# Lesson 28: Project - Design a Game

## 225 minutes

## Overview

Students will plan and build their own game using the project guide from the previous two lessons. Working individually or in pairs, students will first decide on the type of game they'd like to build, taking as inspiration a set of sample games. They will then complete a blank project guide where they will describe the game's behavior and scope out the variables, sprites, and functions they'll need to build. In Code Studio, a series of levels prompts them on a general sequence they can use to implement this plan. Partway through the process, students will share their projects for peer review and will incorporate feedback as they finish their game. At the end of the lesson, students will share their completed games with their classmates. This project will span multiple classes and can easily take anywhere from 3-5 class periods.

**Question of the Day:** How can the five CS practices (problem-solving, persistence, communication, collaboration, and creativity) help programmers to complete large projects?

## Standards

Full Course Alignment

**CSTA K-12 Computer Science Standards (2017)**

▶ **AP** - Algorithms & Programming

## Agenda

**Warm Up (5 minutes)**

**Activity (215 minutes)**
- **Review Project Guide**
- **Step 1: Define - Scope Game**
- **Step 2: Prepare - Complete Project Guide**
- **Step 3: Try - Write Code**
- **Step 4: Reflect - Peer Review**
- **Iterate - Update Code**
- **Share**

**Wrap Up (5 minutes)**
- **Journal**
- **Reflect**
- **Teacher End-Of-Unit Survey**

**After the Lesson**

## Objectives

Students will be able to:

- Create a plan for building a piece of software by describing its major components
- Implement a plan for creating a piece of software
- Independently scope the features of a piece of software

## Preparation

- Print copies of the project guide, one for each student / pair of students
- Print copies of the rubric, one for each student / pair of students
- Print copies of the peer review guide, one for each student / pair of students
- Review sample games in Code Studio
- 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 - Design a Game** - Slides ▾ Make a Copy

**Post-Project Test**
**End of Course Survey**

For the students

- **Computer Science Practices** - Activity Guide Make a Copy
- **Make Your Own Game** - Activity Guide Make a Copy
- **Make Your Own Game** - Rubric Make a Copy
- **Make Your Own Game - Peer Review** - Activity Guide Make a Copy
- **Make Your Own Game - Student Checklist** - Resource Make a Copy
- **Problem Solving with Programming** - Resource Make a Copy

# Teaching Guide

## Warm Up (5 minutes)

**Prompt:** Today, you'll start the final project of the unit, in which you will design and code your own game. Before you start, what are three skills or qualities that you think will be important as you complete this project?

**Share:** Allow students to share out their ideas.

**Discussion Goal:** This discussion serves to set cultural norms for the project. Students should expect that the project will be challenging, but that they will have plenty of opportunities to iterate on their work as they work together to learn as a community.

### *Remarks*

> There are lots of practices that programmers use when working on large projects. As you design and build your game, try to reflect on how you are using the five practices of problem solving, persistence, communication, collaboration, and creativity.

**Question of the Day: How can the five CS practices (problem solving, persistence, communication, collaboration, and creativity) help programmers to complete large projects?**

## Activity (215 minutes)

**Group:** This project can be completed individually or in pairs. At your discretion, you may choose to have students form larger groups as well.

Teaching Tip

**Facilitating Group Projects:** If students are working in pairs or small teams to complete projects, consider showing these two videos to the class:

- **How Teamwork Works**
- **Dealing with Disagreements**

Depending on your goals with this project, consider having teams complete a **Student Guide to Team Planning**, which reinforces the message in the video

## Review Project Guide

**Distribute:** Each student or group of students should be given a copy of the project guide. As a class, review the different steps of the project and where they appear in the project guide.

**Distribute:** Give each student a copy of the rubric or student checklist so that they know from the beginning what components of the project you will be looking for.

## Step 1: Define - Scope Game

**Circulate:** Students should spend the first 15-20 minutes playing the sample games, reviewing past work, and discussing as a group the type of game they'd like to build. If they want they can sketch ideas on scratch paper or in their journals.

1 **Sample Games**

Teaching Tip

**Class Brainstorm:** Before diving into individual projects, consider leading a class brainstorm of what some example projects could look like that fit this criteria. You may also decide to show one or two of the exemplar apps to help inspire the brainstorm. Using these ideas, students may find it easier to latch onto an initial idea for their project

## Step 2: Prepare - Complete Project Guide

**Circulate:** Once students have discussed their ideas for the project, they should complete the project guide. While this should be a fairly familiar process, encourage students to make each component as clear and detailed as they can. Planning ahead can help them identify issues in their plan before they'll need to make more significant changes to their code.

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 3: Try - Write Code

**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 Code Studio to start programming. The levels provide some guidance on how students may go about implementing their project guide. None of the steps are significantly different from what students have seen from the previous two lessons. If they wish, students can work in a different order than the one suggested in these levels.

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 [r extra*code*in*challenge*levels/csd/2026] as a resource to quickly find where new codes were introduced in earlier challenge levels.

**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.

2-5 **Project - Background and Variables**

2 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/**

6-10 **Project - Sprites and Interactions**

6 7 8 9 10

## Step 4: Reflect - Peer Review

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

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

11 Review Your Game

## 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 game.

Students should refer to the rubric or student checklist to ensure they meet the requirements of the project.

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

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)

Assessment Opportunity

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

You may also choose to assign the post-project test through Code Studio.

## Share

**Share:** Give students a chance to share their games. If you choose to let students do a more formal presentation of their projects, the project guide provides students a set of components to include in their presentations including:

- The original game they set out to build
- A description of the programming process including at least one challenge they faced and one new feature they decided to add
- A description of the most interesting or complex piece of code they wrote
- A live demonstration of the actual game

# Wrap Up (5 minutes)

## Journal

**Question of the Day: How can the five CS practices (problem solving, persistence, communication, collaboration, and creativity) help programmers to complete large projects?**

**Prompt:** Have students reflect on their development of the **five practices of CS Discoveries** (Problem Solving, Persistence, Creativity, Collaboration, Communication). Choose one or more of the following prompts as you deem appropriate.

- Choose one of the five practices in which you believe you demonstrated growth in this unit. Write something you did that exemplified 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 the project we completed today. What made it so important?

### Reflect

Send students to Code Studio to complete a quick end-of-unit survey. Although their answers are anonymous, the aggregated data will be available to you once at least five students have completed the survey.

12 **End Of Unit Survey**

### Teacher End-Of-Unit Survey

We also have a teacher end-of-unit survey to learn more about how the unit went for you and your students. While students take their survey, **please complete this end of unit survey for teachers** as well. Your feedback is valued and appreciated!

## After the Lesson

### Post-Project Test

**Post-Project tests** are included at the end of every unit. These include several multiple choice and matching questions as well as open ended reflections on the final project of the unit. These tests are aligned to the **learning framework** of each unit and are designed to assess parts of the framework that may not have been covered by the project rubrics. To holistically assess the learning objectives of the unit, the post-project test should be paired with the end-of-unit project which is the primary student assessment in each unit.

**Unlocking The Tests:** This test is locked and hidden from student view by default. In order for students to see and take this test, you'll need to unlock it by clicking the "Lock Settings" button and following the instructions that appear. **Click here for more information about unlocking and admistering assessments**

### End of Course Survey

If this is the last unit of CS Discoveries that you are teaching, also have students take the end-of-course survey. See the **CSD Instructions resource** for more information about the End-of-Course survey and how to assign and see the results.