents what the "Walk" step would look like. If necessary, you can keep going back further to a "Crawl" step as well.

Digging Deeper: This is sometimes referred to as the Minimal Viable Product - you can learn more about this process and adapt it into your project strategies by reading this article: **Making Sense of MVP** by Henrik Kniberg

Step 1: Define - Plan Your Card

Circulate: As students fill out a description of the interactive card they want to create, circulate the room to ask students about their ideas and help guide them to make sure they are thinking of how they could include all concepts they've learned about so far.

Step 2: Prepare - Design Your Card

As students fill in the activity guide about the design of their card, continue to support student ideas and brainstorms as they are asked to consider the different aspects of their card:

1. The first layer of the interactive card is a background drawn with just the commands in the Drawing drawer. The front of the Activity Guide provides a grid for students to lay out their background, a reference table of drawing commands, and an area for students to take notes and write pseudocode.
2. Next, students think through the sprites they'll need, filling out a table with each sprite's label, images, and properties.
3. Finally, students consider the conditionals they'll need in order to make their card interactive.

Optional: Students can visit Code Studio - Level 2 for more interactive card ideas if needed.

2 **Interactive Card Examples**

Teaching Tip

New Code and Previous Challenge Levels: Throughout the unit, students may have learned additional code in the challenge levels of different lessons. As students approach this project, they may want to revisit the code they learned in these levels or visit them for the first time to learn new codes to use in this project. To help guide students back to previous levels, you can use the **Extra Code in Challenge Levels** as a resource to quickly find where new codes were introduced in earlier challenge levels.

Step 3: Try - Develop Your Card

Remarks

> Many of you are ready to start creating your programs. Don't forget about using the problem solving process to help with the *blank screen* effect. If you feel stuck or you're not sure what to do next, remember you can always follow the steps of the problem-solving process to **define** your next step, **prepare** for what you want to code, **try** it out, then **reflect** on whether or not it solved your problem.

Distribute: Hand out a copy of the **Problem Solving with Programming** to pairs of students or have students take this resource out if they kept it after the last project. Encourage students to look over the guide.

Display: Show the slide with the problem solving process graphic.

Transition: Once students have completed their planning sheet, it's time to head to the Code.org website. The short level sequence asks students to complete each element of their project.

Teaching Tip

Scaffolded Tasks: The tasks for this mini-project are broken down for students with each level focusing on a particular part of the project (background, sprites, text, etc). Encourage students to follow the instructions in each level, focusing on one task at a time.

Variable Names Take an opportunity to go over the importance of naming variables with descriptive names. (Note that the Exemplar uses Surpise1 and Suprise2 as variable names so that it doesn't give away what the surprises are). You can have a discussion around variable names and how it can impact their program if they have vague names making it harder to identify their variables later as they code.

Circulate: Encourage students to use the steps in the Problem-Solving Process for Programming when they get stuck or are unsure of what to do next.

3-7 **Project Work**

3 4 5 6 7

Teaching Tip

Debugging Strategies: As students design and implement their own project ideas, they may find themselves with new bugs that they need to untangle and you may find yourself looking at completely unfamiliar code as students look for help troubleshooting their errors. To help smooth out the debugging experience, consider the following strategies:

- Review the **Teacher Guide to Debugging** for some common questions and strategies to help support students in debugging their code
- Have students follow the steps in the **Student Guide to Debugging** and use the **Bug Report Quarter-Sheets** as an initial step in the debugging process. This helps students prepare and communicate their issue before asking for help.
- If students haven't seen it yet, consider showing the **Debugging Video** to the class to reinforce debugging best practices.

Digging Deeper: Consider supplying students with an object to talk to as part of the debugging process. This is sometimes known as Rubber Duck Debugging - you can learn more on the website **https://rubberduckdebugging.com/**

Peer Review

Distribute: Give each student a copy of the peer review guide.

Students should spend 15 minutes reviewing the other student's card and filling out the peer review guide.

Iterate - Update Code

Circulate: Students should complete the peer review guide's back side and decide how to respond to the feedback they were given. They should then use that feedback to improve their cards.

Students should use the rubric and student checklist to self-assess and make sure their project matches with the rubric.

Teaching Tip

Rubric and Checklist: Students have two resources they can use for self-reflection and making sure they are on the right track: the rubric and the student checklist. We recommend having students use the checklist for their own self-assessment and reflection, since it may be easier to digest and understand when reviewing their own project. However, we recommend teachers use the full rubric for evaluating projects to give more accurate feedback to students. You can see examples of this with the Sample Marked Rubrics resource at the top of the lesson plan (only visible to verified teachers)

Step 4: Reflect

Using the rubric, students should assess their own project before submitting it.

Send students to Code Studio to complete their reflection on their attitudes toward computer science. Although their answers are anonymous, the aggregated data will be available to you once at least five students have completed the survey.

8 **Reflection**

Assessment Opportunity

Use the project rubric attached to this lesson to assess student mastery of the learning goals of this unit.

Wrap Up (5 minutes)

Reflection

Question of the Day: What skills and practices are important when creating an interactive program?

Prompt: Have students reflect on their development of the five practices of CS Discoveries (Problem Solving, Persistence, Creativity, Collaboration, Communication).

- Choose one of the five practices which you demonstrated growth in during this lesson. Write something you did that showed this practice.
- Choose one practice you think you can continue to grow in. What's one thing you'd like to do better?

- Choose one practice you thought was especially important for this project. What made it so important?