Unit 3 - Interactive Animations and Games

In the Animations and Games unit, students build on their coding experience as they create programmatic images, animations, interactive art, and games. Starting off with simple, primitive shapes and building up to more sophisticated sprite-based games, students become familiar with the programming concepts and the design process computer scientists use daily. They then learn how these simpler constructs can be combined to create more complex programs. In the final project, students develop a personalized, interactive program. Along the way, they practice design, testing, and iteration, as they come to see that failure and debugging are an expected and valuable part of the programming process.

Chapter 1: Images and Animations

Big Questions

- What is a computer program?
- What are the core features of most programming languages?
- How does programming enable creativity and individual expression?
- What practices and strategies will help me as I write programs?

Week 1

Lesson 1: Programming for Entertainment

Unplugged

Question of the Day: How is computer science used in entertainment? The class is asked to consider the "problems" of boredom and self expression, and to reflect on how they approach those problems in their own lives. From there, they will explore how Computer Science in general, and programming specifically, plays a role in either a specific form of entertainment or as a vehicle for self expression.

Lesson 2: Plotting Shapes

Unplugged

Question of the Day: How can we clearly communicate how to draw something on a screen? This lesson explores the challenges of communicating how to draw with shapes and use a tool that introduces how this problem is approached in Game Lab.The class uses a Game Lab tool to interactively place shapes on Game Lab's 400 by 400 grid. Partners then take turns instructing each other how to draw a hidden image using this tool, which accounts for many of the challenges of programming in Game Lab.

Lesson 3: Drawing in Game Lab

Game Lab

Question of the Day: How can we communicate to a computer how to draw shapes on the screen? The class is introduced to Game Lab, the programming environment for this unit, and begins to use it to position shapes on the screen. The lesson covers the basics of sequencing and debugging, as well as a few simple commands. At the end of the lesson, students will be able to program images like the ones they made with the drawing tool in the previous lesson.

Lesson 4: Shapes and Parameters

Question of the Day: How can we use parameters to give the computer more specific instructions? In this lesson, students continue to develop a familiarity with Game Lab by manipulating the width and height of the shapes they use to draw. The lesson kicks off with a discussion that connects expanded block functionality (e.g. different sized shapes) with the need for more block inputs, or "parameters." Finally, the class learns to draw with versions of ellipse() and rect() that include width and height parameters and to use the background() block.

Week 2

Lesson 5: Variables

Game Lab

**Question of the Day: How can we use variables to store information in our programs? ** This lesson introduces variables as a way to label a number in a program or save a randomly generated value. The class begins the lesson with a very basic description of the purpose of a variable and practices using the new blocks, then completes a level progression that reinforces the model of a variable as a way to label or name a number.

Lesson 6: Random Numbers

Question of the Day: How can we make our programs behave differently each time they are run? Students are introduced to the randomNumber() block and how it can be used to create new behaviors in their programs. They then learn how to update variables during a program, and use those skills to draw randomized images.

Lesson 7: Sprites

Game Lab

Question of the Day: How can we use sprites to help us keep track of lots of information in our programs? In order to create more interesting and detailed images, the class is introduced to the sprite object. The lesson starts with a discussion of the various information that programs must keep track of, then presents sprites as a way to keep track of that information. Students then learn how to assign each sprite an image, which greatly increases the complexity of what can be drawn on the screen.

Lesson 8: Sprite Properties

Question of the Day: How can we use sprite properties to change their appearance on the screen? Students extend their understanding of sprites by interacting with sprite properties. The lesson starts with a review of what a sprite is, then moves on to Game Lab for more practice with sprites, using their properties to change their appearance. The class then reflects on the connections between properties and variables.

Week 3

Lesson 9: Text

Question of the Day: How can we use text to improve our scenes and animations? This lesson introduces Game Lab's text commands, giving students more practice using the coordinate plane and parameters. At the beginning of the lesson, they are asked to caption a cartoon created in Game Lab. They then move onto Code Studio where they practice placing text on the screen and controlling other text properties, such as size.

Lesson 10: Mini-Project - Captioned Scenes

Game Lab | Project

Question of the Day: How can we use Game Lab to express our creativity? After a quick review of the code learned so far, the class is introduced to the first creative project of the unit. Using the problem solving process as a model, students define the scene that they want to create, prepare by thinking of the different code they will need, try their plan in Game Lab, then reflect on what they have created. They also have a chance to share their creations with their peers.

Lesson 11: The Draw Loop

Game Lab

Question of the Day: How can we animate our images in Game Lab? This lesson introduces the draw loop, one of the core programming paradigms in Game Lab. Students learn how to combine the draw loop with random numbers to manipulate some simple animations first with dots and then with sprites.

Lesson 12: Sprite Movement

Game Lab

Question of the Day: How can we control sprite movement in Game Lab? In this lesson, the class learns how to control sprite movement using a construct called the counter pattern, which incrementally changes a sprite's properties. After brainstorming different ways that they could animate sprites by controlling their properties, students explore the counter pattern in Code Studio, using the counter pattern to create various types of sprite movements.

Week 4

Lesson 13: Mini-Project - Animation

Game Lab | Project

Question of the Day: How can we combine different programming patterns to make a complete animation? In this lesson, the class is asked to combine different methods from previous lessons to create an animated scene. Students first review the types of movement and animation that they have learned, and brainstorm what types of scenes might need that movement. They then begin to plan out their own animated scenes, which they create in Game Lab.

Lesson 14: Conditionals

Game Lab

Question of the Day: How can programs react to changes as they are running? This lesson introduces students to booleans and conditionals, which allow a program to run differently depending on whether a condition is true. The class starts by playing a short game in which they respond according to whether particular conditions are met. They then move to Code Studio, where they learn how the computer evaluates boolean expressions, and how they can be used to structure a program.

Lesson 15: Keyboard Input

Game Lab

Question of the Day: How can our programs react to user input? Following the introduction to booleans and if statements in the previous lesson, students are introduced to a new block called keyDown(), which returns a boolean and can be used in conditionals statements to move sprites around the screen. By the end of this lesson they will have written programs that take keyboard input from the user to control sprites on the screen.

Lesson 16: Mouse Input

Game Lab

Question of the Day: What are more ways that the computer can react to user input? The class continues to explore ways to use conditional statements to take user input. In addition to the keyboard commands learned yesterday, they learn about several ways to take mouse input. They also expand their understanding of conditionals to include else, which allows for the computer to run a certain section of code when a condition is true, and a different section of code when it is not.

Week 5

Lesson 17: Project - Interactive Card

Game Lab | Project | Pair Programming

Question of the Day: What skills and practices are important when creating an interactive program? In this culminating project for Chapter 1, students plan for and develop an interactive greeting card using all of the programming techniques they've learned to this point.

Chapter Commentary

Students build up toward programming interactive animations in the Game Lab environment. They begin with simple shapes and sprite objects, then use loops to create flipbook style animations. Next, they learn to use booleans and conditionals to respond to user input. At the end of the chapter, students design and create an interactive animation that they can share with the world.

Chapter 2: Building Games

Big Questions

- How do software developers manage complexity and scale?
- How can programs be organized so that common problems only need to be solved once?
- How can I build on previous solutions to create even more complex behavior?

Week 6

Lesson 18: Velocity

Game Lab

Question of the Day: How can programming languages hide complicated patterns so that it is easier to program? After a brief review of how the counter pattern is used to move sprites, the class is introduced to the idea of hiding those patterns in a single block, in order to help manage the complexity of programs. They then head to Code Studio to try out new blocks that set a sprite's velocity directly, and look at the various ways that they are able to code more complex behaviors in their sprites.

Lesson 19: Collision Detection

Game Lab

Question of the Day: How can programming help make complicated problems more simple? In this lesson, the class learns about collision detection on the computer. Working in pairs, they explore how a computer could use math, along with the sprite location and size properties, to detect whether two sprites are touching. They then use the isTouching() block to create different effects when sprites collide, and practice using the block to model various interactions.

Lesson 20: Mini-Project - Side Scroller

Game Lab | Project

Question of the Day: How can the new types of sprite movement and collision detection be used to create a game? Students use what they have learned about collision detection and setting velocity to create simple side scroller games. After looking at a sample side scroller game, they brainstorm what sort of side scroller they would like to make, then use a structured process to program the game in Code Studio.

Week 7

Lesson 21: Complex Sprite Movement

Game Lab

Question of the Day: How can previous blocks be combined in new patterns to make interesting movements? The class learns to combine the velocity properties of sprites with the counter pattern to create more complex sprite movement. After reviewing the two concepts, they explore various scenarios in which velocity is used in the counter pattern, and observe the different types of movement that result. In particular, students learn how to simulate gravity. They then reflect on how they were able to get new behaviors by combining blocks and patterns that they already knew.

Lesson 22: Collisions

Game Lab

Question of the Day: How can programmers build on abstractions to create further abstractions? In this lesson, the class programs their sprites to interact in new ways. After a brief review of how they used the isTouching block, students brainstorm other ways that two sprites could interact. They then use isTouching to make one sprite push another across the screen before practicing with the four collision blocks (collide, displace, bounce, and bounceOff).

Lesson 23: Mini-Project - Flyer Game

Game Lab | Project

Question of the Day: How can the new types of collisions and modeling movement be used to create a game? 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, they brainstorm what sort of flyer they would like, then use a structured process to program the game in Code Studio.

Week 8

Lesson 24: Functions

Game Lab

Question of the Day: How can programmers use functions to create their own abstractions? This lesson covers functions as a way for students to organize their code, make it more readable, and remove repeated blocks of code. The class learns that higher level or more abstract steps make it easier to understand and reason about steps, then begins to create functions in Game Lab.

Lesson 25: The Game Design Process

Game Lab

Question of the Day: How does having a plan help to make a large project easier? This lesson introduces the process that students will use to design games for the remainder of the unit. This process is centered around a project guide that asks students to define their sprites, variables, and functions before they begin programming their game. They walk through this process in a series of levels. At the end of the lesson, students have an opportunity to make improvements to the game to make it their own.

Lesson 26: Using the Game Design Process

Game Lab

Question of the Day: How can the problem solving process help programmers to manage large projects? In this multi-day lesson, the class uses the problem solving process from Unit 1 to create a platform jumper game. After looking at a sample game, they define what their games will look like and use a structured process to build them. Finally, the class reflects on how the games could be improved, and implements those changes.

Week 9

Lesson 27: Project - Design a Game

Game Lab | Project | Presenting Information

Question of the Day: How can the five CS practices (problem solving, persistence, communication, collaboration, and creativity) help programmers to complete large projects? Students plan and build original games using the project guide from the previous two lessons. Working individually or in pairs, they plan, develop, and give feedback on the games. After incorporating the peer feedback, students share out their completed games.

Chapter Commentary

In this chapter students combine the constructs that they learned in the first chapter to program more complex movement and collisions in their sprites. As they create more complex programs, they begin to use functions to organize their code. In the end, students use a design process to create an original game.

Lesson 1: Programming for Entertainment

Overview

Question of the Day: How is computer science used in entertainment?

Students are asked to consider the "problems" of boredom and self expression, and to reflect on how they approach those problems in their own lives. From there, students will explore how computer science in general, and programming specifically, plays a role in either a specific form of entertainment or as a vehicle for self expression.

Purpose

This lesson is intended to kick off this programming unit in a way that engages students of all backgrounds and interests. Though the end point of this unit asks students to develop a game, you should avoid starting out with a strong emphasis on *video* games. Instead, we attempt to broaden students' perspective about how programming is relevant to a form of entertainment or self expression that is personally engaging. This will provide an anchor for students to come back to throughout the unit as they consider the potential applications of the various programming skills that they learn.

Assessment Opportunities

1. **Identify how computer science is used in a field of entertainment**

   In the activity guide, look at the "Interesting Fact or Use" section of the second page and make sure students have identified a use of computer science in their chosen fields.

Agenda

Warm Up (10 min)

- **The Entertainment Problem**

Activity (45 min)

- **CS in Entertainment**
- **Researching your Topic**
- **Research Notes**
- **Exploring Games with Game Lab**

Wrap Up (5 min)

- **Looking Forward**

View on Code Studio

Objectives

Students will be able to:

- Identify how computer science is used in a field of entertainment

Preparation

- [ ] Review the research resources linked in Code Studio
- [ ] Print a copy of the activity guide for each group of three students

Links

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

For the Teachers

- **CS in Entertainment** - Exemplar
- **Programming for Entertainment** - Slides

For the Students

- **CS in Entertainment** - Activity Guide
  Make a Copy ▾

Teaching Guide

Warm Up (10 min)

The Entertainment Problem

Prompt: What is your favorite form of entertainment, and what problem does it solve for you?

Discuss: Allow students to share out their ideas with the group.

Question of the Day: How is computer science used in entertainment?

Discussion Goal

The goal here is to get students to reflect more critically on *why* people seek out entertainment. Students are likely to come up with "boredom" as a common answer, but push them to think more deeply about what their chosen form of entertainment actually does for them - what problem does it solve. Potential answers include:

- Connection with others
- Learning or experiencing new things
- Sparking creativity
- An escape from reality

Activity (45 min)

CS in Entertainment

Remarks

> Whether it's movies, music, art, games, or any number of other options, we have many types of entertainment open to us. Today, we're going to look at how computer science plays a role in some of these different fields.

Group: Place students in groups of three. Consider allowing students to group based on common interest as each group will be exploring a field of entertainment together.

Distribute: Give each group a copy of the activity guide.

Entertainment Exploration

During this activity student groups will do some light research into the role that CS and programming play in various fields of entertainment. The primary goal of this activity is to broaden students' perspectives about how programming can be used to make fun or entertaining things. Some of the fields that students could research (such as art, animation, and games) can be directly connected to programs they will write later in this unit, while others may serve more as an inspiration for how the skills that they learn here may be applied in different domains.

Topics

In the activity guide there are a number of potential fields for research. These specific fields were chosen to go along with resources that are provided on Code Studio, but you can have students look into other fields if they wish.

Researching your Topic

On Code Studio, inside the blue teacher box, there are a handful of useful sites to help students kick off their research. Share the links that you feel are most helpful and appropriate for your class.

Teaching Tip

Because many of these sites change their content daily, we have not shared them directly with students. We suggest that you check these links the morning before class to ensure that everything is still appropriate to be shared in school.

Code Studio levels

- Levels
- **2**

Student Instructions

View on Code Studio

Starting Your Search

To find out more about how computer science and programming play a role in entertainment, you'll need to do some research. Try searching for "Computer science and ______" and trying out different types of entertainment such as film, television, music, games, animation, fashion, etc.

Teaching Tip

Here are a few sites to get students started. These sites are generally appropriate for school, but the content with them changes frequently, so **we strongly suggest that you check each site for inappropriate content before sharing it with students**.

View on Code Studio to access answer key(s)

Research Notes

Circulate As students search the web, they may need support in staying focused on the research task. The goal of this exploration is twofold:

- First, develop a deeper understanding of how programming is used in the chosen field. How is computer technology changing this field, and what are some of the problems that people are trying to solve with technology?
- Second, identify some interesting applications of CS or facts to share. What are some cool things that people are doing in this field that make use of CS?

