CodeHS

Video Game Design Syllabus

1 year for High School (175 contact hours)

Course Overview and Goals

The CodeHS video game design curriculum teaches the foundations of creating video games in JavaScript. While this course is introductory, it is an honors-level course. Its curriculum teaches the foundations of computer science and basic programming, with an emphasis on helping students develop logical thinking and problem solving skills. Once students complete the course, they will have learned material equivalent to a semester college introductory course in Computer Science and be able to program in JavaScript

Learning Environment: The course utilizes a blended classroom approach. The content is fully web-based, with students writing and running code in the browser. Teachers utilize tools and resources provided by CodeHS to leverage time in the classroom and give focused 1-on-1 attention to students. Each module of the course is broken down into lessons. Lessons consist of video tutorials, short quizzes, example programs to explore, and written programming exercises, adding up to over 100 hours of hands-on programming practice in total. Each module ends with a comprehensive module test that assesses students' mastery of the material from that module.

Programming Environment: Students write and run JavaScript programs in the browser using the CodeHS editor.

More information: Browse the content of this course at https://codehs.com/course/20517

Prerequisites

The Video Game Design course is designed for complete beginners with no previous background in computer science, but does teach advanced topics. The course is highly visual, dynamic, and interactive, making it engaging for new coders.

Course Breakdown

Module 1: Introduction to Programming in JavaScript with Karel the Dog (5 weeks/25 hours)

Students learn the basics of programming by giving Karel the Dog commands in a grid world.

Objectives / Topics Covered	• Commands • Defining vs. Calling Methods • Designing methods • Program entry points • Control flow • Looping • Conditionals • Commenting code

	● Preconditions and Postconditions ● Top Down Design
Assignments / Labs	● 30 Karel Programming Exercises and Challenges in total ● Program-specific tasks for Karel the Dog ○ Example Exercise: Pyramid of Karel Write a program to have Karel build a pyramid. There should be three balls on the first row, two in the second row, and one in the third row. ● Teach Karel new commands like `turnRight()` or `makePancakes()` ○ Example Exercise: Pancakes Karel is the waiter. He needs to deliver a stack of pancakes to the guests on the 2nd, 4th, and 6th avenue. Each stack of pancakes should have three pancakes. Create a method called `makePancakes()` to help Karel solve this problem. ● Solve large Karel problems by breaking them down into smaller, more manageable problems using Top Down Design ○ Example Exercise: The Two Towers In this program, Karel should build two towers of tennis balls. Each tower should be 3 tennis balls high. At the end, Karel should end up on top of the second tower, facing East. ● Using control structures and conditionals to solve general problems ○ Example Exercise: Random Hurdles Write a program that has Karel run to the other side of first street, jumping over all of the hurdles. However, the hurdles can be in random locations. The world is fourteen avenues long. ○ Example Exercise: Super Cleanup Karel Karel's world is a complete mess. There are tennis balls all over the place, and you need to clean them up. Karel will start in the bottom left corner of the world facing east, and should clean up all of the tennis balls in the world. This program should be general enough to work on any size world with tennis balls in any locations.

Unit 2: JavaScript Basics (1 week/5 hours)

Students learn the basics of JavaScript including variables, user input, mathematics, and functions.

Objectives / Topics Covered	● Variables ● User Input ● Arithmetic Expressions ● Constants ● Collaborative Programming ● Random Numbers ● Functions
Assignments / Labs	● 12 JavaScript programming exercises in total ● Using variables and getting user input using JavaScript ○ Example Exercise: Dinner Plans Prompt the user for their name, then ask them what time you should meet for dinner. Greet them by name and tell them you will meet them at the time they specified!

Unit 3: The Canvas and Graphics (1 week/3-5 hours)

Students learn how to add graphics objects and position them on the canvas.

Objectives / Topics Covered	● JavaScript Canvas ● JavaScript Graphics ● Positioning Graphics Objects
Assignments / Labs	● 7 JavaScript programming and graphics exercises in total ○ Example Exercise: Create Your Own Meme In this exercise, you are going to create your own meme! The only requirements are that you add at least one image and one text element.

Unit 4: Graphics Challenges (1 week/3-5 hours)

Students apply what they have learned about graphics and basic JavaScript to complete a set of challenges.

Objectives / Topics Covered	● Solving large and more complex problems using graphics
Assignments / Labs	● 3 graphics challenges to tie everything learned in the JavaScript & Graphics module together ○ Example Exercise: Ghost Write a program to draw a ghost on the screen. You must do this by using the constant values given (this will allow us to easily alter the size or color of the ghost.)

Unit 5: JavaScript Control Structures (3 weeks/10-15 hours)

Students learn how to use control structures such as if/else statements and loops to make advanced programs in JavaScript.

