Lesson 24: Mini-Project - Flyer Game

45 minutes

Overview

This lesson is another chance for students to get more creative with what they have learned. Students use what they have learned about simulating gravity and the different types of collisions to create simple flyer games. After looking at a sample flyer game, students brainstorm what sort of flyer games they would like, then use a structured process to program the game in Code Studio.

Question of the Day: How can the new types of collisions and modeling movement be used to create a game?

Standards

Full Course Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Agenda

Warm Up (5 minutes)

Activity (35 minutes)

- **Step 1: Define - Plan the Game**
- **Step 2: Prepare - Design Your Game**
- **Step 3: Try - Program Your Game**
- **Step 4: Reflect**

Wrap up (5 minutes)

- **Share Out and Journal 3-2-1**

Preparation

- 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

- **Mini-Project - Flyer Game** - Slides ▾ Make a Copy

For the students

- **Flyer Game** - Rubric ▾ Make a Copy
- **Flyer Game** - Activity Guide ▾ Make a Copy
- **Problem Solving with Programming** - Resource ▾ Make a Copy

Teaching Guide

Warm Up (5 minutes)

Review

Ask students to think of all of the things that they have learned how to do in the unit so far, and display their answers to the class. This is a good time to check in on any concepts that have been challenging for students.

Remarks

> You've already learned all of the sprite interactions and types of movement that we will cover this unit. Today you'll have a chance to put them all together to make a flyer game.

Question of the Day: How can the new types of collisions and modeling movement be used to create a game?

Teaching Tip

Facilitating Mini-Projects: Mini-Projects act as checkpoints in the curricula and cover the subset of skills students have seen so far in the unit. They are designed for 1-2 days of implementation as a way to check-in with how well students understand the course content so far. You may decide to extend these projects as a way to support or challenge students, which could allow you to revisit difficult concepts or support students who may have missed lessons and are trying to catch up. However, we recommend deciding this ahead of time and being firm with students about how much time they have for each project - otherwise, it's easy for projects to drag-out to multiple days and for student's work to spiral beyond the scope of this project.

Activity (35 minutes)

Distribute: Pass out copies of the **Flyer Game**. Student should use this activity guide to plan out and brainstorm how to program the flyer game before they head to the computer so that once they get to the computer they are just executing the plan.

Optional Transition You can choose to either send students to Code Studio to play the flyer game example game or demo the game for the students.

1 **Flyer Game Example**

Step 1: Define - Plan the Game

Circulate: As students fill out a description of the aspects of the flyer game they want to keep the same and what they want to change, circulate the room to ask students about their ideas and help guide them.

Step 2: Prepare - Design Your Game

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

1. Students are asked to identify parts of programming the game that they are not sure how to do yet. If students identify concepts that have previously been covered, suggest they go back and review.
2. Next, students are given the opportunity to identify new animations for the three sprites in the game.
3. The first layer of the game is a background. The activity guide provides a space for students to lay out their background and an area for students to take notes and write pseudocode.
4. Next, students think through how each sprite is programmed to move and interact with the other sprites and given an opportunity to decide if they want to change anything.

Step 3: Try - Program Your Game

Remarks

> Many of you are ready to start creating your game. 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.

2 Make Your Sprites

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, player controls, etc). Encourage students to follow the instructions in each level, focusing on one task at a time.

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.

3-5 Player Controls

3 4 5

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/**

6 Sprite Movement

7 Sprite Interactions

8 **Review Your Game**

Assessment Opportunity

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

Step 4: Reflect

Direct students to complete the last step of their activity guide to reflect on how they did programming this game.

Wrap up (5 minutes)

Share Out and Journal 3-2-1

Share: Allow students time to play each other's flying games. Ask them to focus not just on the new behavior that they added but also the code they used to create it.

Journal: Have students write and reflect about the following prompts.

- What are three things you saw in someone else's game that you really liked?
- What are two improvements you'd make to your game if you had more time?
- What's one piece of advice you'd give to someone making this type of game?

Question of the Day: How can the new types of collisions and modeling movement be used to create a game?