Interesting Information

Once groups have learned a bit about how CS is used in their chosen field, they can complete the second page of the activity guide, which asks them to do the following:

- **What Problem Does It Solve?:** "Problem" in this context can be fairly broad. It might be simplifying an otherwise laborious or complex task, doing things that wouldn't otherwise be possible, or any number of things that make this form of entertainment more accessible to end users.
- **How is it an Improvement?:** This could be tightly coupled with the answer to the previous question. Push students to consider how their form of entertainment was created before programming was prevalent.
- **An Interesting Fact or Use:** Share something fun or interesting about how CS is used in this field.
- **An Open Question:** It's likely that students come away with more questions than answers after just some quick research. Encourage them to reflect on what they don't yet know, and what questions they'd still like answered.

Share: Give groups a minute each to share their findings.

Exploring Games with Game Lab

Remarks

> In this unit, we're going to have a chance to create our own entertainment through animations and games using a tool called Game Lab. Here are some samples of programs made using the Game Lab tool.

Teaching Tip

While some of these examples are pulled directly from lessons that students will complete later in the unit, a few of them use commands or techniques that aren't explicitly covered in the course. You can view the source for all of these programs in order to support students who may want to incorporate some of these techniques later on.

Display: Show either as a whole class, or let students explore independently, the example programs at the end of this lesson's Code Studio progression. These programs are designed to show a variety of different kinds of programs that it's possible to make in Game Lab.

Code Studio levels

- Levels
- **3**
- **4**
- **5**
- **6**

Student Instructions

View on Code Studio

Stamp Pad

Click "Run" to start the program, then use the stamp pad to draw pictures with simple colors and animal stamps.

Student Instructions

View on Code Studio

Animated Comics

Combining images, text, and some subtle animation can make for really interesting comics or graphic stories. Click "Run" to see an example.

Student Instructions

View on Code Studio

Alien Jumper

Press "Run" to play the game on the left. You can make the alien jump with the space bar, and move it to the left and right with the arrow keys. You score by collecting stars, and if you score high enough, the background will change.

Student Instructions

View on Code Studio

Hungry Bunny

The bunny is hungry, and it's looking for mushrooms and carrots for dinner.

To win, you'll need to find a dinner bowl, then collect at least ten carrots and five mushrooms.

Make sure to avoid the bugs. Ladybugs and snails will eat your food, and bees will sting you, making you drop everything!

Use the space bar to jump. You can squash ladybugs and snails by jumping on them.

Click "Run" to start the program.

Wrap Up (5 min)

Looking Forward

Question of the Day: How is computer science used in entertainment?

Journal: Based on what you saw today, both in your research and the example apps, what kinds of programs are you most interested in learning to create?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **IC** - Impacts of Computing

Lesson 2: Plotting Shapes

Overview

Question of the Day: How can we clearly communicate how to draw something on a screen?

Students explore the challenges of communicating how to draw with shapes and use a tool that introduces how this problem is approached in Game Lab. The warm up activity quickly demonstrates the challenges of communicating position without some shared reference point. In the main activity, students explore a Game Lab tool that allows students to interactively place shapes on Game Lab's 400 by 400 grid. They then take turns instructing a partner how to draw a hidden image using this tool, accounting for many challenges students will encounter when programming in Game Lab. Students optionally create their own image to communicate before a debrief discussion.

Purpose

The primary purpose of this lesson is to introduce students to the coordinate system they will use in Game Lab. Students may have limited experience with using a coordinate grid or may struggle with the "flipped" y-axis in Game Lab. The drawing tool also forces students to think about other features of Game Lab students will see when they begin programming in the next lesson. These include the need to consider order while drawing, the need to specify color, and the fact that circles are positioned by their center and squares by their top-left corner. By the end of this activity, students should be ready to transfer what they have learned about communicating position to the programming they will do in the next lesson.

Assessment Opportunities

1. **Reason about locations on the Game Lab coordinate grid**

   See the final reflection questions. This objective will also be assessed in the next lesson, within the context of programming in Game Lab.

2. **Communicate how to draw an image in Game Lab, accounting for shape position, color, and order**

   See the final reflection questions.

Agenda

Warm Up (10 min)

Communicating Drawing Information

Activity (35 min)

Drawing with a Computer

View on Code Studio

Objectives

Students will be able to:

- Reason about locations on the Game Lab coordinate grid
- Communicate how to draw an image in Game Lab, accounting for shape position, color, and order

Links

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

For the Teachers

- **Drawing Shapes** - Exemplar
- **Plotting Shapes** - Slides
- **Sample Shape Drawing** - Exemplar

For the Students

- **Drawing Shapes (Version B)** - Activity Guide Make a Copy
- **Drawing Shapes (Version A)** - Activity Guide Make a Copy

Wrap Up (5 min)

Journaling

Teaching Guide

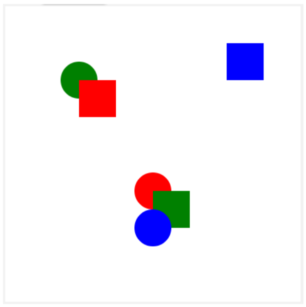
Warm Up (10 min)

Communicating Drawing Information

Remarks

> We saw a lot of different programs yesterday, and you started to think about what types you might want to create. When we create a program, one of the things we need to do is draw everything on the screen. We're going to try that with a "student" computer today.

Ask one or two volunteers to come to the front of the room and act as "computers" for the activity. They should sit with their backs to the board so that they cannot see what is being projected. Give each volunteer a blank sheet of paper.



Display: Project **Sample Shape Drawing - Exemplar** where it can be seen by the class.

Remarks

> You will need to explain to our "computer" how to draw the picture. In the end, we'll compare the drawing to the actual picture.

Give the students a minute or two to describe the drawing as the students at the front of the room try to draw it. After one minute, stop them and allow the students to compare both pictures.

Prompt: What are the different "challenges" or problems we're going to need to solve in order to successfully communicate these kinds of drawings?

Discuss: Students should silently write their answers in their journals. Afterwards they should discuss with a partner and then with the entire class.

Discussion Goal

Goal This discussion is intended to bring up some challenges that students will need to address in the next few lessons, such as how to specify position, order, and color. The students don't necessarily need to decide how they would specify such things, but recognize that they will need a method for doing so. Students who have just come out of the HTML unit may make connections to how HTML helped solve similar problems.

Remarks

> There were several challenges we needed to solve in this activity. We need to be able to clearly communicate position, color, and order of the shapes. We're going to start exploring how to solve this problem.

Question of the Day: How can we clearly communicate how to draw something on a screen?

Activity (35 min)

Drawing with a Computer

Group: Place students in pairs.

Transition: Have one member of each group open a laptop and go to the contents for this lesson. There is a single level with a Game Lab tool.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Drawing Shapes 2 ***(click tabs to see student view)***

Prompt: Working with your partner, take two or three minutes to figure out how this tool works. Afterwards be ready to share as a class.

Discuss: After pairs have had a chance to work with the tool, run a quick share-out where students discuss features they notice.

Discussion Goal

Goal: Rather than doing a live-demo of the tool, use this strategy to let students explore the tool themselves. Afterwards use the debrief to ensure all students are aware of key features of the tool. The most important components are the grid and the fact that the mouse coordinates are displayed below the drawing space.

Drawing Shapes

Distribute activity guides to each pair, ensuring that one student (Student A) receives Version A and the other student (Student B) receives Version B. Students should not look at each other's papers.

Set Up: In this activity, students will try to recreate images based on a partner's directions. The student who is drawing will use the shape drawing tool on Game Lab to draw the shapes. Students should keep their drawings hidden from one another throughout the activity. While completing a drawing the instruction-giver should also not be able to see the computer screen.

Content Corner

Location of the Origin: The origin of this grid, as well as the origin in Game Lab, lies at the top left corner. This reflects the fact that documents tend to start at the top left, and ensures that every point on the plane has positive coordinates.

Drawing 1: Each member of the pair should complete their first drawing, taking turns giving instructions and using the tool. These drawings do not feature any overlapping shapes, but students may need to grapple with the fact that circles are drawn from the middle and squares from the top left corner. Additionally, students may just struggle with the direction of the Y-axis.

Discuss: Give pairs a couple of minutes to discuss any common issues they're noticing in trying to complete their drawings.

Drawing 2: Each member of the pair should describe their second drawing to their partner. These drawings feature overlapping shapes and so students will need to consider the order in which shapes are being placed as well as when they should change the color of the pen.

Draw Your Own: If time allows, give students a chance to create their own drawing to communicate to their partner.

Teaching Tip

When to Move On: Determine whether this last activity is worth the time in your schedule. Giving students a chance to create and communicate their own drawing can help reinforce their knowledge, but if students are obviously achieving the learning objectives of the lesson without it then you can also just move to the wrap up to synthesize their learning.

Remarks

When we make images, we need a way to communicate exactly where each shape goes. The coordinate plane helps us to do that. Our coordinate plane has two coordinates, x and y. The x-coordinate tells us how far our shape is from the left of the grid. The y-coordinate tells us how far our shape is from the top of the grid. The black dots on the shapes help you be very specific about how the shape is placed on the grid.

Wrap Up (5 min)

Question of the Day: How can we clearly communicate how to draw something on a screen?

Journaling

Prompt: Have students reflect on each of the following prompts:

- What things were important in communicating about position, color, and order of the shapes in this activity?
- What's a way you have seen similar problems solved in the past?

Discuss: Have pairs share their answers with one another. Then open up the discussion to the whole class.

Remarks

> At the beginning of class we saw that communicating how to draw even simple shapes can be pretty challenging. The grid we learned about today is one solution to this problem but there are many others that could've worked. In fact, a lot of you probably noticed that the grid in Game Lab is "flipped". Computer screens come in all different shapes and sizes, as does the content we show on them. We need to agree on one point where all the content can grow from. Since we read starting at the top left corner, the grid on a computer screen starts at the top left corner as well. There's also the benefit of not having to use any negative numbers to talk about locations on the screen. Don't worry if this flipped grid is a little tricky still. We'll have plenty more time to work on it in coming lessons.

Assessment Opportunity

Students should be able to explain the coordinate system of Game Lab. Make sure the students understand that it was important to note not only where the shape was positioned on the coordinate system, but the exact point of the shape (corner or center) that falls on that point on the grid.

Students should also explain that the code needs to set a color and specify the shape to be drawn, and that the order of the code determines which shape is on top.

The second prompt may be used to reinforce that the grid system students saw today is different from the one they may have seen in a math class. They should see, however, that both solve the same problem. Finally, this discussion can lead into a teacher explanation of why the grid in Game Lab is "flipped" from ones they may have seen previously.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

Lesson 3: Drawing in Game Lab

Overview

Question of the Day: How can we communicate to a computer how to draw shapes on the screen?

Students are introduced to Game Lab, the programming environment for this unit, and begin to use it to position shapes on the screen. They learn the basics of sequencing and debugging, as well as a few simple commands. At the end of the lesson, students will be able to program images like the ones they made with the drawing tool in the previous lesson.

Purpose

The main purpose of this lesson is to give students a chance to get used to the programming environment, as well as the basic sequencing and debugging that they will use throughout the unit. Students begin with an introduction to the GameLab interactive development environment (IDE), then learn the three commands (`rect` , `ellipse` , and `fill`) that they will need to code the same types of images that they created on paper in the previous lesson. Challenge levels provide a chance for students who have more programming experience to further explore Game Lab.

Assessment Opportunities

1. **Use a coordinate system to place elements on the screen.**

   See level 10 in Code Studio, in particular the placement of the square in the picture.

2. **Sequence code correctly to overlay shapes.**

   See level 10 in Code Studio, in particular that the square is displayed in front of the circles.

Agenda

Warm Up (5 minutes)

  Programming Images

Activity (30 minutes)

  Simple Drawing in Game Lab
  Share Drawings

Wrap Up (10 minutes)

View on Code Studio

Objectives

Students will be able to:

- Use a coordinate system to place elements on the screen.
- Sequence code correctly to overlay shapes.

Preparation

- ☐ Read the Forum
- ☐ Prepare projector or other means of showing videos if you wish to watch as a class

Links

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

For the Teachers

- **Drawing in Game Lab** - Slides

Vocabulary

- **Bug** - Part of a program that does not work correctly.
- **Debugging** - Finding and fixing problems in an algorithm or program.
- **Program** - An algorithm that has been coded into something that can be run by a machine.

Introduced Code

- `fill(color)`
- `ellipse(x, y, w, h)`
- `rect(x, y, w, h)`

Teaching Guide

Warm Up (5 minutes)

Programming Images

Prompt: Based on what you know about computers, what do you think will be different between telling a person about your image and telling a computer about your image?

Share: Allow students time to think individually and discuss with a partner, then bring the class together and write their ideas on the board.

Discussion Goal

Goal From Unit 2, students may remember that a computer can only understand what to do if you use a particular language in order to communicate with it. For Web Development, that language was HTML. Computers can't "figure out" what you mean in the same way that a human can, and they are usually much more specific in how they follow instructions.

Remarks

> In order to give instructions to a computer, we need to use a language that a computer understands. In the last unit, we used HTML, which is great for making web pages. To make our animations and games, we will use a version of Javascript that uses blocks. The environment that we'll be programming in is called Game Lab.

Question of the Day: How can we communicate to a computer how to draw shapes on the screen?

Activity (30 minutes)

Simple Drawing in Game Lab

Group: Place students in pairs to program together.

Transition: Send students to Code Studio.

Teaching Tip

Pair programming is a great way to increase student confidence and foster the practices of collaboration and communication. You can read more about how to use Code Studio's pair programming feature **here**.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Introduction to Game Lab 2 ***(click tabs to see student view)***

Video: Drawing in Game Lab - Part 1 3 ***(click tabs to see student view)***

Using the Grid 4 ***(click tabs to see student view)***

Video: Drawing in Game Lab - Part 2 5 ***(click tabs to see student view)***

Skill Building 6 7 8 ***(click tabs to see student view)***

Practice 9 9a 9b 9c

Assessment

10

(click tabs to see student view)

Drawing Challenges

11 | 11a | 11b | 11c | 11d | 11e | 11f | 11g

Support: As students work on the levels, you can help them, but encourage them to try to spend some time figuring things out themselves first. If you need help supporting students, see the exemplars in the teacher answer viewer. When students hit the **challenge levels**, they can choose to pursue one or more of the challenges, return to improve upon previous levels, or help a classmate.

Teaching Tip

Students that are new to programming often have some common misconceptions they run into. In order to prevent those, keep reminding students about the following things:

- One command per line
- Commands run in order from top to bottom
- Order of inputs into shape commands matter
- Each input into shape commands are separated by commas
- (0,0) is in the upper left corner of the display
- All x and y values on the display are positive

Share Drawings

Share: Once students have completed their drawings, have them share with the class. One way to do this is with a gallery walk.

Wrap Up (10 minutes)

Question of the Day: How can we communicate to a computer how to draw shapes on the screen?

