Unit 3 Coding with Karel the Dog

Content Area: **Computer Science**
Course(s):
Time Period: **Marking Period 1**
Length: **10 days**
Status: **Published**

Summary

In this unit, students learn the foundations of programming using CodeHS and Karel the Dog, a beginner-friendly JavaScript environment. Students start by giving Karel simple commands, then learn how Karel's world is organized (streets, avenues, and directions). They progress to writing their own functions to teach Karel new commands, organizing their code with the start function to make it readable, and finally using top-down design and decomposition to break complex problems into smaller, reusable pieces. Throughout, students write, run, and debug real JavaScript programs, building the problem-solving habits and programming vocabulary that carry into later coding work.

Revision Date: July 2026

Essential Questions

- How do we give a computer precise instructions to accomplish a task?
- Why do we write functions, and how do they make programs easier to read and reuse?
- How does breaking a problem into smaller parts (decomposition) help us solve it?
- What makes code readable, and why does readability matter?
- How do we find and fix errors (debug) in our programs?

Enduring Understandings

- Computer programming is a way to create solutions to problems.
- Programs are made of precise instructions, and programming languages have specific syntax that must be followed.
- Functions let us teach the computer new commands and avoid repeating code.
- Top-down design and decomposition break a complex program into smaller, readable, reusable pieces.

- There is not always one ‘right answer,’ and failure (bugs) is an opportunity for learning.

Students Will Know

- The basic Karel commands (move, turnLeft, putBall, takeBall) and how Karel’s world is organized into streets, avenues, and directions.
- The difference between defining a function and calling a function.
- That a program runs by calling the start function, and why the start function improves readability.
- What top-down design and decomposition are, and how to recognize good vs. poor decomposition.
- Correct syntax rules (capitalization, parentheses, semicolons) and common sources of bugs.

Students Will Be Skilled At

- Writing, running, and reading Karel programs in JavaScript on CodeHS.
- Defining their own functions and calling them to build higher-order programs that go beyond the basic command set.
- Using the start function and meaningful function names to make programs readable.
- Decomposing a complex problem into smaller functions using top-down design.
- Debugging programs that use commands or functions incorrectly.

Learning Plan

- Introduce CodeHS navigation and the idea of programming/algorithms.
- Work through Part A (Karel Basics): commands, Karel’s world, and defining a first function (turnRight).
- Work through Part B (Functions): defining vs. calling, custom-function practice, the start function for readability, and top-down design/decomposition.

- Watch lesson videos, take quizzes, and complete CodeHS exercises with teacher circulating for support.
- Use reflections (Teaching Karel New Commands; Top-Down Design) as exit tickets or written responses.
- Encourage students to help one another and use CodeHS references (Cheat Sheet, textbook, problem guides).

Assessment

Assessments

- Formative: CodeHS exercises, lesson quizzes (Karel Commands, More Basic Karel, Functions, Start Function, Top-Down Design), and written reflections.
- Summative: Programming exercises that require student-defined functions and decomposition (e.g., Fireman Karel, Hurdle Karel, The Two Towers, Make a 'Z'), scored with the CodeHS problem guides/rubrics.
- Benchmark: CodeHS check-for-understanding assessments on programming terms (command, function, define/call, start function, decomposition, debugging).
- Alternative Assessment: Original Sandbox program that defines and calls multiple functions; peer code review using the Peer Code Review template.

Lessons at a Glance

The unit follows the CodeHS "Introduction to Programming with Karel" sequence, organized into two parts.

Lesson	Focus	Key Activities (CodeHS)
Part A – Karel Basics (CodeHS Unit 1)		
1.1 Introduction to Programming with Karel	Commands & first programs	Intro video & quiz; Our First Karel Program; Your First Karel Program; Short Stack; Dancing Karel
1.2 More Basic Karel	Karel's world: streets, avenues, directions	More Basic Karel video & quiz; Tennis Ball Square; Make a Tower; Pyramid of Karel; Go Through the Door
1.3 Karel Can't Turn Right	Defining functions (e.g., turnRight)	Video & quiz; Tower and Turn Right; Slide Karel; Fireman Karel; Reflection
Part B – Functions (CodeHS Unit 2)		
2.1 Functions in Karel	Defining vs. calling functions	Dance activity; video & quiz; Turn Around; Pancakes; Backflip
2.2 Functions Practice	Custom functions for higher-order programs; debugging	Digging Karel; Build a Shelter; Build a Tent
2.3 The Start Function	Readable code; the start function	The Start Function & quiz; Tower with Start; Pancakes with Start; Digging Karel with Start

2.4 Top-Down Design & Decomposition	Breaking problems into smaller parts	Top-Down Design & quiz; Hurdle Karel; The Two Towers; Make a 'Z'; Reflection

Part A – Karel Basics

Lesson 1.1 – Introduction to Programming with Karel

Description: Students are introduced to CodeHS and how Karel the Dog can be given a set of instructions to perform a simple task.

Objective: Introduce students to Karel and the commands she can be given.

Key Activities

- Intro video "Introduction to Programming with Karel" and Quiz: Karel Commands.
- Our First Karel Program (example); Your First Karel Program; Short Stack; Dancing Karel.

Lesson 1.2 – More Basic Karel

Description: Students learn about Karel's 'world' and the ways Karel can interact with it.

Objective: Introduce Karel's world (streets, avenues, directions) and more commands.

Key Activities

- More Basic Karel video and quiz.
- Tennis Ball Square; Make a Tower; Pyramid of Karel; Go Through the Door.