Objectives / Topics Covered	● Booleans ● If/Else Statements ● Logical Operators ● Comparison Operators ● Conditionals ● While Loops ● Break Statements ● For Loops ● Nested Control Structures
Assignments / Labs	● 31 control structures programming exercises in total ● Using comparison and logical operators to control the flow of the program ○ Example Exercise: Inventory Write a program that keeps track of a simple inventory for a store. While there are still items left in the inventory, ask the user how many items they would like to buy. Then print out how many are left in inventory after the purchase. You should use a while loop for this problem. Make sure you catch the case where the user tries to buy more items than there are in the inventory. In that case, you should print a message to the user saying that their request isn't possible. ● Using for loops

	○ Example Exercise: Jukebox ■ In the days before the internet, many restaurants would have a jukebox that allowed customers to choose what music they wanted to play. Customers would enter a coin and choose from the jukebox's music collection by selecting a song's number. You could choose one song per coin. In this exercise, you will create a digital jukebox where the user can enter any number of quarters to create a playlist of songs. ● Drawing basic graphics using JavaScript ○ Example Exercise: Caterpillar This graphics program should draw a caterpillar. A caterpillar has NUM_CIRCLES circles. Every other circle is a different color, the even circles are red, and the odd circles are green (by even we mean when i is an even number). Use a for loop to draw the caterpillar, centered vertically on the screen. Also, be sure that the caterpillar is still drawn across the whole canvas even if the value of NUM_CIRCLES is changed.

Unit 6: Control Structures Challenges (1 week/3-5 hours)

Students apply the foundational concepts from the Control Structures module to solve new challenges.

Objectives / Topics Covered	● Solving large and more complex problems using control structures
Assignments / Labs	● 3 challenges using control structures to tie everything learned in the JavaScript Control Structures module together ○ Example Exercise: Guessing Game The computer picks a number between 1 and 100, and you have to guess it. The computer will tell you whether your guess was too high, too low, or correct. Your assignment is to generate a random number and let the user guess numbers until they guess the correct number. Make sure to let the user know what they should do at the beginning of the program!

Unit 7: Functions (2 weeks/5-10 hours)

Students learn to write reusable code with functions, parameters, and return values, and explore the impact of variable scopes.

Objectives / Topics Covered	● Parameters ● Return Values ● Default Parameters ● Scope
Assignments / Labs	● 12 functions programming exercises in total ● Using various kinds of functions such as functions with and without parameters, and functions with and without return values ○ Example Exercise: Is it even? Write a program that continually asks the user for integers and then prints whether their input is even or odd. The user should keep entering numbers until they enter 0; at that point, print "Done!" on its own line.

	In order to check if the inputted integer is even or odd, you should define and call a function named `isEven()`. This function should return a Boolean value of `true` or `false` depending if the number is even or not.

Unit 8: Functions Challenges (1 week/3-5 hours)

Students use what they have learned in the Functions module to solve new challenges.

Objectives / Topics Covered	● Solving large and more complex problems using functions
Assignments / Labs	● 3 challenges using functions to tie everything learned in the Functions module together ○ Example Exercise: Balloons You should use lines, circles, and random colors to draw a bunch of balloons. All the balloon strings should start two-thirds down the canvas. Each string line should travel upward to a random point and have a circle placed on top of the endpoint. Each balloon should be a random color and have a radius between `MIN_RADIUS` and `MAX_RADIUS`.

Unit 9: Animation and Games (3 weeks/10-15 hours)

Students learn how to make objects move around the screen and let users interact using the mouse!

Objectives / Topics Covered	● Timers ● Randomizing Games ● Mouse Events ● Keyboard Events
Assignments / Labs	● 19 animations programming exercises in total ● Throughout the lessons in this module, you will be developing a simple game that incorporates basic animation techniques and input events. ● Using timers to add randomizations to graphical programs ○ Example Exercise: Paint Splatter Write a program that splatters paint on the screen every DELAY milliseconds. To splatter paint, pick a random color and draw CIRCLES_PER_SPLATTER circles of that color at random places on the screen. The radius of each circle should be a random value between MIN_RADIUS and MAX_RADIUS. Remember to use helper functions. ● Using mouse events for interactive programs ○ Example Exercise: Target Draw a target on the screen that moves to aim at where your mouse is located. A target consists of a horizontal line that goes from 0 to the window width and a vertical line that goes from 0 to the window height. The lines should cross paths where the mouse is. If you're feeling adventurous, you can extend this to draw a small red circle whenever you click. If you're feeling really adventurous, you can have a bouncing ball on the screen and see if you can remove it when it gets clicked. You

	can use remove(obj) to remove something from the screen and getElementAt(x, y) to get an object at the given position. It will return the object or will return null if there is no object there. ● Using keyboard events for interactive programs ○ Example Exercise: Basic Snake Write a basic version of the snake game. The way our game works is by first creating a green square at the center of the screen. The snake should be moving to the right. If you hit an arrow key, you should change the snake's direction.