Prompt: Today you learned how to draw in Game Lab for the first time. What type of advice would you share with a friend who was going to learn about drawing in Game Lab to make it easier for them?

Share: Allow students to share out their responses with the class.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

Lesson 4: Shapes and Parameters

Overview

Question of the Day: How can we use parameters to give the computer more specific instructions?

In this lesson students continue to develop their familiarity with Game Lab by manipulating the width and height of the shapes they use to draw. The lesson kicks off with a discussion that connects expanded block functionality (e.g. different sized shapes) with the need for more block inputs, or "parameters". Students learn to draw with versions of `ellipse()` and `rect()` that include width and height parameters. They also learn to use the `background()` block.

Purpose

This lesson gives students a chance to slightly expand their drawing skills while continuing to develop general purpose programming skills. They will need to reason about the x-y coordinate plane, consider the order of their code, and slightly increase their programs' complexity. This lesson should be focused primarily on skill-building.

Assessment Opportunities

1. **Use and reason about drawing commands with multiple parameters**

   See level 8 in Code Studio.

Agenda

Warm Up (5 min)

  Shapes of Different Sizes

Activity (40 min)

  Programming with Parameters

Wrap Up (5 min)

  Journal

View on Code Studio

Objectives

Students will be able to:

- Use and reason about drawing commands with multiple parameters

Preparation

- [ ] Review the level sequence in Code Studio

Links

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

For the Teachers

- **Shapes and Parameters** - Slides

Vocabulary

- **Parameter** - An extra piece of information passed to a function to customize it for a specific need

Introduced Code

- `background(color)`
- `ellipse(x, y, w, h)`
- `rect(x, y, w, h)`

Teaching Guide

Warm Up (5 min)

1. `ellipse()`
2. `rect()`

Shapes of Different Sizes

Prompt: The `rect` block has two inputs that control where it's drawn - the x and y position. If you wanted these commands to draw different sizes of rectangles, what additional inputs would you need to give these blocks?

Discuss: Students should brainstorm ideas silently, then share with a neighbor, then share out with the whole class. Record ideas as students share them on the board.

Discussion Goal

Goal: This discussion introduces the vocabulary word "parameter" and also helps students understand the need for parameters. Students will be seeing versions of the `ellipse()` and `rect()` block in this lesson that have additional parameters. Students may say they want inputs for the size of the shapes, their color, etc. During this conversation tie the behaviors the students want to the inputs the block would need. For example, if you want rectangles to be a different size, the block will need an input that lets the programmer decide how large to make it.

Remarks

> If we want our blocks to draw shapes in different ways they'll need more inputs that let us tell them how to draw. The inputs or openings in our blocks have a formal name, parameters, and today we're going to be learning more about how to use them.

Key Vocabulary: Parameter - An extra piece of information passed to a function to customize it for a specific need

Question of the Day: How can we use parameters to give the computer more specific instructions?

Activity (40 min)

Programming with Parameters

Group: Put students into pairs for the programming activity.

Transition: Move students onto Code Studio.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Skill Building 2 3 4 5 6 ***(click tabs to see student view)***

Practice with Parameters 7 7a 7b 7c

Assessment ✓ 8 ***(click tabs to see student view)***

Shapes and Parameters Challenges 9 9a 9b 9c 9d 9e

Wrap Up (5 min)

Journal

Question of the Day: How can we use parameters to give the computer more specific instructions?

Prompt: You use parameters to control your shape's location and size. Can you think of any other situations in which parameters might be useful?

Share: Allow students to share our their ideas.

Discussion Goal

Student answers will vary, but they should all follow the pattern of giving more specific information for how to carry out a task. If students have trouble thinking of something, you may want to give some examples for things a computer could do, such as ring an alarm (with a parameter for a certain time) or play a song (with a parameter for the song title).

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▸ **AP** - Algorithms & Programming

Lesson 5: Variables

Overview

Question of the Day: How can we use variables to store information in our programs?

In this lesson students learn how to use variables to label a number. Students begin the lesson with a very basic description of the purpose of a variable within the context of the storage component of the input-output-storage-processing model. Students then complete a level progression that reinforces the model of a variable as a way to label or name a number.

Purpose

This lesson is the first time students will see variables in the course, and they are not expected to fully understand how variables work by its conclusion. Students should leave this lesson knowing that variables are a way to label a value in their programs so that they can be reused or referenced later. In the following lesson students will be introduced to random numbers, in which they will see a more powerful use for variables.

Using variables to manipulate drawings is a surprisingly challenging skill that requires a great deal of forethought and planning. While students will use or modify many programs in this lesson, they are not expected to compose programs that use variables to modify the features of a drawing. In later lessons, students will expand their understanding of variables and more advanced ways they can be used.

Assessment Opportunities

1. **Identify a variable as a way to label and reference a value in a program**

   See the reflection prompt in the Wrap Up.

2. **Use variables in a program to store a piece of information that is used multiple times**

   See Level 8 in Code Studio.

Agenda

Warm Up (10 Mins)

- **Input-Output-Storage-Processing**

Activity (30 Mins)

- **Programming with Variables**

Wrap Up (5 Mins)

- **Reflection**

View on Code Studio

Objectives

Students will be able to:

- Identify a variable as a way to label and reference a value in a program
- Use variables in a program to store a piece of information that is used multiple times

Preparation

☐ Review the level progression in Code Studio

Links

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

For the Teachers

- **Variables** - Slides

For the Students

- **Introduction to Variables** - Video (**download**)

Vocabulary

- **Variable** - A label for a piece of information used in a program.

Introduced Code

- `Declare and assign a value to a variable`
- `Declare a variable`

Teaching Guide

Warm Up (10 Mins)

Input-Output-Storage-Processing

Prompt: At the beginning of the course, we learned that input, output, storage, and processing were common to all computers. Where do you see input, output, storage, and processing in Game Lab?

Share: Allow students to share out their answers.

Remarks

> Today we're going to focus on storage. We're going to look at variables, which are a very common way for computers to store information in a program.

Discussion Goal

Allow students to share out their different ideas, but eventually bring the conversation back to storage to tie into the lesson topic of variables. Students' answers may include:

- input: values passed as parameters, typing into the Game Lab workspace
- output: shapes shown on the Game Lab screen
- storage: remembering the code
- processing: the If/then and matching that turn the code into the pictures on the screen

Key Vocabulary: variable - a label for a piece of information used in a program

Question of the Day: How can we use variables to store information in our programs?

Activity (30 Mins)

Programming with Variables

Transition: Send students to Code Studio.

Code Studio levels

Lesson Overview 1 *(click tabs to see student view)*

Prediction 2 *(click tabs to see student view)*

Video: Introduction to Variables 3 *(click tabs to see student view)*

Skill Building 4 5 6 *(click tabs to see student view)*

Variables Practice 7 7a 7b 7c 7d

Assessment 8 *(click tabs to see student view)*

Variables Challenges 9 9a 9b 9c 9d

Wrap Up (5 Mins)

Reflection

Question of the Day: How can we use variables to store information in our programs?

Prompt: Give students the following prompts:

- What is your own definition of a variable?
- Why are variables useful in programs?

Discuss: Have students silently write their ideas before sharing in pairs and then as a whole group.

Assessment Opportunity

Use this discussion to assess students' mental models of a variable. You may wish to have students write their responses so you can collect them to review later. You should be looking to see primarily that they understand that variables can label or name a number so that it can be used later in their programs. While there are other properties of a variable students may have learned, this is the most important thing they should understand before moving on to the next lesson.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

Lesson 6: Random Numbers

Overview

Question of the Day: How can we make our programs behave differently each time they are run?

Students are introduced to the `randomNumber()` block and how it can be used to create new behaviors in their programs. They them learn how to update variables during a program. Combining all of these skills, students draw randomized images.

Purpose

This lesson introduces randomness, which is important both as a way to make programs more interesting, but also to motivate the use of variables. In the middle of the activity, students are exposed to a variable that is updated multiple times in the program, expanding their understanding of how variables can be used.

Assessment Opportunities

1. Generate and use random numbers in a program

See level 7 in Code Studio.

2. Update a value stored in a variable

See level 5 in Code Studio. Check that students have updated the value of "petalSize" between drawing the two flowers.

Agenda

View on Code Studio

Objectives

Students will be able to:

- Generate and use random numbers in a program
- Update a value stored in a variable

Preparation

Review the level progression in Code Studio

Links

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

For the Teachers

- **Random Numbers** - Slides

Introduced Code

- `randomNumber()`

Teaching Guide

Warm Up

Prompt: So far, our programs have done the same thing every time that we run them. Are there any times that you'd want a program to do something differently each time it was run?

Discuss: Allow students time to write down some ideas, then discuss as a group.

Discussion Goal

The goal of this discussion is to set context for the introduction of random numbers. Students may come up with various ideas related to user interaction or gathering input from other sources. Allow them to discuss the different ideas that they have, but eventually turn the conversation to the idea of randomness.

Remarks

> So far, we've wanted our programs to do exactly as we've coded, and most of our surprises have been bugs. Today we're going to look at how we can code random behaviors into our programs so that we can get some good surprises.

Question of the Day: How can we make our programs behave differently each time they are run?

Activity

Programming Images

Transition: Move students onto Code Studio

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Exploration 2 ***(click tabs to see student view)***

Skill Building 3 4 5 ***(click tabs to see student view)***

Practice with Random Numbers 6 6a 6b

Assessment 7 ***(click tabs to see student view)***

Challenges: Random Numbers 8 8a 8b 8c 8d 8e 8f

Share: If some students have taken extra time to work on their projects, give them a chance to share their more complex rainbow snakes. Focus conversation on which parameters students are manipulating or randomizing to create their drawings.

Wrap Up

Question of the Day: How can we make our programs behave differently each time they are run?

Prompt: So far, we've only looked at random numbers. Are there any other things that you might like to be random in your program?

Share: Allow students to share out what sorts of random things they might like in their programs.

Discussion Goal

The discussion is intended to have students think about the wider implications of randomness in games and other programs. Although there is no block to general random data other than numbers, in later lessons students will learn techniques that will allow them to use random numbers to randomly choose from a variety of behaviors.

Lesson 7: Sprites

Overview

Question of the Day: How can we use sprites to help us keep track of lots of information in our programs?

In order to create more interesting and detailed images, students are introduced to the sprite object. The lesson starts with a discussion of the various information that programs must keep track of, then presents sprites as a way to keep track of that information. Students then learn how to assign each sprite an image, which will greatly increase the complexity of what they can draw on the screen.

Purpose

Keeping track of many shapes and the different variables that control aspects of those shapes can get very complex. There will be lots of variables with different variable names. Instead, computer scientists created something called an **object** which allows for one variable name to control both the shape and all its aspects. In Game Lab we use a certain type of object called a **sprite**. A sprite is just a rectangle with **properties** for controlling its look. Properties are the variables that are attached to a sprite. You can access them through **dot notation**.

Using the Animation Tab, students can create or import images to be used with their sprites. Later on, these sprites will become a useful tool for creating animations, as their properties can be changed and updated throughout the course of a program.

Assessment Opportunities

1. **Create and use a sprite**

   See levels 12 and 17 on Code Studio.

2. **Use dot notation to update a sprite's properties**

   See level 17 on Code Studio.

Agenda

Warm Up (5 minutes)

   How Much Information?

Activity

   Introduction to Sprites

Wrap Up (5-10 min)

   Journal

View on Code Studio

Objectives

Students will be able to:

- Create and use a sprite

Preparation

- ☐ (Optional) Print a copy of the activity guide for each student

Links

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

For the Teachers

- **Sprites** - Slides

Vocabulary

- **Dot notation** - the way that sprites' properties are used in Game Lab, by connecting the sprite and property with a dot.
- **Property** - A label for a characteristic of a sprite, such as its location and appearance
- **Sprite** - A character on the screen with properties that describe its location, movement, and look.

Introduced Code

- `drawSprites(group)`
- `var sprite = createSprite(x, y, width, height)`

Teaching Guide

Warm Up (5 minutes)

How Much Information?

Review: So far we've written programs that put simple shapes on the screen. List of all of the different pieces of information that you have used to control *how* these shapes are drawn.

Prompt: If you wanted to create programs with more detailed images, maybe even characters that you could interact with, what other pieces of information might you need in your code?

Share: Allow students to share out their lists.

Discussion Goal

The goal here is to get students thinking about all of the different values that go into drawing a single shape on the screen, and how many more values they may need to control a more detailed character in a program. If students are struggling to come up with ideas, you might use some of the following prompts: *How do you tell a shape where to go on the screen?* How do you tell a shape what size it needs to be? *How do you tell a shape what color it should be? What about its outline?* What if you wanted to change any of those values during your program, or control other things like rotation?

Remarks

> Today we'll learn how to create characters in our animations called **sprites**. These sprites will help us keep track of all of the information that we need in our programs.

Question of the Day: How can we use sprites to help us keep track of lots of information in our programs?

Activity

Introduction to Sprites

Transition: Send students to Code Studio

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Exploration 2 ***(click tabs to see student view)***

Video: Introduction to Sprites 3 ***(click tabs to see student view)***

Skill Building 4 5 6 ***(click tabs to see student view)***

Video: The Animation Tab 7 ***(click tabs to see student view)***

Skill Building 8 9 ***(click tabs to see student view)***

Creating sprites and animations 10 10a 10b 10c 10d 10e 10f

Assessment

✓ 11 *(click tabs to see student view)*

Challenges: Sprites and Animations

12 | 12a | 12b | 12c

Wrap Up (5-10 min)

Key Vocabulary:

- **Sprite** - A character on the screen with properties that describe its location, movement, and look.
- **Property** - A label for a characteristic of a sprite, such as its location and appearance.
- **Dot Notation** - the way that sprites' properties are used in Game Lab, by connecting the sprite and property with a dot.

Journal

Question of the Day: How can we use sprites to help us keep track of lots of information in our programs?

Prompt: So far we've been able to chance a sprite's location and image. What else might you want to change about your sprites?

Share: Allow students to share out their ideas.

Discussion Goal

This discussion prompts students to think about the different properties that a sprite might have, and prepares them for the next lesson, which explicitly covers sprite properties.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▸ **AP** - Algorithms & Programming

Lesson 8: Sprite Properties

Overview

Question of the Day: How can we use sprite properties to change their appearance on the screen?

Students extend their understanding of sprites by interacting with sprite properties. Students start with a review of what a sprite is, then move on to Game Lab to practice more with sprites, using their properties to change their appearance. They then reflect on the connections between properties and variables.

Purpose

In the last lesson, when students were introduced to sprites, they focused mainly on creating a sprite and assigning it an animation. This lesson starts to dig into what makes sprites such a powerful programming construct--that they have properties that can be modified as a program is running. This lays the foundation for much of what students will be doing in the rest of the unit in terms of accessing and manipulating sprite properties to create interesting behaviors in their programs.

Assessment Opportunities

Use dot notation to update a sprite's properties

See Code Studio level 6.