Lesson 1.3 – Karel Can't Turn Right

Description: Karel can learn new commands through functions. Defining a function has syntax rules.

Objective: Use Karel and commands to introduce students to functions.

Key Activities

- Karel Can't Turn Right video and quiz.
- Tower and Turn Right; Slide Karel; Fireman Karel; Reflection: Teaching Karel New Commands.

Part B – Functions

Lesson 2.1 – Functions in Karel

Description: Functions teach Karel a new command and let us break a program into smaller, easier-to-understand pieces.

Objective: Help students understand what functions are for and how they improve programs.

Key Activities

- Unplugged “dance” activity to model breaking steps into reusable parts.
- Functions in Karel video and quiz; Turn Around; Pancakes; Backflip.

Lesson 2.2 – Functions Practice

Description: Extra practice creating custom functions.

Objective: Create custom functions; use functions to build higher-order programs beyond the basic Karel toolbox; debug programs that use functions incorrectly.

Key Activities

- Digging Karel; Build a Shelter; Build a Tent.

Lesson 2.3 – The Start Function

Description: All programs start by “calling” the start function.

Objective: Deepen understanding of functions; explain the importance of readable code; compare the readability of programs; use the start function to make programs more readable.

Key Activities

- The Start Function (example) and quiz.
- Tower with Start Function; Pancakes with Start; Digging Karel with Start.

Lesson 2.4 – Top-Down Design & Decomposition

Description: Top-down design and decomposition break a program into smaller functions to avoid repeated code and improve readability.

Objective: Break a large problem into smaller pieces; write methods to solve each piece; solve a complex problem using top-down design; identify good and poor decomposition.

Key Activities

- Top-Down Design and Decomposition in Karel and quiz.
- Hurdle Karel; The Two Towers; Make a ‘Z’; Reflection: Top-Down Design.

Key Vocabulary

Karel: A dog who listens to your commands.

Command: An instruction you can give to Karel (e.g., move, turnLeft, putBall, takeBall).

Define a function: Teaching the computer a new command and explaining what it should do when it receives that command.

Call a function: Giving the command so the computer runs the code for that function.

Start function: The function that every program calls first to begin running.

Top-down design: Planning a program by starting with the big task and breaking it into smaller sub-tasks.

Decomposition: Breaking a problem into smaller, manageable parts, each solved by its own function.

Street / Avenue: The rows (streets) and columns (avenues) that organize Karel's world.

Pseudocode: A plain-language outline of a program's steps before writing actual code.

Debug: Finding and fixing errors in a program.

Teaching & Learning Strategies

- Open the unit with discussion: "What is a computer? What is computing?" List the computers in the room and debate whether a dog could be a computer.
- Have students set up CodeHS accounts; keep a printed copy of student logins handy.
- Use the unplugged "program-a-dancer" activity to introduce functions and decomposition.
- Write pseudocode for exercises before coding; encourage experimenting with order, capitalization, parentheses, and semicolons to see how syntax errors occur.
- Use the Peer Code Review template and Class Discussion Notes template for collaboration and reflection.
- Pair programming: when one partner dominates, ask the other to confirm the logic and add their thinking.

Standards

English Language Development Standard 4: Language for Science: English language learners communicate information, ideas and concepts necessary for academic success in the content area of science.

L.SS.6.1	Demonstrate command of the system and structure of the English language when writing or speaking. Troubleshooting a problem is more effective when knowledge of the specific device along with a systematic process is used to identify the source of a problem.

8.1.8.CS.4	Systematically apply troubleshooting strategies to identify and resolve hardware and software problems in computing systems.
L.VL.6.3.E	Verify the preliminary determination of the meaning of a word or phrase (e.g., by checking the inferred meaning in context or in a dictionary).
	Individuals design algorithms that are reusable in many situations. Algorithms that are readable are easier to follow, test, and debug.
8.1.8.AP.1	Design and illustrate algorithms that solve complex problems using flowcharts and/or pseudocode.
	Programmers create variables to store data values of different types and perform appropriate operations on their values.
8.1.8.AP.2	Create clearly named variables that represent different data types and perform operations on their values.
	Control structures are selected and combined in programs to solve more complex problems.
8.1.8.AP.3	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
	Programs use procedures to organize code and hide implementation details. Procedures can be repurposed in new programs. Defining parameters for procedures can generalize behavior and increase reusability.
8.1.8.AP.4	Decompose problems and sub-problems into parts to facilitate the design, implementation, and review of programs.
8.1.8.AP.5	Create procedures with parameters to organize code and make it easier to reuse.
	Individuals design and test solutions to identify problems taking into consideration the diverse needs of the users and the community.
8.1.8.AP.6	Refine a solution that meets users' needs by incorporating feedback from team members and users.
8.1.8.AP.7	Design programs, incorporating existing code, media, and libraries, and give attribution.
8.1.8.AP.8	Systematically test and refine programs using a range of test cases and users.
8.1.8.AP.9	Document programs in order to make them easier to follow, test, and debug.

Materials

Core instructional materials:

- CodeHS.com – Introduction to Programming with Karel (videos, exercises, quizzes, textbook, problem guides, solution references).
- CodeHS handouts: Karel Commands, Meet Karel the Dog, Naming Functions, Dancing with Functions, What's Wrong With This Function? (student and teacher versions).
- Peer Code Review template; Class Discussion Notes template.
- Student devices with internet access; speakers for the dance/functions activity.

Supplemental materials: Khan Academy; code.org.

Suggested Strategies for Modification

Suggested Strategies for Modification for Computer Science