

Lesson 27: Using the Game Design Process

90 minutes

Overview

In this multi-day lesson, students use the problem-solving process from Unit 1 to create a platform jumper game. This lesson also builds on the use of the Project Guide in the previous lesson by having students complete more of this project guide independently before using it to build a game. Students begin the lesson by looking at an example of a platform jumper, then define what their games will look like. Next, they use a structured process to plan the backgrounds, variables, sprites, and functions they will need to implement their game. After writing the code for the game, students will reflect on how the game could be improved, and implement those changes.

Question of the Day: How can the problem-solving process help programmers to manage large projects?

Standards

Full Course Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

Agenda

Warm Up (5 minutes)

Activity (80 minutes)

Play Alien Jumper

Discuss Project Guide

Share out

Wrap Up (5 minutes)

Journal

Objectives

Students will be able to:

- Implement different features of a program by following a structured project guide
- Identify core programming constructs necessary to build different components of a game

Preparation

- Print one copy of the project guide for each student or pair of students
- 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

- **Using the Game Design Process** - Slides [▼ Make a Copy](#)

For the students

- **Planning Your Platform Game** - Activity Guide [▼ Make a Copy](#)

Teaching Guide

Warm Up (5 minutes)

Prompt: The Problem Solving Process helps us work through all kinds of problems. Think about the problem of building a larger piece of software, like the game we built in the last lesson. What did each of the 4 steps look like? Why were they important?

Discuss: Students should brainstorm quietly and write down what each step might be. Afterward, lead a share-out discussion. You can record ideas on the board. Possible parts of each step include:

- Define: Figuring out what you want the game to look like, how it should work, and who will play it.
- Prepare: Plan ahead what your code will look like. Decide on a structure for your game.
- Try: Write the code following your plan.
- Reflect: Test your code, play the game to make sure it works, and get feedback from other people to make the game better.

Discussion Goal: Students should share their thoughts but if it doesn't come up naturally then suggest the examples provided above. This discussion will motivate the use of the project guide for building a game later in the lesson.

Remarks

When you build software, the problem solving process can be a helpful guide. Obviously we need to write the code, but being careful to define what you want to build, making a good plan to build it, and reflecting afterwards on how to improve it are all part of making good software. Today we're going to use this process to make a new game.

Question of the Day: How can the problem solving process help programmers to manage large projects?

Activity (80 minutes)

Play Alien Jumper

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

Remarks

We're going to be building a jumper game today. You'll have a chance to play a sample game, then plan out how you would create the game on your Project Guide.

Discuss Project Guide

Circulate: Students should complete the project guide in the style of the one they saw in the previous lesson. They will likely want to keep the game up as they try to determine the behavior each of the sprites will have.

Share: Students share out their plans for making the game. Reassure them that there are many correct ways to make the same piece of software, and that they will have a chance to try out their ideas in code studio.

✔ Assessment Opportunity ▲

Identify core programming constructs necessary to build different components of a game

You can check that the student has identified key functions, sprites, and variables needed in the program and that the general description of the program is accurate in the Project Guide.

💡 Teaching Tip ▲

Making the game will take at least two class periods. If there's not enough time for all students to finish the lesson, groups of students can work to code different aspects, then share their code with each other. For example, one group could work on the platforms, one on the stars, and another on the player. Students who have finished early can choose more challenges from the later levels.

🖥️ 1

Sample Platform Jumper Game

📄 2

Build a Platform Jumper

🖥️ 3-6

Platform Jumper - Background and Variables

3

4

5

6

🖥️ 7-9

Platform Jumper - Platforms

7

8

9

🖥️ 10-12

Platform Jumper - Items

10

11

12

🖥️ 13-16

Platform Jumper - Player

13

14

15

16

🖥️ 17



Platform Jumper Review

✅ Assessment Opportunity ▲

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

Share out

Share: Students share their games with their classmates.

Wrap Up (5 minutes)

Journal

Question of the Day: How can the problem solving process help programmers to manage large projects?

Prompt: Before you started coding your game, you first had to fill out a project guide with a plan. How did having a plan change the way that you coded your game? Will you do anything differently when you make your plan for your final project?



This work is available under a **Creative Commons License (CC BY-NC-SA 4.0)**.

If you are interested in licensing Code.org materials for commercial purposes **[contact us](#)**.