Agenda

Warm Up
Activity
Wrap Up

View on Code Studio

Objectives

Students will be able to:

- Use dot notation to update a sprite's properties

Links

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

For the Teachers

- **Sprite Properties** - Slides

Vocabulary

- **Property** - A label for a characteristic of a sprite, such as its location and appearance

Introduced Code

- `sprite.rotation`
- `sprite.scale`
- `sprite.x`
- `sprite.y`

Teaching Guide

Warm Up

Prompt: What is your definition of a sprite? What sprite properties do you know how to use? What other sprite properties might be useful?

Allow students time to reflect on their own and then with a partner before sharing out to the entire group. It's okay if students do not have a canonical definition of a sprite, but they should recognize that a sprite is a part of the program that has several different properties that control its location and appearance.

Remarks

> So far, we've only been able to control our sprite's location and animation, but today, we're going to learn how to update other sprite properties so we can make even better programs.

Question of the Day: How can we use sprite properties to change their appearance on the screen?

Activity

Group: Put students in pairs.

Transition: Send students to Code Studio.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Prediction 2 ***(click tabs to see student view)***

Skill Building 3 4 ***(click tabs to see student view)***

Sprite Properties 5 5a 5b

Assessment ✓ 6 ***(click tabs to see student view)***

Challenges: Sprite Properties 7 7a 7b

Wrap Up

Question of the Day: How can we use sprite properties to change their appearance on the screen?

Journal Prompt: What is one way sprite properties are the same as variables? What's one way that sprite properties are different from variables?

Discuss: Allow students to discuss in pairs or small groups before sharing out to the entire group.

Discussion Goal

Students may note that sprite properties and variables are similar in that they both store information. They are different in that variables can be anything, but sprites have particular properties that are used in certain ways on the screen.

Lesson 9: Text

Overview

Question of the Day: How can we use text to improve our scenes and animations?

This lesson introduces Game Lab's text commands, giving students more practice using the coordinate plane and parameters. At the beginning of the lesson, students are asked to caption a cartoon created in Game Lab. They then move onto Code Studio where they practice placing text on the screen and controlling other text properties, such as size. Students who complete the assessment early may go on to learn more challenging blocks related to text properties.

Purpose

This lesson introduces text, which students will need as they begin to build more complex programs (e.g. games with scoreboards). This is the last type of element that students will be placing on the screen. After this, students will focus on how they can control the movement and interactions of these elements.

Agenda

View on Code Studio

Objectives

Students will be able to:

- Place text on the screen using a coordinate plane.
- Use arguments to control how text is displayed on a screen.

Links

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

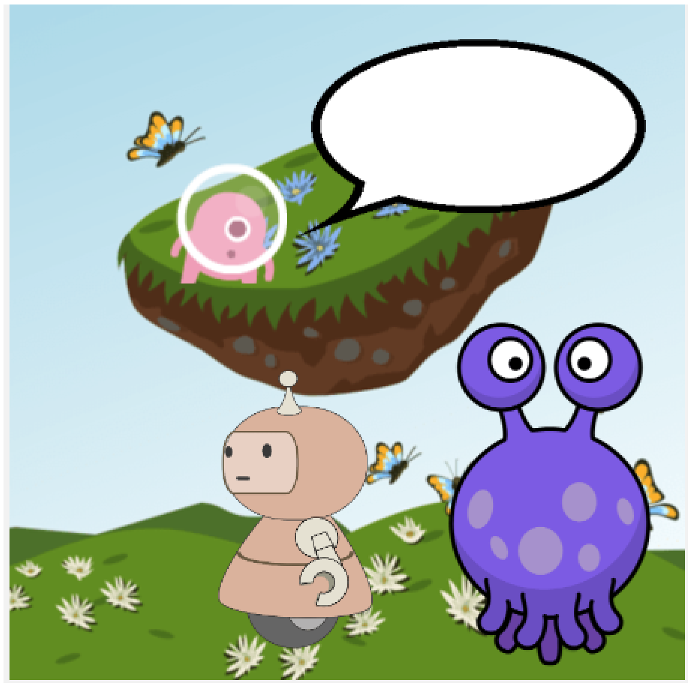
For the Teachers

- **Text** - Slides

Teaching Guide

Warm Up

Journal



Display: Display the cartoon for students to see.

Prompt: Look at the cartoon that was made in Game Lab. What do you think the alien should be saying?

Share: Allow volunteers to share out their ideas.

Remarks

> We've had a lot of fun drawing things and using our sprites, but there's been one thing missing from our Game Lab pictures: text! Today we're going to learn how to add text to Game Lab projects.

Question of the Day: How can we use text to improve our scenes and animations?

Activity

Transition Send students to Code Studio.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Prediction 2 ***(click tabs to see student view)***

Skill Building 3 4 ***(click tabs to see student view)***

Practice 5 5a 5b

Assessment

6 *(click tabs to see student view)*

Text Challenges

7 7a 7b 7c 7d

Wrap up

Journal

Question of the Day: How can we use text to improve our scenes and animations?

Prompt: You've drawn with both text and shapes on the screen.

- What are two ways drawing with text is similar to drawing shapes?
- What is one way that drawing with text is different from drawing with shapes?

Share: If there is time, allow students to share out their answers.

Discussion Goal

Student answers to the question may vary, but some similarities are that they both use the coordinate plane, and that they are drawn automatically, unlike sprites, which must use the `drawSprites` command. One possible difference that it is more difficult to control the exact size of text, since the amount of text and font size are not as specific as height and width parameters.

Lesson 10: Mini-Project - Captioned Scenes

Overview

Question of the Day: How can we use Game Lab to express our creativity?

After a quick review of the code they have learned so far, students are introduced to their first creative project of the unit. Using the problem-solving process as a model, students define the scene that they want to create, prepare by thinking of the different code they will need, try their plan in Game Lab, then reflect on what they have created. They also have a chance to share their creations with their peers.

Purpose

This lesson is a chance for students to get more creative with what they have learned. Some students may spend more time in the animation tab drawing than programming. Encourage students to spend time on parts of the activity that interest them, as long as they meet the requirements of the assignment.

The open-ended nature of this lesson also provides flexibility for the teacher to decide how long students should spend on their work, depending on the scheduling demands of the particular course implementation.

Agenda

Warm Up
- **Review**

Activity
- **Gallery Walk**

Wrap up
- **Journal**

View on Code Studio

Objectives

Students will be able to:

- Use a structured process to plan and develop a program.

Links

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

For the Teachers

- **Mini Project - Captioned Scenes** - Exemplar
- **Mini-Project: Captioned Scenes** - Slides

For the Students

- **Captioned Scenes** - Rubric Make a Copy
- **Sprite Scene Planning** - Activity Guide Make a Copy

Teaching Guide

Warm Up

Review

Prompt: Write down as many blocks as you can remember from Game Lab. Make sure you know what each one does, especially the inputs, or parameters, for each of the blocks.

Share: Allow students to share out what they remember as a group review.

Discussion Goal

The goal of this discussion is to review the different blocks and skills that students have learned. As students talk about the different blocks, try to steer the conversation into how they can be used creatively, to link into the main topic of the day.

Remarks

> You've already learned how to do some really great things in Game Lab. Today you'll have a chance to put them all together to make an interesting scene to share with the world. That means instead of trying to recreate someone else's idea, you're going to get to come up with an idea of your own, so it's time to get creative!

Question of the Day: How can we use Game Lab to express our creativity?

Activity

Distribute: (Optional) pass out copies of the activity guide. Students can use this sheet to plan out the scene they create at the end of this lesson, but the planning can also be completed on scratch paper.

Transition Send students to Code Studio.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Sprite Scenes 2 ***(click tabs to see student view)***

Create a Background 3 ***(click tabs to see student view)***

Add Sprites 4 ***(click tabs to see student view)***

Add Text 5 ***(click tabs to see student view)***

Review Your Scene 6 ***(click tabs to see student view)***

Gallery Walk

Allow students to walk around the room and see the pictures that each of their classmates has coded. Celebrate all of the different ideas that students were able to implement with the same basic code.

Teaching Tip

You may choose to formalize this process by having each student write down one positive quality of each project, or having students "draw names" to comment on one particular classmate's work.

Wrap up

Journal

Question of the Day: How can we use Game Lab to express our creativity?

Prompt: What was one especially creative way you saw someone else use the blocks today?

Share: Have students share out what they appreciated about their classmates' projects. You may want to do this "popcorn" style, with each student who responds choosing the next person to share.

Discussion Goal

This discussion should serve as a celebration of what the students have accomplished. As students share out what they have seen, encourage them to learn from each other and ask questions if they were not sure how to do something. Highlight how students were able to do very different things with the same tool.

Lesson 11: The Draw Loop

Overview

Question of the Day: How can we animate our images in Game Lab?

In this lesson students are introduced to the draw loop, one of the core programming paradigms in Game Lab. To begin the lesson students look at some physical flipbooks to see that having many frames with different images creates the impression of motion. Students then watch a video explaining how the draw loop in Game Lab helps to create this same impression in their programs. Students combine the draw loop with random numbers to manipulate some simple animations with dots and then with sprites.

Purpose

The draw loop is a core component of Game Lab. The fact that the Game Lab environment repeatedly calls this function many times a second (by default 30) is what allows the tool to create animations. This lesson has two goals. The first is for students to see how animation in general depends on showing many slightly different images in a sequence. The second goal is for students to understand how the draw loop allows them to create this behavior in Game Lab. Students should leave the lesson understanding that the commands in the draw loop are called after all other code but are then called repeatedly to create animation. Students will have a chance to continue to develop an understanding of this behavior in the next two lessons, but laying a strong conceptual foundation in this lesson will serve them well for the rest of the unit.

Assessment Opportunities

1. **Explain how the draw loop allows for the creation of animations in Game Lab**

   In the wrap up discussion, check that students mention that the code in the draw loop is run over and over, whereas code outside the draw loop is just run once, at the beginning of the program.

2. **Use the draw loop in combination with the randomNumber() command, shapes, and sprites to make simple animations**

   See level 9 in Code Studio.

Agenda

Warm Up (5 mins)
Activity (60 min)
Wrap Up (10 mins)

View on Code Studio

Objectives

Students will be able to:

- Explain how the draw loop allows for the creation of animations in Game Lab
- Use the draw loop in combination with the randomNumber() command, shapes, and sprites to make simple animations

Preparation

- ☐ Print and assemble the manipulatives
- ☐ Prepare the video

Links

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

For the Teachers

- **The Draw Loop** - Slides
- **Random Dot Flipbook** - Manipulative
- **Random Sprite Flipbook** - Manipulative
- **Flipbook Example - by Marnic Bos** - Video

Introduced Code

- `function draw() { }`
- `World.frameRate`

Teaching Guide

Warm Up (5 mins)

Display: Show **Flipbook Example - by Marnic Bos - Video**. (Video by Marnic Bos on Youtube)

Prompt: This video shows a flipbook to make animation. In your own words, how is it working? Why does it "trick our eyes" into thinking something is moving?

Discuss: Have students write their ideas independently, then share with partners, then share as a whole group.

Discussion Goal

Goal: This discussion should introduce some key understandings about animation. Students should understand that the key is seeing many pictures in a row that are slightly different. Introduce the vocabulary word "frame" as being one of those many pictures--a single image within an animation. Then transition to the fact that soon students will be creating animations of their own.

Remarks

> We're going to start learning how to make animations, not just still images. In order to do this we need a way to make our programs draw many pictures a second. Our eyes will blur them together to make it look like smooth motion. To do this, though, we're going to need to learn an important new tool.

Question of the Day: How can we animate our images in Game Lab?

Activity (60 min)

Video: Watch the video introducing the draw loop.

Demo: If you choose, use the provided manipulatives to provide a hands-on demo for exploring how the draw loop and animations are connected.

Teaching Tip

Pair Programming: This lesson introduces a challenging new paradigm. Consider having students pair program. Remember to give students clear instructions on when to switch driver and navigator.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Video: Introduction to the Draw Loop 2 ***(click tabs to see student view)***

Exploration 3 ***(click tabs to see student view)***

Skill Building 4 ***(click tabs to see student view)***

Prediction 5 ***(click tabs to see student view)***

Skill Building 6 7 ***(click tabs to see student view)***

Practice 8 8a 8b 8c 8d

Assessment

✓ 9 *(click tabs to see student view)*

Draw Loop Challenges

10 10a 10b

Wrap Up (10 mins)

Question of the Day: How can we animate our images in Game Lab?

Prompt: How does the draw loop help us make animations?

Review: Return to the resources students saw at the beginning of the lesson (Video, physical flip books) or under Help and Tips during the lesson. Address misconceptions that have arisen in the lesson.

Assessment Opportunity

Key Understandings: There are many common misconceptions with the draw loop. Make sure students understand the following:

- The draw loop is run after all other code in your program. It does not actually matter where it is located in your program.
- The draw loop is run by Game Lab at a constant frame rate of 30 frames per second. You do not actually need to call the function yourself.
- The "frames" in Game Lab can be thought of as transparency sheets. Unless you draw a background, then all your new shapes or sprites will simply appear on top of your old ones.
- You should only have one draw loop in your program.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 12: Sprite Movement

Overview

Question of the Day: How can we control sprite movement in Game Lab?

In this lesson, students learn how to control sprite movement using a construct called the counter pattern, which incrementally changes a sprite's properties. Students first brainstorm different ways that they could animate sprites by controlling their properties, then explore the counter pattern in Code Studio. After examining working code, students try using the counter pattern to create various types of sprite movements.

Purpose

This lesson builds on the draw loop that students learned previously to create programs with *purposeful* motion. By either incrementing or decrementing sprite properties, such as `sprite.x`, students can write programs that move sprites in expected patterns, instead of the randomization that they used in the past. The animations that students learn to create in this lesson lay the foundation for all of the animations and games that they will make throughout the rest of the unit.

Assessment Opportunities

1. **Use the counter pattern to increment or decrement sprite properties**

   See Level 8 in Code Studio.

2. **Identify which sprite properties need to be changed, and in what way, to achieve a specific movement**

   See Level 8 in Code Studio.

Agenda

Warm Up (5 minutes)

  Reviewing Sprite Properties

Activity (40 minutes)

  Levels: Sprites and Images

Wrap Up (5 minutes)

  Journal

View on Code Studio

Objectives

Students will be able to:

- Use the counter pattern to increment or decrement sprite properties
- Identify which sprite properties need to be changed, and in what way, to achieve a specific movement

Links

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

For the Teachers

- **Sprite Movement** - Slides

For the Students

- **Animating Sprites** - Video (**download**)

Teaching Guide

Warm Up (5 minutes)

Reviewing Sprite Properties

Review: On a piece of scratch paper, list out all of the sprite properties you can think of and what aspect of a sprite they affect.

Prompt: What kinds of animations could you make if you could control these properties? Consider adding and subtracting from properties, or even updating multiple properties at the same time.

Discuss: Allow students to share their ideas with a partner before sharing with the entire class.

Question of the Day: How can we control sprite movement in Game Lab?

Discussion Goal

The purpose of this discussion is to start students thinking about how they might use the various sprite properties they've seen so far to make animations with purposeful motion. If students struggle to come up with ideas, you can narrow down the question to specific properties. For example:

- What would happen to a sprite if you constantly increased its `x` property?
- What would happen to a sprite if you constantly increased its `y` property?
- What about other properties, or combining multiple properties?