Unit 10: Animations Challenges (1 week/2-3 hours)

Students apply all the foundational concepts from the Animations module to solve new challenges.

Objectives / Topics Covered	● Solving additional problems using animation
Assignments / Labs	● 2 challenges using animation to tie everything learned in the Animation & Games module together ○ Example Exercise: Blinking Rectangles You should divide the canvas into an imaginary grid with `NUM_RECTANGLES_ACROSS` rectangles across, and `NUM_RECTANGLES_DOWN` rectangles down. Each time the user moves the mouse, a rectangle aligned with this grid should be drawn so that the mouse's location is within the rectangle. The rectangle should change color each time the mouse passes over it.

Unit 11: Project: Breakout (2 weeks/10 hours)

Students learn how to make their own Breakout game from scratch using JavaScript.

Objectives / Topics Covered	● Basic graphics ● Mouse events ● Collision detection
Assignments / Labs	● Guided exercises to build a Breakout Game ● Breakout is made up of bricks at the top of the screen, a paddle that you control at the bottom of the screen, and a ball that bounces around. Your goal is to direct the paddle with your mouse to bounce the ball until all of the bricks have been hit and disappear.

Module 12: Project: Snake (2 weeks/ 10 hours)

Students make their own interactive snake game through a series of guided exercises.

Objectives / Topics Covered	● Basic Graphics ● Key Events ● Collision Detection
Assignments / Labs	● Create your very own game of Snake where you control the ever elongating snake with the arrow keys to eat as much food and avoid crashing into your own body.

Module 13: Data Structures: Arrays (3 weeks/10-15 hours)

Students learn about arrays, how to iterate through them, and how to take advantage of their default methods.

Objectives / Topics Covered	● Array creation and basic operations ● Iterating through arrays ● Array methods
Assignments / Labs	● 17 exercises in total ● Basic array operations ○ Example Exercise: List of Places to Travel Create an array of the top 5 places you would like to travel called travelList. Print out the item at index 2. ● Iterating through arrays ○ Example Exercise: Draw a Barcode In this program, you will draw a barcode on the screen given an array that represents the data in the barcode. ● Array methods ○ Example Exercise: Mutual Friends In this program, you are going to create function that will find the mutual friends between two array lists of friends.

Module 14: Data Structures: Objects (3 weeks/10-15 hours)

Students learn about objects, how to create object properties and methods, iterate through them, and build constructors.

Objectives / Topics Covered	● Basic Objects and Object Properties ● Object Methods ● Iterating through an Object ● Object Constructors
Assignments / Labs	● 17 exercises in total ● Basic Objects and properties ○ Example Exercise: Shopping Cart In this activity, you are going to create a digital shopping cart program that is able to add item objects, print the list of items, and calculate the total cost. ● Object methods ○ Example Exercise: Bank Account In this exercise, you need to define a function `createAccount()` that creates a new bank account object and returns it. ● Object iteration ○ Example Exercise: Starry Night In this program, you have been given an `artwork` object with properties and a method. You need to complete the two functions `printAllKeys()` and `printProperties()` so that the former prints every key in the parameter object and the latter prints only the key-value pairs of the properties (i.e. no methods). ● Object Constructors ○ Example Exercise: Hobby Constructors Choose a hobby, activity, or topic that you are interested in and make an object constructor related to your choice. Then create at least two instances of the object and demonstrate the functionality of their properties and methods.

Module 15: Project: Tic Tac Toe (2 weeks/5-10 hours)

Students create their own game of Tic-Tac-Toe by applying what they have learned about data structures.

Objectives / Topics Covered	● Using data structures to solve a problem ● Combining data structures and graphics
Assignments / Labs	● Guided exercises to build a game of Tic Tac Toe

Module 16: Project: Helicopter (2 weeks/5-10 hours)

Students demonstrate their programming prowess as they develop the classic Helicopter game one step at a time.

Objectives / Topics Covered	● Basic Graphics ● Collision detection ● Scrolling background ● Generating random obstacles
Assignments / Labs	● Guided exercises to explain the basic elements of game design and build a Helicopter Game. ● Helicopter Game is played by controlling a helicopter with the mouse to navigate through a changing terrain with flying obstacles.

Module 17: Final Project: Your Own Game (4 weeks/20 hours)

Students apply the skills they've learned throughout the course to create an original game!

Objectives / Topics Covered	● Basic Graphics ● Collision detection ● Scrolling background ● Generating random obstacles
Assignments / Labs	● Use everything you've learned to plan and build your own game!