Activity (40 minutes)

Levels: Sprites and Images

Transition: Send students to Code Studio.

Code Studio levels

Lesson Overview 1 *(click tabs to see student view)*

Prediction 2 *(click tabs to see student view)*

Video: Sprite Movement 3 *(click tabs to see student view)*

Skill Building 4 5 6 7 *(click tabs to see student view)*

Practice 8 8a 8b 8c

Assessment 9 *(click tabs to see student view)*

Sprite Movement Challenges 10 10a 10b 10c

Wrap Up (5 minutes)

Journal

Question of the Day: How can we control sprite movement in Game Lab?

Prompt: You've seen two ways to create animations with the draw loop: random numbers and the counter pattern. What is one type of movement that you'd want to use random numbers for? What is one type of movement that you would want to use the counter pattern for? Are there any movements that might combine the counter pattern and random numbers?

Discuss: Allow students to discuss with a partner before sharing with the entire class.

Discussion Goal

Student answers may vary, but movements such as shaking use random numbers, and movements that are more purposeful use the counter pattern. It's less important that students have a specific answer than that their answers reflect a growing understanding of how the different programming constructs work.

Students may have seen random numbers used on one property (rotation) and the counter pattern used on another (x position) in one of the challenges, but if there is time, encourage them to think about what would happen if you used the counter pattern to add or subtract a random number to a sprite property, or to combine the two constructs in other ways.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

Lesson 13: Mini-Project - Animation

Overview

Question of the Day: How can we combine different programming patterns to make a complete animation?

In this lesson, students are asked to combine different methods that they have learned to create an animated scene. Students first review the types of movement and animation that they have learned, and brainstorm what types of scenes might need that movement. They then begin to plan out their own animated scenes, which they create in Game Lab.

Purpose

This is a chance for students to get more creative with what they have learned. Some students may spend more time in the animation tab drawing than programming. Encourage students to spend time on parts of the activity that interest them, as long as they meet the requirements of the assignment.

Agenda

View on Code Studio

Links

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

For the Teachers

- **Animated Scene Planning** - Exemplar
- **Mini-Project: Animation** - Slides

For the Students

- **Animated Scene** - Rubric Make a Copy
- **Animated Scene** - Activity Guide Make a Copy

Teaching Guide

Warm Up

Review

Prompt: Write down as many types of movement and animations as you can remember from the previous lesson. Make sure you know what blocks and patterns you need to make those movements, and when those movements would be useful.

Share: Allow students to share out what they remember as a group review.

Discussion Goal

The goal of this discussion is to review the different types of movement and animations that students have learned. As students talk about the different methods, make sure that they are mentioning why that type of movement would be useful. Press them to be specific about what they might animate using the various methods.

Remarks

> Now that we can control the way that our sprites move a little better, we're going to have a chance to put everything together to make an animation from scratch.

Question of the Day: How can we combine different programming patterns to make a complete animation?

Activity

Distribute: (Optional) pass out copies of the activity guide. Students can use this sheet to plan out the animation they create at the end of this lesson, but the planning can also be completed on scratch paper.

Transition Send students to Code Studio.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Example Animated Scene 2 ***(click tabs to see student view)***

Plan Your Scene 3 ***(click tabs to see student view)***

Draw a Background 4 ***(click tabs to see student view)***

Add Sprites 5 ***(click tabs to see student view)***

Add Text 6 ***(click tabs to see student view)***

Add Movement 7 ***(click tabs to see student view)***

Review Your Animated Scene 8 ***(click tabs to see student view)***

Gallery Walk

Allow students to walk around the room and see the pictures that each of their classmates has coded. Celebrate all of the different ideas that students were able to implement with the same basic code.

Teaching Tip

You may choose to formalize this process by having each student write down one positive quality of each project, or having students "draw names" to comment on one particular classmate's work.

Wrap up

Journal

Question of the Day: How can we combine different programming patterns to make a complete animation?

Prompt: What was one interesting way that you saw sprite movement used today?

Share: Have students share out what they appreciated about their classmates' projects. You may want to do this "popcorn" style, with each student who responds choosing the next person to share.

Discussion Goal

This discussion should serve as a celebration of what the students have accomplished. As students share out what they have seen, encourage them to learn from each other and ask questions if they were not sure how to do something. Highlight how students were able to do very different things with the same tool.

Lesson 14: Conditionals

Overview

Question of the Day: How can programs react to changes as they are running?

This lesson introduces booleans and conditionals, which allow a program to run differently depending on whether a condition is true. Students start by playing a short game in which they respond according to whether particular conditions are met. They then move to Code Studio, where they learn how the computer evaluates Boolean expressions, and how they can be used to structure a program.

Purpose

This lesson follows closely the booleans model that students first experienced in the Booleans Unplugged lesson. As before, we start with using booleans directly before using booleans to trigger *if* statements. In the following lesson we will introduce some boolean producing blocks, such as `keyDown()` , which can be used in place of simple boolean comparisons to write programs that respond to user input.

Assessment Opportunities

1. **Use conditionals to react to changes in variables and sprite properties**

   See Level 10 in Code Studio.

Agenda

Warm Up (5 min)
 Introducing Conditionals
Activity (40 min)
 Conditionals
Wrap Up (5 min)
 Considering Conditions

View on Code Studio

Objectives

Students will be able to:

- Use conditionals to react to changes in variables and sprite properties

Links

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

For the Teachers

- **Conditionals** - Slides

Vocabulary

- **Boolean Expression** - in programming, an expression that evaluates to True or False.
- **Condition** - Something a program checks to see whether it is true before deciding to take an action.
- **Conditionals** - Statements that only run when certain conditions are true.

Introduced Code

- `If statement`
- `Equality operator`
- `Inequality operator`
- `Greater than operator`
- `Greater than or equal operator`
- `Less than operator`
- `Less than or equal operator`

Teaching Guide

Warm Up (5 min)

Introducing Conditionals

Display: Display the following questions on the board and ask students to answer them.

1. If your last name has more than five letters, draw a square on your paper.
2. If your last name less than seven letters, draw a circle.
3. If you are wearing anything green, add 3 + 2.
4. If the teacher is tapping their pencil, draw an 'X'.
5. If the teacher is in the front of the room, fold your paper in half.

Teaching Tip

The last two prompts ask students to react to information that may change as they are answering the questions. In order to drive this point home, consider moving from the front of the room and back, as well as alternately tapping and not tapping a pencil, so that students realize that their answers may change according to when they are answering each question.

Prompt: When we program, we give the computer instructions on what to do. How are the instructions you just followed different from the instructions that we have been giving in Game Lab?

Think-Pair-Share: After students have had a chance to write down their answers, allow them to talk to a partner before sharing out as a class.

Discussion Goal

This discussion should prepare students to investigate programs that react to changing conditions. Students should realize that there is no "one way" that these questions will be answered, and that the answers will change according to changing conditions.

Remarks

> These instructions were a little different because we first had to decide whether something was true or not before we knew what we should do. In programming, instructions that depend on whether or not something is true are called **conditionals**, and the thing that is checked is called the **condition**. Conditionals are especially useful when we want the program to react to situations that change while the program is running.

Key Vocabulary:

- **Condition** - Something a program checks to see whether it is true before deciding whether to take an action.
- **Conditionals** - Statements that only run under certain conditions.

Question of the Day: How can programs react to changes as they are running?

Activity (40 min)

Conditionals

Transition: Send students to Code Studio.

Content Corner

Though seemingly simple, understanding how a boolean statement will evaluate can be difficult given that different programming languages have differing opinions on 'truthiness' and 'falsiness'. In fact, JavaScript (the language used in this course) has two different operators to test boolean equality `==` and `===`.

The double equals operator (`==`) is pretty generous in determining truthiness. For example, each of the following is considered `true` in JavaScript when using the `==` operator, but would be `false` using the `===` operator:

```
1 == true;
"1" == true;
5 == "5";
null == undefined;
"" == false;
```

We use the `==` operator in this course because it's more forgiving, but it's important to be aware that it can sometimes report back truth when you really didn't intend it to (in which case you might want to use the more strict `===` operator)

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Prediction 2 ***(click tabs to see student view)***

Video: Booleans 3 ***(click tabs to see student view)***

Quick Check 4 ***(click tabs to see student view)***

Skill Building 5 6 ***(click tabs to see student view)***

Video: Conditional Statements 7 ***(click tabs to see student view)***

Skill Building 8 ***(click tabs to see student view)***

Conditionals Practice 9 9a 9b 9c

Assessment 10 ***(click tabs to see student view)***

Conditionals Challenges 11 11a 11b

Wrap Up (5 min)

Considering Conditions

Question of the Day: How can programs react to changes as they are running?

Key Vocabulary:

- **Condition** - Something a program checks to see whether it is true before deciding to take an action.
- **Conditionals** - Statements that only run under certain conditions.
- **Boolean Expression** - In programming, an expression that evaluates to True or False.

Prompt: Now that you know how conditionals work, where you do think that they are used in games or other programs and apps that you already use?

Discuss: Have students share responses.

Discussion Goal

This discussion should get students thinking about how conditionals are used in games they have already seen, and help them connect those ideas to how they might want to use conditionals in their own programs. Student responses might include:

- If my username and password are correct, log me into Facebook
- If Pacman has collected all the balls, start the next level
- If my keyboard or mouse hasn't moved in 10 minutes, turn on the screensaver

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 15: Keyboard Input

Overview

Question of the Day: How can our programs react to user input?

Following the introduction to booleans and *if* statements in the previous lesson, students are introduced to a new block called `keyDown()` which returns a boolean and can be used in conditionals statements to move sprites around the screen. By the end of this lesson, students will have written programs that take keyboard input from the user to control sprites on the screen.

Purpose

One common way conditionals are used is to check for different types of user input, especially key presses. Having a way for a user to interact with a program makes it more interesting and dynamic. Without interaction from the user it is very difficult to create a game. Therefore the introduction of conditionals and user inputs for decision making is a critical step toward creating games.

Assessment Opportunities

1. **Use conditionals to react to keyboard input**

   See Level 7 in Code Studio.

2. **Move sprites in response to keyboard input**

   See Level 7 in Code Studio.

Agenda

Warm Up (5 min)
- **Taking Input**

Activity (40 min)
- **Keyboard Input**

Wrap Up (5 min)
- **Adding Conditionals**

View on Code Studio

Objectives

Students will be able to:

- Use conditionals to react to keyboard input
- Move sprites in response to keyboard input

Links

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

For the Teachers

- **Keyboard Input** - Slides

Introduced Code

- `keyDown(code)`

Teaching Guide

Warm Up (5 min)

Taking Input

Discuss: So far all of the programs you've written run without any input from the user. How might adding user interaction make your programs more useful, effective, or entertaining? How might a user provide input into your program?

Question of the Day: How can our programs react to user input?

Discussion Goal

The goal here isn't to get into the technical specifics of how programs can take input (students will get to that in the online portion of the lesson), but rather to get students thinking about how user input could change the programs they've made. Encourage students to think back to Unit 1 and the various computer inputs and outputs they explored then. Which inputs would be most useful for the types of programs they've been making?

Activity (40 min)

Keyboard Input

Transition: Send students to Code Studio

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Prediction 2 ***(click tabs to see student view)***

Skill Building 3 4 5 ***(click tabs to see student view)***

Keyboard Input Practice 6 6a 6b 6c

Assessment ✓ 7 ***(click tabs to see student view)***

Keyboard Input Challenges 8 8a 8b 8c

Wrap Up (5 min)

Adding Conditionals

Question of the Day: How can our programs react to user input?

Journal: Think back to all of the programs you've written so far; how might you use user interaction to improve one of your programs from past lessons? What condition would you check, and how would you respond to it?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

Lesson 16: Mouse Input

Overview

Question of the Day: What are more ways that the computer can react to user input?

In this lesson students continue to explore ways to use conditional statements to take user input. In addition to the keyboard commands learned yesterday, students will learn about several ways to take mouse input. They will also expand their understanding of conditionals to include `else`, which allows for the computer to run a certain section of code when a condition is true, and a different section of code when it is not.

Purpose

Students have learned how to make simple decisions with conditionals. Sometimes, however, we want to make decisions based on whether the condition we asked about originally was false. That's where else statements come in. Else statements are a second statement which is attached to an if statement. Else statements execute when the conditions they are attached to are false.

This concept is introduced alongside several new mouse input commands, allowing students to gradually build up programs that use input in different ways.

Assessment Opportunities

1. **Use an if-else statement to control the flow of a program.**

   See Level 9 in Code Studio.

2. **Respond to a variety of types of user input.**

   Use the wrap up discussion to check students' understanding of the different types of user input. You may also informally check their ability to use them as they progress through the Code Studio lesson.

Agenda

Warm Up (5 minutes)

- 3-2-1 Review

Activity (40 minutes)

- If/Else and More Input

Wrap Up (5 minutes)

- Wrap Up

View on Code Studio

Objectives

Students will be able to:

- Use an if-else statement to control the flow of a program.
- Respond to a variety of types of user input.

Links

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

For the Teachers

- **Mouse Input** - Slides

Vocabulary

- **Conditionals** - Statements that only run when certain conditions are true.

Introduced Code

- `keyWentDown(code)`
- `keyWentUp(code)`
- `mouseDidMove()`
- `mouseDown(button)`
- `mouseWentDown(button)`
- `mouseWentUp(button)`
- `sprite.visible`
- `If/else statement`

Teaching Guide

Warm Up (5 minutes)

3-2-1 Review

Prompt: What are three different things that you've been able to do with conditionals? What are two big things that everyone should remember when using conditionals? What's one thing you still want to learn how to program?

Share: Allow students to share out their responses.

Discussion Goal

Students may come up with many different specific scenarios, but make sure that they are differentiating between conditionals that respond to user input and those that respond to other changes in the program, such as a score increase, the location of a sprite, or player lives lost. Major things to remember may include putting the conditionals in the draw loop or paying attention to how the blocks are nested.

Remarks

> That's great! Today we're going to look at a way to make our conditionals even more powerful, and see some new ways to get user input.

Question of the Day: What are more ways that the computer can react to user input?

Activity (40 minutes)

If/Else and More Input

Transition: Move the class to Code Studio, and have students complete the prediction level as a class or in small groups, then talk about what they found.

Video: Watch the video as a class and review the discussion questions together.

Code Studio levels

Lesson Overview 1 *(click tabs to see student view)*

Prediction 2 *(click tabs to see student view)*

Video: If/Else Statements 3 *(click tabs to see student view)*

Skill Building 4 5 6 7 *(click tabs to see student view)*

Mouse Input and If/else Practice 8 8a 8b

Assessment 9 *(click tabs to see student view)*

Mouse Challenges 10 10a 10b 10c 10d 10e

Wrap Up (5 minutes)

Wrap Up

Question of the Day: What are more ways that the computer can react to user input?

Prompt: You now have many different ways to detect user input. With a partner, choose three different user input commands and think of an example of when you might use them. Be ready to share with the class!

Assessment Opportunity

This discussion serves as a brief review and assessment of the new user input commands. As students share out, press them to explain why their choice is better than other, similar choices (mouseDown / mouseWentDown / mouseWentUp). If any commands are missing after all the groups have shared out, elicit the missing ones from the group before moving on.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

Lesson 17: Project - Interactive Card

Overview

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

In this culminating project for Chapter 1, students plan for and develop an interactive greeting card using all of the programming techniques they've learned to this point.

Purpose

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. Students should still be working with code that is easily readable and doesn't involve very many high level abstractions. 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.

Assessment Opportunities

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

Agenda

Warm Up (10 min)

Journal

Activity (2 days)

Demo Project Exemplars (Level 2)
Unplugged: Interactive Card Planning
Levels: Implementing Interactive Card (Level 3 - 7)
Peer Review
Iterate - Update Code
Reflect

Wrap Up (10 minutes)

Reflection

View on Code Studio

Objectives

Students will be able to:

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

Preparation

- ☐ Read the Forum
- ☐ Print out a copy of the project guide for each student

Links

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

For the Teachers

- **Interactive Card** - Exemplar
- **Interactive Card** - Slides

For the Students

- **Interactive Card** - Project Guide Make a Copy ▾
- **Interactive Card** - Rubric Make a Copy ▾
- **Interactive Card** - Peer Review Make a Copy ▾
- **Computer Science Practices** - Reflection Make a Copy ▾

Teaching Guide

Warm Up (10 min)

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 (2 days)

Demo Project Exemplars (Level 2)

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.

Code Studio levels

- Levels
- **2**

Student Instructions

View on Code Studio

Example Project

Run the program a few times and answer the following questions:

1) Which elements appear to use drawing commands?

2) Which elements appear to be sprites?

3) For each sprite, which properties are being updated?

4) Where do you see conditionals being used?

5) Are there elements that you don’t understand?

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. Review the different components of the rubric with them to make sure they understand the components of the project.

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

Unplugged: Interactive Card Planning

Goal: Students should 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.

Distribute: Hand out the project guide to students. This is the tool students will use to scope out their projects before creating them. Give students some time to brainstorm the type of card they want to create and who the recipient will be.

Steps

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.

Levels: Implementing Interactive Card (Level 3 - 7)

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.

Code Studio levels

- Levels
- **3**
- **4**
- **5**
- **6**
- **7**

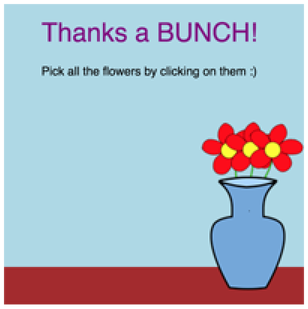
Student Instructions

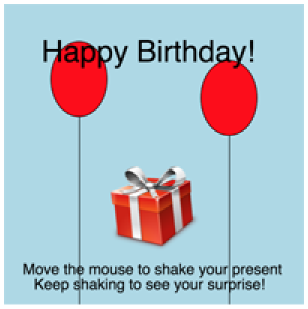
View on Code Studio

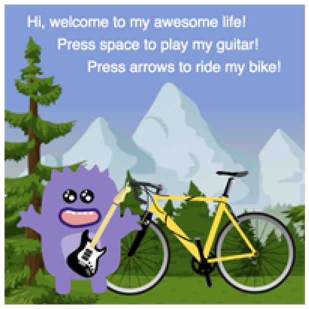
Your Interactive Card

In the next few levels, you'll be completing your own interactive card. Here are some examples to give you some ideas. Don't forget to look at the code to see how they work.

Examples




Happy Birthday!
Move the mouse to shake your present
Keep shaking to see your surprise!


Hi, welcome to my awesome life!
Press space to play my guitar!
Press arrows to ride my bike!

Student Instructions

View on Code Studio

Laying Out Your Background

Before beginning this project, you should have already completed the Interactive Card Planning activity, and you'll want to have that paper with you as you develop your program. Preparation is one of the **most important** elements of successfully creating a program!

Do This

Refer to your planning activity sheet to help you lay out the shapes that will become the background to your card.

- First, figure out what the lowest layer in your image is (this should use the `background()` block) and add it to the very top of the draw loop.
- Next, layer each additional drawing block in the order you want them to appear in the stack.
- Finally, add a comment to the top of this section of code to describe what it does, and if you have any particularly complicated chunks of code within (such as code to draw a tree or a house), add a descriptive comment to that as well.

Challenge: Can you use variables or `randomNumber()` to add some subtle animation to your background layer?

Student Instructions

View on Code Studio

Adding Sprites

Now that you have the more static elements of your card layed out, it's time to add the Sprites. Your Sprites should provide the primary animations and interactions for your card - so feel free to get creative here and have fun.

Do This

Check out the Sprites table on the back of your planning sheet. For each Sprite in your table:

- Initialize the Sprite at the top of your program with `createSprite()` .
- Find or create the image(s) for the Sprite and set it with `setAnimation()` .
- Inside the `draw()` loop update any Sprite properties that we will be constantly animating (we'll deal with conditionals in a minute).

Student Instructions

View on Code Studio

User Input

You've got a background, you've got Sprites, now it's time to give your user something to do!

Do This

On the interactions table from your planning sheet, find all of the interactions that rely on user input (key presses and mouse movements). For each of those interactions:

- Add an `if` block (or `if-else` block if you need a fallback action) inside the `draw()` loop.
- Add the appropriate input block for your condition (such as `keyDown()` or `mouseDown()`).
- Add the necessary actions inside the `if` block.

Challenge: Can you create more sophisticated conditionals by nesting them or using compound booleans?

Student Instructions

View on Code Studio

Other Conditionals

The *surprise* in your card comes from conditionals that don't directly respond to user input, but to some other element of your card. This could be triggered by a variable that gets updated as the user interacts with your card, or a Sprite moving into a certain part of the screen.

Do This

For each of the remaining items on your interactions table:

- Add an `if` block (or `if-else` block if you need a fallback action) inside the draw loop.
- Add the appropriate Boolean comparison block to the condition (eg. `<`, `>`, or `==`).
- Add the necessary actions inside the `if` block.

Challenge: Can you create more sophisticated conditionals by nesting them or using compound booleans?

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.

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.

Code Studio levels

- Levels
- **8**

Student Instructions

View on Code Studio

Finishing Touches

Now's your chance to put some finishing touches on your card. We've included some new blocks that you haven't seen before, so take some time to look around and try out some new blocks.

Do This

Consider adding any of the following to finish up your card:

- Text
- Additional images for your sprites
- Subtle animation in the background
- Sound effects (Can you figure out how to do this?)
- More ways for a user to interact with your card

Wrap Up (10 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?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

Lesson 18: Velocity

Overview

Question of the Day: How can programming languages hide complicated patterns so that it is easier to program?

After a brief review of how they used the counter pattern to move sprites in previous lessons, students are introduced to the idea of hiding those patterns in a single block. Students then head to Code Studio to try out new blocks that set a sprite's velocity directly, and look at various ways that they are able to code more complex behaviors in their sprites.

Purpose

This lesson launches a major theme of the chapter: that complex behavior can be represented in simpler ways to make it easier to write and reason about code.

In this lesson students are taught to use the velocity blocks to simplify the code for moving a sprite across the screen. This marks a shift in how new blocks are introduced. Whereas previously blocks were presented as enabling completely new behaviors, they are now presented as simplifying code students could have written with the blocks previously available. Over the next several lessons, students will see how this method of managing complexity allows them to produce more interesting sprite behaviors.

Assessment Opportunities

1. **Use the velocity and rotationSpeed blocks to create and change sprite movements**

   See Level 10 in Code Studio. (Level 6 can be used to check rotationSpeed.)

2. **Describe the advantages of simplifying code by using higher level blocks**

   In the wrap up, check students' descriptions of the blocks that they would create and why they would want to create them.

Agenda

Warm Up (5 min)
Activity (30 minutes)
Wrap Up (5 min)
Journal

View on Code Studio

Objectives

Students will be able to:

- Use the velocity and rotationSpeed blocks to create and change sprite movements
- Describe the advantages of simplifying code by using higher level blocks

Links

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

For the Teachers

- **Velocity** - Slides

Introduced Code

- `sprite.rotationSpeed`
- `sprite.velocityX`
- `sprite.velocityY`

Teaching Guide

Warm Up (5 min)

Demonstrate: Ask for a volunteer to come to the front of the class and act as your "sprite". Say that you will be giving directions to the sprite as though you're a Game Lab program.

When your student is ready, face them so that they have some space in front of them and ask them to "Move forward by 1". They should take one step forward. Then repeat the command several times, each time waiting for the student to move forward by 1 step. You should aim for the repetitiveness of these instructions to be clear. After your student has completed this activity, have them come back to where they started. This time repeat the demonstration but asking the student to "Move forward by 2" and have the student take 2 steps each time. Once the student has done this multiple times ask the class to give them a round of applause and invite them back to their seat.

Prompt: I was just giving instructions to my "sprite", but they seemed to get pretty repetitive. How could I have simplified my instructions?

Discuss: Give students a minute to write down thoughts before inviting them to share with a neighbor. Then have the class share their thoughts. You may wish to write their ideas on the board.

Discussion Goal

Goal: The earlier demonstration should have reinforced the fact that repeatedly giving the same instruction is something you would never do in real life. You would instead come up with a way to capture that the instruction should be repeated, like "keep moving forward by 1."

Remarks

> Programming languages also have ways to simplify things for us. Today, we're going to look at some blocks in Game Lab that hide complicated coding patterns to make things easier for programmers.

Question of the Day: How can programming languages hide complicated patterns so that it is easier to program?

Activity (30 minutes)

Remarks

> One way to simplify these instructions is to just tell our "sprite" to keep moving by 1 or 2, or however many steps we want. As humans, this would make instructions easier to understand, and as we're about to see there's a similar way to make our code simpler as well.

Transition: Move students to Code Studio

Circulate: These levels introduce the velocityX, velocityY, and rotationSpeed properties that you just discussed with students. Check in with students to see how they are doing and keep track of when everyone has made it to the end of level 10.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Skill Building 2 ***(click tabs to see student view)***

Video: Velocity 3 ***(click tabs to see student view)***

Skill Building 4 5 6 7 8 ***(click tabs to see student view)***

Velocity Practice

9 | 9a | 9b

Assessment

✓ 10 *(click tabs to see student view)*

Velocity Challenges

11 | 11a | 11b

Wrap Up (5 min)

Journal

Prompt: You learned a few new blocks today. At first glance, these blocks did the same sorts of things we'd already done with the counter pattern, but made it simpler for us to do them. As you went through the puzzles, though, you started doing some interesting movements that we hadn't been able to do before.

- Describe one of those movements, and how you made it.
- Describe another block that you'd like to have.
    - What would you name it?
    - What would it do?
    - What code would it hide inside?
    - How would it help you?

Assessment Opportunity

As students describe the blocks that they would make, ensure that they are relating the specific code that a block would use to the higher-level concept of what it would do. For example, a "velocity" block would move a sprite across the screen, and it would include blocks that use the counter pattern on a position property.

Students should describe the advantages of having these blocks, such as not needing to re-write the code all the time, or making it easier to read what the program is doing.

Remarks

> All of the movements that we did today are possible without the new blocks, but it would be very complicated to code them. One of the benefits of blocks like velocity is that when we don't have to worry about the details of simple movements and actions, we can use that extra brainpower to solve more complicated problems. As you build up your side scroller game, we'll keep looking at new blocks that make things simpler, so we can build more and more complicated games.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

Lesson 19: Collision Detection

Overview

Question of the Day: How can programming help make complicated problems more simple?

Students learn about collision detection on the computer. Working in pairs, they explore how a computer could use sprite location and size properties and math to detect whether two sprites are touching. They then use the `isTouching()` block to create different effects when sprites collide, and practice using the block to model various interactions.

Purpose

This lesson formally introduces the use of abstractions, simple ways of representing underlying complexity.

In the last lesson, students were exposed to the idea of using one block to represent complex code. Students further explore this idea in the context of the intentionally complex mathematical challenge of determining whether two sprites are touching. By using a single block to represent this complexity, in this case the isTouching block, it becomes much easier to write and reason about code, and students can appreciate the value of using abstractions. In later lessons, students will continue to build on the `isTouching()` abstraction to create more complex sprite interactions.

Assessment Opportunities

1. **Detect when sprites are touching or overlapping, and change the program in response.**

   See Level 7 in Code Studio.

2. **Describe how abstractions help to manage the complexity of code**

   In the Wrap Up discussion, make sure students can identify how more abstract blocks can help with creating larger or more complex programs.

Agenda

Warm Up (10 min)
Activity
- **Collisions Unplugged**
- **isTouching()**

Wrap Up (5 min)

View on Code Studio

Objectives

Students will be able to:

- Detect when sprites are touching or overlapping, and change the program in response.
- Describe how abstractions help to manage the complexity of code

Preparation

- [ ] Print copies of the activity guide such that each pair of students has a part A and a part B

Links

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

For the Teachers

- **Collision Detection** - Exemplar
- **Collision Detection** - Slides

For the Students

- **Collision Detection (Version B)** - Activity Guide Make a Copy
- **Collision Detection (Version A)** - Activity Guide Make a Copy

Vocabulary

- **Abstraction** - a simplified representation of something more complex. Abstractions allow you to hide details to help you manage complexity, focus on relevant concepts, and reason about problems at a higher level.

Introduced Code

- `sprite.isTouching(target)`
- `sprite.debug`

Teaching Guide

Warm Up (10 min)

Display: On a projector or a computer screen, demonstrate the game that appears in the first level in Code Studio for this lesson.

Prompt: An interesting aspect of this animation is that the sprites change when they touch each other. Can you think of any way that the computer could use the sprites' properties to figure out whether they are touching each other?

Discuss: Allow the students to brainstorm ideas for how the computer could determine whether the two sprites are touching. List their ideas on the board and tell them that they'll have a chance to try out their theories in a moment.

Teaching Tip

Showing the Game: Avoid signing students into Code Studio at this point to play the game themselves. They have a fairly significant unplugged activity immediately afterwards and it will likely lead to difficulties in transitioning.

Discussion Goal

Goal: The purpose of this discussion is just to get some ideas on the board for students to use in the next activity. There's no need to actually evaluate or try them, because students will be working together to do so immediately after the discussion.

Remarks

> This is a tough problem, and we're going to get to dig into it today. As we work on it, we're also going to look at ways that the computer can help us make these tough, complicated problems more simple.

Question of the Day: How can programming help make complicated problems more simple?

Activity

Collisions Unplugged

Group: Group students into pairs.

Remarks

> Now you're going to have a chance to try out the strategies that you came up with as a group. Each activity guide has four sheets of paper. One partner should take the papers with the "A" on the top, and the other should take the papers with the "B" on the top. You're each going to draw two secret sprites on the chart, and your partner will try to figure out whether or not they are touching, based on the same information that the computer will have about each sprite's properties. Don't let your partner see what you are drawing.

Distribute the activity guides to each set of partners. Ensure that one partner has taken Version A and the other has taken Version B.

Each student will have a line on which to draw two squares. The student chooses the location and the size of each of the squares, and then records the information about the squares in a table. They then switch tables (not drawings) and try to determine whether or not the two sprites are touching based on the width of each sprite and the distance between them.

The math to determine whether the sprites are touching is as follows:

1. Subtract the x (or y) positions of the sprites to find the distance between their centers.
2. Divide the width (or height) of each square by 2 to get the distance from the center to the edge.
3. If the distance between the centers of the sprites is greater than the sum of the distances from their centers to their edges, the sprites are not touching.
4. If the distance between the centers of the sprites is equal to the sum of the distances from their centers to their edges, the sprites are barely touching.
5. If the distance between the centers of the sprites is less than the sum of the distances from their centers to their edges, the sprites are overlapping.

Circulate: Support students as they complete the worksheet. If students are not sure how to determine whether the sprites are touching, encourage them to use one of the ideas on the board. Remind them that they are not being graded on whether they are right or wrong, but on their ability to use the problem-solving process. If any students are finished early, challenge them to find a method that will work for sprites anywhere on the grid, not just on the same line.

Share: After students have all had a chance to test their solutions, ask them to share what they discovered.

Remarks

> People can use a lot of different strategies to solve a problem like this. Because computers can't "see" the drawings in the same way that people can, they need to use math to figure out whether two things are touching. We looked at how this can work along a line, but we can combine these methods to work anywhere on the game screen.

isTouching()

Transition: Send students to Code Studio.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Sample Game 2 ***(click tabs to see student view)***

Skill Building 3 4 5 6 ***(click tabs to see student view)***

Collision Practice 7 7a 7b

Assessment ✓ 8 ***(click tabs to see student view)***

Collision Challenges 9 9a 9b 9c

Wrap Up (5 min)

Question of the Day: How can programming help make complicated problems more simple?

✓ **Prompt:** At the beginning of the lesson, you saw that it's possible to do everything that the `isTouching` block does without using the block at all. What makes this block useful?

✓ Assessment Opportunity

Students should note that even though they could program the code to detect whether sprites are touching each time, it is error-prone and would take up more time and energy than having a block that performs the same code automatically. The new block helps them to take on more difficult challenges by reducing the risk of errors and freeing them to think about the bigger picture.

Remarks

> One of the great things about programming is that once you've figured out a solution to a problem, you can often program it into the computer to be used over and over again. When you don't have to worry about the details of that particular problem, you can take on bigger challenges, as you did today. Using a simplified representation of something to hide the details so you can think about the big picture is called "abstraction", and it's a key tool programmers use to help them write complicated programs.

Key Vocabulary:

- **abstraction** - a simplified representation of something more complex.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

Lesson 20: Mini-Project - Side Scroller

Overview

Question of the Day: How can the new types of sprite movement and collision detection be used to create a game?

Students use what they have learned about collision detection and setting velocity to create a simple side scroller game. After looking at a sample side scroller game, students brainstorm what sort of side scroller they would like to make, then use a structured process to program the game in Code Studio.

Purpose

This lesson is a chance for students to get more creative with what they have learned. Encourage students to spend time on parts of the activity that interest them, as long as they meet the requirements of the assignment. This lesson can be shortened or lengthened depending on time constraints.

Agenda

View on Code Studio

Links

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

For the Teachers

- **Mini-Project Guide - Side Scroller** - Exemplar
- **Mini-Project: Side Scroller** - Slides

For the Students

- **Side Scroller** - Rubric Make a Copy
- **Side Scroller** - Project Guide Make a Copy

Teaching Guide

Warm Up

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

> Now that you've learned how to detect sprite interactions, you can start making some more interesting games. Today, we're going to look at how you can use what you've learned to make a side scroller game.

Question of the Day: How can the new types of sprite movement and collision detection be used to create a game?

Activity

Distribute: (Optional) pass out copies of the activity guide. Students can use this sheet to plan out the Side Scroller they create at the end of this lesson, but the planning can also be completed on scratch paper.

Transition Send students to Code Studio.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Intro to Side Scrollers 2 ***(click tabs to see student view)***

Draw Your Background 3 ***(click tabs to see student view)***

Create Your Sprites 4 ***(click tabs to see student view)***

Player Controls 5 ***(click tabs to see student view)***

Looping 6 ***(click tabs to see student view)***

Sprite Interactions 7 ***(click tabs to see student view)***

Scoring & Scoreboard 8 ***(click tabs to see student view)***

Review Your Game 9 ***(click tabs to see student view)***

Wrap up

Question of the Day: How can the new types of sprite movement and collision detection be used to create a game?

Prompt: What was one challenge in making this game? What is your advice for someone else who has the same challenge?

Lesson 21: Complex Sprite Movement

Overview

Question of the Day: How can previous blocks be combined in new patterns to make interesting movements?

Students learn to combine the velocity properties of sprites with the counter pattern to create more complex sprite movement. After reviewing the two concepts, they explore various scenarios in which velocity is used in the counter pattern, and observe the different types of movement that result. In particular, students learn how to simulate gravity. They then reflect on how they were able to get new behaviors by combining blocks and patterns that they already knew.

Purpose

This lesson does not introduce any new blocks and in fact only uses patterns students have seen in Chapter 1. Instead, it demonstrates how combining these tools, in particular the abstractions students learned in the previous two lessons, allows them to build new behaviors for their sprites. This highlights the broader point that abstractions not only simplify code, but also can themselves be used as building blocks of even more complex behavior.

Assessment Opportunities

1. **Use sprite velocity with the counter pattern to create different types of sprite movement**

   See Level 7 in Code Studio.

2. **Explain how individual programming constructs can be combined to create more complex behavior**

   In the wrap up discussions, check that students can explain how they are creating new types of movement without new blocks.

Agenda

Warm Up (10 mins)
Activity (30 mins)
Wrap Up (10 mins)

View on Code Studio

Objectives

Students will be able to:

- Use sprite velocity with the counter pattern to create different types of sprite movement
- Explain how individual programming constructs can be combined to create more complex behavior

Preparation

Links

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

For the Teachers

- **Complex Sprite Movement** - Slides

Teaching Guide

Warm Up (10 mins)

Display: Show the two images of a frog jumping to the class.

Prompt: Here are two images of a frog jumping. The first is from the side scroller. Do you have any ideas for how to make the second type of jumping?

Think-Pair-Share Allow students to discuss their ideas with a classmate before sharing with the entire class.

Discussion Goal

It is unlikely that students will come up with the solution on their own, but encourage them to think of as many ideas as possible, and that they will keep working on the problem throughout the class.

Remarks

> We've learned a lot of new blocks that have helped us create some fun animations. Today, we're going to look at how we can use the blocks that we already know in new ways to make more interesting types of movements. By the end of the class, you'll be able to code the new type of jumping with blocks that you already know.

Question of the Day: How can previous blocks be combined in new patterns to make interesting movements?

Activity (30 mins)

Transition: Move students to Code Studio.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Prediction 2 ***(click tabs to see student view)***

Skill Building 3 4 5 ***(click tabs to see student view)***

Sprite Movement Practice 6 6a 6b

Assessment 7 ***(click tabs to see student view)***

Collision Challenges 8 8a 8b 8c

Wrap Up (10 mins)

Share: Have students share with their classmates what additions they made to their final flyer game. Have students focus not just on how the game works, but on what the code to create that kind of functionality looks like.

Prompt: On your paper make two lists. First, make a list of new things you can program sprites to do after today's lesson. On the second list write down all the new blocks you learned today.

Discuss: Have students share their lists with classmates. Afterwards share lists as a class. They should hopefully have listed many new sprite movements but students haven't actually learned any new blocks in this lesson.

Prompt: Today we built lots of new sprite movements like gravity and jumping, but none of this required us to learn new blocks. How were you able to do new things without learning any new blocks?

Discuss: Lead a quick follow-up to your initial discussion about this point.

Discussion Goal

Goal: This conversation should highlight that students did not learn any new blocks in today's lesson, they just learned new ways to combine blocks and patterns they had learned previously. The broader point here is that programming is not always about learning new blocks, but being creative about combining the tools you already know how to use.

Assessment Opportunity

Check that students can explain the new programming structures and algorithms that they were able to use to get the new behaviors in the program.

Remarks

> We're going to keep learning a few more tools in Game Lab, but as we do, remember what we saw today. To create new kinds of programs you don't always need to learn new blocks. Most of the time the creativity of programming comes from learning to combine things you already know in new and creative ways.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

Lesson 22: Collisions

Overview

Question of the Day: How can programmers build on abstractions to create further abstractions?

In this lessson, students program their sprites to interact in new ways. After a brief review of how they used the `isTouching` block, students brainstorm other ways that two sprites could interact. They then use `isTouching` to make one sprite push another across the screen before practicing with the four collision blocks (`collide` , `displace` , `bounce` , and `bounceOff`).

Purpose

This lesson introduces collisions, another useful abstraction that will allow students to manipulate their sprites in entirely new ways. While students could theoretically have written their own displace, collide, or bounce commands, the ability to ignore the details of this code allows them to focus their attention on the high level structure of the games they want to build.

This lesson is also intended to give students more practice using the new commands they have learned in the second chapter. This the last time they will learn a new sprite behavior, and following this lesson students will transition to focusing on how they organize their increasingly complex code.

Assessment Opportunities

1. **Model different types of interactions between sprites.**

   See Level 8 in Code Studio.

2. **Describe how abstractions can be built upon to develop even further abstractions**

   In the wrap up, make sure students understand that blocks such as `isTouching` hide the complexity or details of the code inside, allowing them to tackle more complex problems.

Agenda

Warm Up
Activity
Wrap Up (10 mins)

View on Code Studio

Objectives

Students will be able to:

- Model different types of interactions between sprites.
- Describe how abstractions can be built upon to develop even further abstractions

Links

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

For the Teachers

- **Collisions** - Slides

Vocabulary

- **Abstraction** - a simplified representation of something more complex. Abstractions allow you to hide details to help you manage complexity, focus on relevant concepts, and reason about problems at a higher level.

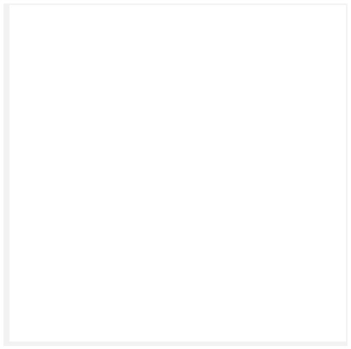
Introduced Code

- `sprite.bounce(target)`
- `sprite.bounceOff(target)`
- `sprite.collide(target)`
- `sprite.displace(target)`
- `setCollider(type, xOffset, yOffset, width/radius, height, rotationOffset)`
- `sprite.bounciness`

Teaching Guide

Warm Up

Display: Display the animated image. It is also available as a level in code studio.



Discussion Goal

Goal: The goal of this discussion is for students to brainstorm ways to solve the problem of having one sprite push another across the screen. There's no need for students to come to a consensus, because they will each have a chance to try out a solution in the next level in Code Studio. Students should understand that it is possible to use blocks to produce the desired movement just with the blocks that they have already learned.

Prompt: Using the blocks we already know how to use, how could we create the sprite interaction we can see in this program?

Share: Allow students to share out their ideas.

Remarks

> The first part of the problem is figuring out when the two sprites are touching, but we already figured out how to do that and can now use the `isTouching` block. That means we don't need to think about those details anymore. Using abstraction to hide the complicated details in that part of the problem means we can focus on the new part.

Teaching Tip

Students have seen this vocabulary before, but given its importance to the chapter, it is introduced again here.

Vocabulary Review:

- **abstraction** - a simplified representation of something more complex

Question of the Day: How can programmers build on abstractions to create further abstractions?

Activity

Prompt: This was a challenging problem, but we were able to solve it. What helped us to solve this problem?

Remarks

> All of these things are very important, and they come up in Computer Science a lot. One thing that was particularly helpful was the `isTouching` block, which hid the complicated code that tells us whether the two sprites are touching. There's also a `displace` block that hides the code we just wrote, and some other blocks that hide the code for some other types of sprite interactions. You'll have a chance to try out these blocks in the next few levels.

Assessment Opportunity

Make sure students talk about the importance of higher-level blocks, such as isTouching, and that while these blocks don't provide new functionality, hiding the complexity of the code inside of a single block allows them to tackle more complex problems.

This is also a good time to call out how far the students have progressed in their skills since the beginning of the unit. This problem would have seemed almost impossible at the beginning of the year. Some things that made the problem easier to solve were:

- Preparation: The students brainstormed and thought about solutions before trying out their code.
- Cooperation: Students worked as a group to come up with a solution
- Abstraction: Students were able to use the `isTouching` and `velocityY` blocks to hide part of the solution's complexity.

Code Studio levels

Lesson Overview
1 ***(click tabs to see student view)***

Code Prediction
2 ***(click tabs to see student view)***

Skill Building
3 | 4 | 5 | 6 ***(click tabs to see student view)***

Collisions Practice
7 | 7a | 7b | 7c | 7d

Assessment
8 ***(click tabs to see student view)***

Collision Challenges
9 | 9a | 9b | 9c | 9d

Wrap Up (10 mins)

Question of the Day: How can programmers build on abstractions to create further abstractions?

Vocabulary Review:

- **abstraction** - a simplified representation of something more complex

Prompt: How did having the `isTouching` block and the `velocityX` block make it easier to solve the problem of one sprite pushing another?

Assessment Opportunity

Students should understand that these two blocks represent partial solutions to the problem of one sprite pushing the other, and that by hiding the details of those partial solutions, they can more easily focus on how to fit those partial solutions together to solve larger and more complex problems.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

Lesson 23: Mini-Project - Flyer Game

Overview

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

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.

Purpose

This lesson is a chance for students to get more creative with what they have learned. Encourage students to spend time on parts of the activity that interest them, as long as they meet the requirements of the assignment.

Agenda

Warm Up
Activity
Wrap up
  Share Out and Journal 3-2-1

View on Code Studio

Links

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

For the Teachers

- **Mini-Project: Flyer Game** - Slides

For the Students

- **Flyer Game** - Rubric Make a Copy
- **Flyer Game** - Project Guide Make a Copy

Teaching Guide

Warm Up

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?

Activity

Distribute: (Optional) pass out copies of the project guide. Students can use this sheet to plan out the Flyer Game they create at the end of this lesson, but the planning can also be completed on scratch paper.

Transition Send students to Code Studio.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Intro to Flyer Game 2 ***(click tabs to see student view)***

Make Your Sprites 3 ***(click tabs to see student view)***

Player Controls 4 5 6 ***(click tabs to see student view)***

Sprite Movement 7 ***(click tabs to see student view)***

Sprite Interactions 8 ***(click tabs to see student view)***

Review Your Game 9 ***(click tabs to see student view)***

Wrap up

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?

Lesson 24: Functions

Overview

Question of the Day: How can programmers use functions to create their own abstractions?

Students learn how to create functions to organize their code, make it more readable, and remove repeated blocks of code. Students first think about what sorts of new blocks they would like in Game Lab, and what code those blocks would contain inside. Afterwards students learn to create functions in Game Lab. They will use functions to remove long blocks of code from their draw loop and to replace repeated pieces of code with a single function.

Purpose

In previous lessons students have learned to use a number of abstractions in their programs, including the velocity properties, isTouching, and collisions. These abstractions have allowed them to build much more complex programs while ignoring the details of how that behavior is created. In this lesson students learn to build abstractions of their own by creating functions.

Students will primarily use functions to break code into logical chunks that are easier to reason about. This foreshadows the chapter's transition from building technical skill to the organizational processes used to develop software.

Assessment Opportunities

1. **Create and use functions for blocks of code that perform a single high-level task within a program**

   See Level 10 in Code Studio.

2. **Explain the advantages of using functions in a program.**

   In the wrap up discussion, make sure students understand that functions can increase both the readability and organization of code.

3. **Explain how functions allow programmers to reason about a program at a higher level**

   In the wrap up, students should make the connection between functions and abstraction, that functions allow a programmer to name a bit of code so that they can think about it at that higher level, rather than worry about all the details.

Agenda

Warm Up (5 mins)

Your Personal Blocks

View on Code Studio

Objectives

Students will be able to:

- Create and use functions for blocks of code that perform a single high-level task within a program
- Explain the advantages of using functions in a program
- Explain how functions allow programmers to reason about a program at a higher level

Links

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

For the Teachers

- **Functions** - Slides

Vocabulary

- **Function** - A named bit of programming instructions.

Introduced Code

- `Define a function`
- `Call a function`

Activity (30 mins)
Wrap Up (10 mins)

Teaching Guide

Warm Up (5 mins)

Your Personal Blocks

Prompt: What's one block you'd like to have in Game Lab? What would it do? What code would it use to work?

Think-Pair-Share: Allow students to explain their blocks to a partner before sharing out their ideas.

Discussion Goal

As students share their ideas, make sure they have a clear idea of the code that they would use to make the block, reminding them that all of the blocks they have learned this chapter (e.g. velocity, collisions) have been built from blocks that they already knew.

Remarks

> Today, we're going to learn how to create our own blocks so that we decide exactly how they will work. These special blocks are called functions, and they are one of the most powerful parts of programming.

Question of the Day: How can programmers use functions to create their own abstractions?

Activity (30 mins)

Transition: Move students into Code Studio where they will learn to create and call functions.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Video: Functions 2 ***(click tabs to see student view)***

Skill Building 3 4 ***(click tabs to see student view)***

Predict 5 ***(click tabs to see student view)***

Functions Practice 6 6a 6b 6c 6d

Quick Check 7 ***(click tabs to see student view)***

Collector Game 8 9 10 ***(click tabs to see student view)***

Functions Challenges 11 11a 11b

Wrap Up (10 mins)

Key Vocabulary:

- **Function** - a named bit of programming instructions

Prompt: Why would we say that functions allow us to "create our own blocks?" Why is this something we'd want to do? Why would a function count as an abstraction?

Discuss: Have students discuss at their table before talking as a class.

Remarks

> Functions are a useful tool for helping us write and organize more complex pieces of code. As we start looking to the end of the unit and your final project, being able to keep your code organized will be an important skill.

Question of the Day: How can programmers use functions to create their own abstractions?

Assessment Opportunity

Goal: Use this first prompt to review what students learned today. When they create a function they are creating their own block that they can call or use whenever they like. They saw at least two primary motivations for creating functions today including:

- Simplifying code by breaking it into logically named chunks
- Allowing a programmer to think at a higher level by hiding the details or a particular bit of programming instructions
- Avoiding repeated code by making one block you can use multiple times

Students should review the definition of abstraction as "a simple way of representing something complex". Note that a function is an abstraction because it allows you to create one simple name for a more complex block of code.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

Lesson 25: The Game Design Process

Overview

Question of the Day: How does having a plan help to make a large project easier?

This lesson introduces students to the process they will use to design games for the remainder of the unit. This process is centered around a project guide which asks students to define their sprites, variables, and functions before they begin programming their game. In this lesson students begin by playing a game on Game Lab where the code is hidden. They discuss what they think the sprites, variables, and functions would need to be to make the game. They are then given a completed project guide which shows one way to implement the game. 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 game to make it their own.

Purpose

This lesson introduces multi-frame animations, and is the first in a sequence centered around the process of building software.

While previous lessons focused on using abstractions to manage the complexity of the code, this lesson focuses on managing the complexity of the software development process. 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**

   In the Wrap Up discussion, make sure students are clear on how the project guide helps them to break their programming work into manageable chunks, and that they understand how listing the sprites and functions ahead of time can help them focus on specific bits of code rather than the entire program at once.

Agenda

Warm Up (15 mins)

- **Play Cake Defender**
- **Stop: Review Project Guide**

Activity (60 mins)

View on Code Studio

Objectives

Students will be able to:

- Implement different features of a program by following a structured project guide

Preparation

- [ ] Print copies of the the project guide if you will be giving students physical copies. Please note that this project guide is intentionally filled out. (See notes in Lesson Plan.)

Links

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

For the Teachers

- **The Game Design Process** - Slides

For the Students

- **Defender Game** - Project Guide (Filled Out)
  Make a Copy ▾

Implement Project Guide

Wrap Up (5 mins)

Using the Project Guide

Teaching Guide

Warm Up (15 mins)

Play Cake Defender

Transition: This lesson begins immediately in Code Studio. On the first level, students will find a game but will not be able to see the code. They should play the game and follow the instructions which ask them to list the variables, sprites, and functions they think are necessary to create this game.

Code Studio levels

- Levels
- 2

Stop: Review Project Guide

Discuss: Students should have individually created a list of variables, sprites, and functions they would create to make the defender 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 "Variables", "Sprites", 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.

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, including the sprites, variables, and functions you will want in your program. Today we're going to look at how we could implement this plan to build our own defender 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!

Question of the Day: How does having a plan help to make a large project easier?

Activity (60 mins)

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

Prompt: Compare the components of the game you thought would be included to the ones on this project guide. Do you notice any differences?

Discuss: As a class compare the list you made on the board to the list of variables, sprites, 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

Project Guide: The project guide is intentionally filled out for students so that they can experience using it as a reference when programming. This should give them more context when filling out their own project guide in the next two lessons.

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.

Implement Project Guide

Students are given a large amount of starter code in this project. The sprites, variables, and functions have all already been given to them. The work of this project is writing the code for the individual functions. These levels guide students through how to implement those functions. As students move through the levels point out how the project guide is being used.

The most challenging skill students use in these levels is recognizing the need to create new functions to replace repeated code. Students need to build this skill on their own but these levels demonstrate an instance where this might happen.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Same Game ***(click tabs to see student view)***

Plan Your Project 3 ***(click tabs to see student view)***

Set Up Sprites 4 5 ***(click tabs to see student view)***

Control Your Player 6 7 8 ***(click tabs to see student view)***

Sprite Interactions 9 10 11 12 13

(click tabs to see student view)

Challenges 14 14a 14b 14c 14d

Wrap Up (5 mins)

Using the Project Guide

Prompt: Today, you used a filled-out project guide as you completed your program.

- How did the project guide help you as you coded?
- What do you think will be important to remember when you fill out your own project guide?

Assessment Opportunity

As students answer the questions, make sure that they understand that the project guide is there to help them organize their work, so that they can look at smaller parts of the problem rather than thinking about it all at one time. They can use it to focus on one part of their program at a time as they code.

In the next lesson, as they fill out their own project guides, they will need to think carefully about the different parts of their program before they start coding. You may want to remind then of the "Prepare" part of the Problem Solving Process, and to think about how they might need to go back and tweak the project guide even after they have started coding.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 26: Using the Game Design Process

Overview

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

In this multi-day lesson, students use the problem solving process from Unit 1 to create a platform jumper game. They start 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.

Purpose

Students have already learned all of the programming constructs that they need to make a game. This lesson reviews many of those concepts while introducing them to a structured process that will help them to manage the work. It 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. This activity prepares students to write their own game from scratch for the final project.

Assessment Opportunities

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

   In the project guide, 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.

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

   See Level 20 in Code Studio.

Agenda

Warm Up
Activity
- **Play Alien Jumper**
- **Discuss Project Guide**
- **Share out**

Wrap Up
- **Journal**

View on Code Studio

Objectives

Students will be able to:

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

Preparation

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

Links

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

For the Teachers

- **Planning Your Platform Game** - Exemplar
- **Using the Game Design Process** - Slides

For the Students

- **Planning Your Platform Game** - Project Guide Make a Copy

Teaching Guide

Warm Up

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. Afterwards, 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, 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, get feedback from other people to make the game better.

Discussion Goal

Goal: Students should share their thoughts but if it doesn't come up naturally then suggest the examples provided. 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

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.

Code Studio levels

- Levels
- **2**

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.

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.

Code Studio levels

Lesson Overview 1 ***(click tabs to see student view)***

Sample Platform Jumper Game ***(click tabs to see student view)***

Build a Platform Jumper 3 ***(click tabs to see student view)***

Platform Jumper - Background and Variables 4 5 6 7 ***(click tabs to see student view)***

Platform Jumper - Platforms 8 9 10 ***(click tabs to see student view)***

Platform Jumper - Items 11 12 13 ***(click tabs to see student view)***

Platform Jumper - Player 14 15 16 17 ***(click tabs to see student view)***

Platform Jumper Review ✓ 18 ***(click tabs to see student view)***

Challenges 19 20 ***(click tabs to see student view)***

Challenges 21 21a 21b 21c 21d 21e

Share out

Share: Students share their games with their classmates.

Wrap Up

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?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

Lesson 27: Project - Design a Game

Overview

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

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.

Purpose

This lesson is the culmination of Unit 3 and provides students an opportunity to build a Game Lab project of their own from the ground up. The scaffolding provided by the project guide and the practice they have using it are intended to assist students in scoping their projects and seeing their ideas through to completion. This project is an opportunity to showcase technical skills, but they will also need to collaborate with their partner, provide constructive peer feedback, and repeatedly use the problem solving process as they encounter obstacles along the way. This project should be student-directed whenever possible, and provide an empowering and memorable conclusion to this unit of CS Discoveries.

Assessment Opportunities

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.

Agenda

Warm Up (5 mins)
Activity (100-200 mins)
- **Review Project Guide**
- **Define - Scope Game**
- **Prepare - Complete Project Guide**
- **Try - Write Code**

View on Code Studio

Objectives

Students will be able to:

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

Links

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

For the Teachers

- **Make Your Own Game** - Exemplar
- **Project - Design a Game** - Slides

For the Students

- **Make Your Own Game** - Project Guide
  Make a Copy ▾
- **Make Your Own Game** - Rubric
  Make a Copy ▾
- **Make Your Own Game** - Peer Review
  Make a Copy ▾
- **Computer Science Practices** - Reflection
  Make a Copy ▾

- Reflect - Peer Review
- Iterate - Update Code
- Share

Wrap Up (10 mins)

- Journal
- Reflect

Post-Project Test

Teaching Guide

Warm Up (5 mins)

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 (100-200 mins)

Review Project Guide

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

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. Direct students towards the rubric so that they know from the beginning what components of the project you will be looking for.

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.

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.

Try - Write Code

Transition: Students are now ready to program their games on Code Studio. These 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.

Code Studio levels

- Levels
- **1**
- **2**
- **3**

- 4
- 5
- 6
- 7

Student Instructions

View on Code Studio

Student Instructions

View on Code Studio

Teaching Tip

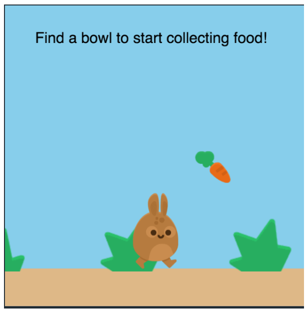
View on Code Studio to access answer key(s)

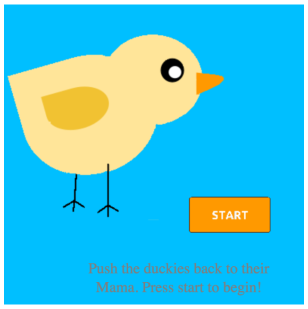
Create your own game

Now that you have all the skills you need, it's time to make your own game!

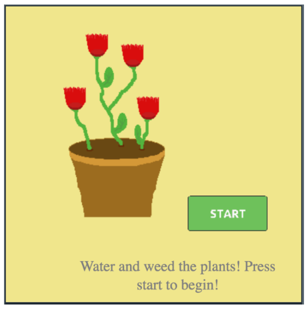
With a partner, brainstorm some different ideas for your game. You can think about the games you've already seen, or look at some more sample games to give you ideas.

Space to Jump, Arrows to Move, Get the Apples!


Find a bowl to start collecting food!



Push the duckies back to their Mama. Press start to begin!



Water and weed the plants! Press start to begin!

Once you have settled on a type of game with your partner, fill out the Project Guide with the backgrounds, variables, sprites, and functions that you will need to make the game. You'll spend the next few levels creating your game.

Student Instructions

View on Code Studio

Create your Variables

First, you'll need to create all of your variables and put them in the variables area of your code. **Show me the block** **Show me the area in the code**

Don't forget, each variable needs a label (name) and a starting value. You can change the value of the variable later in your code.

Student Instructions

View on Code Studio

Create your Backgrounds

Next, you'll create all of the background functions that you need for your game. Some games only have one background, and others have more than one that's chosen according to user score or another aspect of gameplay. You'll need to create a function for each separate background in your game. You'll write the code to choose the correct background in the next level.

- **Show me the block to create a new function**
- **Show me the area in the code to put my function**

After you create your functions, test them by calling them inside the draw loop, one background per test.

- **Show me the block to call my function**

Student Instructions

View on Code Studio

Display Boards

Now that your backgrounds are working, you can add your display boards. Most games have a score board, but you might also want to display information about player level or lives remaining. Look at **Lesson 17 Level 7, Sublevel 2** for an example of how to make a scoreboard.

For each display board: *Create a function to display the information* Call the function in the draw loop

Be sure to test your boards by changing the starting value of your variables and making sure the board also changes when you run the code.

Student Instructions

View on Code Studio

Choose your Backgrounds

Now that you have the backgrounds that you need, you'll write the code to choose the correct background. You've seen this done in **Lesson 24 Level 10**.

After you've written the code, test it by changing the starting value of your variables and making sure the correct background shows up.

Student Instructions

View on Code Studio

Create your Animations

Next you will create your animations in the animation tab. Don't forget to make multiple animations if you want your sprite to change appearance according to how it's moving.

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.

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.

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 (10 mins)

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 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.

Code Studio levels

- Levels
- **13**

Student Instructions

View on Code Studio

This level is an assessment or survey with multiple questions. To view this level click the "View on Code Studio" link.

Post-Project Test

The post-project test is found at the bottom of the Interactive Animations and Games unit overview page on Code Studio (**studio.code.org/s/csd3-2020**).

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.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming