Wethersfield Public Schools

WHS Curriculum: Video Game Development

Grade(s)	9, 10, 11, 12 Honors
Unit Title and Purpose	**Unit 1: Intro to Programming in JavaScript** Students dive into the history of computing, consider how computing impacts today's world, and learn about the various parts that make up modern computers.
Timeframe	18 hours

Vision of the Graduate

Problem Solver: In this unit, students problem-solve large problems by using top-down design.

Unit Priority Standards

CSTA Standards

3A-AP-18. Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

3B-AP-16. Demonstrate code reuse by creating programming solutions using libraries and APIs.

3A-AP-18. Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

3B-AP-16. Demonstrate code reuse by creating programming solutions using libraries and APIs.

Unit Supporting Standards

Common Core Standards

CCSS.ELA-Literacy.RST.9-10.3. Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text.

CCSS.ELA-Literacy.RST.9-10.4. Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to grades 9-10 texts and topics.

CCSS.ELA-Literacy.RST.9-10.5. Analyze the structure of the relationships among concepts in a text, including relationships among key terms.

Essential Questions

What is a computer and what are they used for? What is the difference between a computer and computing?

What is programming and why is it important?

What are variables, and how do we use them to store and manipulate data?

What are data types, and why is it important to know which type of data we are working with?

What are operators, and how do we use them to perform calculations or comparisons in code?

What is a loop, and how can we use it to repeat tasks in a program?

What is a function, and how does it allow us to organize and reuse code?

Performance Expectations: Skills	Performance Expectations: Essential Knowledge/Concepts
1. Write and execute basic code 2. Use variables and constants 3. Work with data types 4. Use arithmetic and logical operators 5. Write conditional statements	1. Explain what a computer is and what they are used for. 2. Explain the difference between a computer and computing. 3. Understand what variables are and how they are

6. Implement loops 7. Define and use basic functions 8. Debug and troubleshoot code 9. Apply top-down design approaches to solve problems	used to store and manipulate data. 4. Identify operators and how they are used to perform calculations and comparisons 5. Understand what a loop is and how it can be implemented in code to repeat tasks. 6. Define what a function is and how it is used to create efficient, organized code. 7. Benefits, uses, and processes of top-down design for programming.
Student Learning Tasks & Resources	**Suggested Teacher Materials & Resources**
● CodeHS tasks to build understanding of: Commands; Defining vs. Calling Methods; Designing methods; Program entry points; Control flow; Looping; Conditionals; Commenting code; Preconditions and Postconditions; Top Down Design. ● Program-specific tasks while writing a program ● Create new commands by coding functions ● Solve large problems by utilizing top-down design ● Use control structures and conditionals to solve general problems	● CodeHS platform containing videos, slides handouts, exercises, coding IDE, and quizzes.

Grade(s)	9, 10, 11, 12 Honors
Unit Title and Purpose	**Unit 2: JavaScript Basics** Students learn the basics of JavaScript including variables, user input, mathematics, and functions.
Timeframe	4 hours

Vision of the Graduate

Communicator, **Problem Solver**, and **Collaborator**: In this unit, students work in pairs to tackle coding exercises while developing communication, collaboration and problem-solving skills to get user data to create a Dinner Plans program.

Unit Priority Standards

CSTA Standards

2-AP-11. Create clearly named variables that represent different data types and perform operations on their values.
2-AP-14. Create procedures with parameters to organize code and make it easier to reuse.
2-AP-19. Document programs in order to make them easier to follow, test, and debug.
3A-AP-13. Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
3A-AP-17. Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
3A-AP-23. Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.
3B-AP-16. Demonstrate code reuse by creating programming solutions using libraries and APIs.
3A-DA-11. Create interactive data visualizations using software tools to help others better understand real-world phenomena.
3A-DA-12. Create computational models that represent the relationships among different elements of data collected from a phenomenon or process.

Unit Supporting Standards

Common Core Standards

CCSS.ELA-Literacy.RST.9-10.3. Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text.
CCSS.ELA-Literacy.RST.9-10.4. Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to grades 9-10 texts and topics.
CCSS.ELA-Literacy.RST.9-10.5. Analyze the structure of the relationships among concepts in a text, including relationships among key terms.
CCSS.ELA-Literacy.RST.9-10.7. Translate quantitative or technical information expressed in words in a text into visual form (e.g., a table or chart) and translate information expressed visually or mathematically (e.g., in an equation) into words.

Essential Questions

What is programming and why is JavaScript an important language?

How do variables and data types work in JavaScript, and why are they important?

What is the role of functions in programming, and how do they help us write more efficient code?

How do conditional statements like "if" and "else" allow a program to make decisions?

How do loops work in JavaScript, and when would we use them to solve a problem?

What does it mean to debug code, and why is debugging an essential skill for programmers?

Performance Expectations: Skills	Performance Expectations: Essential Knowledge/Concepts
1. Write and execute code in JavaScript 2. Use variables and constants 3. Work with data types in JavaScript 4. Use arithmetic and logical operators 5. Write conditional statements for JavaScript 6. Implement JavaScript loops 7. Define and use functions in JavaScript 8. Debug and troubleshoot JavaScript code	1. Explain what programming is and why JavaScript is important. 2. Understand what variables are and how they are used in JavaScript to store and manipulate data. 3. Identify JavaScript functions and understand how they are used to make code more efficient. 4. Recognize how JavaScript conditional statements like "if" and "else" allow a program to make decisions. 5. Explain how loops work in JavaScrip, and evaluate when they should be used to solve a problem. 6. Define a function in JavaScript and how it is used to create efficient, organized code. 7. Understand what it means to debug code, and why it is an essential skill for programmers.
Student Learning Tasks & Resources	**Suggested Teacher Materials & Resources**
● CodeHS tasks to build understanding of: Variables; User Input; Arithmetic Expressions; Constants; Collaborative Programming; Random Numbers; and Functions. ● 12 JavaScript programming exercises (including pair-programming activities) ● Using variables and getting user input using JavaScript ● 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.	● CodeHS platform containing videos, slides handouts, exercises, coding IDE, and quizzes.

Grade(s)	9, 10, 11, 12 Honors
Unit Title and Purpose	**Unit 3: The Canvas and Graphics** Students learn how to add graphics objects and position them on the canvas.
Timeframe	2-4 hours

Vision of the Graduate

Communicator, **Problem Solver**, and **Collaborator**: In this unit, students work on a pair-programming activity developing their communication, collaboration and problem-solving skills while creating a program with canvas coordinates.

Unit Priority Standards

CSTA Standards

3A-AP-13. Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

3A-AP-16. Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.

3A-AP-18. Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

Unit Supporting Standards

Common Core Standards

CCSS.ELA-Literacy.RST.9-10.3. Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text.

CCSS.ELA-Literacy.RST.9-10.4. Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to grades 9-10 texts and topics.

CCSS.ELA-Literacy.RST.9-10.7. Translate quantitative or technical information expressed in words in a text into visual form (e.g., a table or chart) and translate information expressed visually or mathematically (e.g., in an equation) into words.

CSTA Standards

3A-AP-22. Design and develop computational artifacts working in team roles using collaborative tools.

Essential Questions

How can we use JavaScript to draw shapes and create graphics on a webpage?

What are the core methods and properties and how can we use them to manipulate graphics?

How does the coordinate system work, and how do we position shapes on the canvas using X and Y coordinates?

How can we solve large and more complex problems using graphics?

Performance Expectations: Skills	Performance Expectations: Essential Knowledge/Concepts
1. Create objects with set sizes, colors, positions and coordinates 2. Locate the default anchor point of a text object 3. Create a personal meme using JavaScript objects on a canvas	1. Describe how JavaScript can be used to draw shapes and create objects 2. Identify canvas coordinates 3. Explain what the x and y axis are, and how coordinates are used to position shapes

4. Use built-in functions and cast variables 5. Add images to their graphics projects using WebImage 6. Resize image objects using setSize 7. Display text on the canvas 8. Break their code into functions based on objects to be rendered	4. Explain how methods and properties are used to manipulate graphics 5. Summarize how we can use graphics to solve complex problems
Student Learning Tasks & Resources	**Suggested Teacher Materials & Resources**
● CodeHS tasks on: Intro to the Canvas and Graphics; More Graphic Objects; Positioning Graphics Objects ● Pair-Programing Activities: Canvas Coordinates; Creating a Circle and A Circle and a Rectangle; A Ball in a Box; Raise the Flag. ● Individual Activities: Graphics Objects; Exploration: XY Plot; Create Your Meme; Saturday Mornings.	● CodeHS platform containing videos, slides handouts, exercises, coding IDE, and quizzes.

Grade(s)	9, 10, 11, 12 Honors
Unit Title and Purpose	**Unit 4: Graphics Challenges** Students apply what they have learned about graphics and basic JavaScript to complete a set of challenges.
Timeframe	2-4 hours

Vision of the Graduate

Communicator and **Problem Solver**: In this unit, students communicate how breaking complex problems into smaller, manageable tasks can help create more efficient solutions.

Unit Priority Standards

CSTA Standards

3A-IC-24. Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.

3A-AP-13. Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

3A-AP-14. Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.

Unit Supporting Standards

Common Core Standards

CCSS.ELA-Literacy.RST.9-10.1. Cite specific textual evidence to support analysis of science and technical texts, attending to the precise details of explanations or descriptions.

CCSS.ELA-Literacy.RST.9-10.3. Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text.

CCSS.ELA-Literacy.RST.9-10.4. Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to grades 9-10 texts and topics.

Essential Questions

How can breaking down a complex problem into smaller, manageable tasks help create a more efficient solution?

What makes a program easy to read and understand, and why is clarity in coding important for collaboration and future development?

Performance Expectations: Skills	Performance Expectations: Essential Knowledge/Concepts
1. Define a problem in their own words and plan out a solution to the problem 2. Break a large problem down into smaller pieces and solve each of the pieces, then use these solutions as building blocks to solve the larger problem 3. Write clear and readable graphics programs	1. Essential steps in problem-solving processes
Student Learning Tasks & Resources	**Suggested Teacher Materials & Resources**
● CodeHS tasks building understanding of: Solving large and more complex problems using graphics. ● Labs: 3 graphics challenges to apply and synthesize learning in the JavaScript & Graphics modules	● CodeHS platform containing videos, slides handouts, exercises, and coding IDE.

• 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).	

Grade(s)	9, 10, 11, 12 Honors
Unit Title and Purpose	**Unit 5: JavaScript Control Structures** Students learn how to use control structures such as if/else statements and loops to make advanced programs in JavaScript.
Timeframe	8-10 hours

Vision of the Graduate

Problem Solver: In this unit, students problem-solve control structures by utilizing if/else statements and loops.

Unit Priority Standards

CSTA Standards

3A-AP-13. Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

3A-AP-16. Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.

3A-AP-17. Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

3A-AP-23. Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.

Unit Supporting Standards

Common Core Standards

CCSS.ELA-Literacy.RST.9-10.3. Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text.

CCSS.ELA-Literacy.WHST.9-10.1. Write arguments focused on discipline-specific content.

CSTA Standards

3B-AP-16. Demonstrate code reuse by creating programming solutions using libraries and APIs.

Essential Questions

How do control structures, such as if/else statements and loops, help us manage the flow of a program?

What are the differences between various types of loops (while loops vs. for loops), and when should each be used in programming?

How can nested control structures simplify or complicate program logic, and how can we manage that complexity?

How can visual programming, such as drawing shapes or graphics, help us understand the relationship between code and visual outputs?

Performance Expectations: Skills	Performance Expectations: Essential Knowledge/Concepts
1. Create boolean variables to represent meaningful yes/no values. 2. Print out the value of a boolean variable.	1. Describe how control structures help manage the flow of a program. 2. Identify the differences between types of loops. 3. Recognize how nested control structures can simplify or complicate program logic. 4. Understand how visual programming can show

	the relationship between code and visual outputs.
Student Learning Tasks & Resources	**Suggested Teacher Materials & Resources**
● CodeHS tasks on unit topics. ● Students write out the Boolean statement on the board based on student answers. They complete one as a class, then practice writing the rest of the Boolean statements on their own. ● Do You Have a Dog? exercise. ● Booleans are Questions free-response questions. ● Best Day Ever exercise.	● CodeHS platform containing videos, slides handouts, exercises, coding IDE, and quizzes.

Grade(s)	9, 10, 11, 12 Honors
Unit Title and Purpose	**Unit 6: Control Structures Challenges** Students apply the foundational concepts from the Control Structures module to solve new challenges.
Timeframe	2-4 hours

Vision of the Graduate

Communicator, **Problem Solver**, and **Collaborator**: In this unit, students work on three control structure challenges by utilizing their problem-solving skills. They communicate with their classmates as they get their user input to test their programs. Lastly, students collaborate to debug their final programs.

Unit Priority Standards

CSTA Standards

3A-AP-13. Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

3A-AP-15. Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.

3A-AP-21. Evaluate and refine computational artifacts to make them more usable and accessible.

Unit Supporting Standards

Common Core Standards

CCSS.ELA-Literacy.RST.9-10.1. Cite specific textual evidence to support analysis of science and technical texts, attending to the precise details of explanations or descriptions.

CCSS.ELA-Literacy.RST.9-10.4. Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to grades 9-10 texts and topics.

CCSS.ELA-Literacy.RST.9-10.5. Analyze the structure of the relationships among concepts in a text, including relationships among key terms (e.g., force, friction, reaction force, energy)

CCSS.ELA-Literacy.RST.9-10.7. Translate quantitative or technical information expressed in words in a text into visual form (e.g., a table or chart) and translate information expressed visually or mathematically (e.g., in an equation) into words.

CCSS.ELA-Literacy.RST.9-10.9. Compare and contrast findings presented in a text to those from other sources (including their own experiments), noting when the findings support or contradict previous explanations or accounts.

Essential Questions

How can breaking a large problem into smaller, manageable tasks make the problem easier to solve and lead to more efficient solutions?

Why is it important to clearly define a problem and plan a solution before starting to write code, and how does this affect the overall programming process?

How do we write clear and readable code using control structures, decomposition, and comments to ensure our solutions are scalable, efficient, and easy to maintain?

Performance Expectations: Skills	Performance Expectations: Essential Knowledge/Concepts
1. Define a problem in their own words and plan out a solution to the problem 2. Break a large problem down into smaller pieces and solve each of the pieces, then use these solutions as	1. Approaches to writing readable code using control structures, decomposition, and comments

building blocks to solve the larger problem 3. Write clear and readable code using control structures, decomposition, and comments 4. Utilize the proper control structures to create general solutions	
Student Learning Tasks & Resources	**Suggested Teacher Materials & Resources**
● CodeHS tasks on control structures. ● Labs: Guessing Game, Landscape Generator*, and Exploring RGB Color Codes. *These activities rely on a mixture of global and local variables.	● CodeHS platform containing videos, slides handouts, exercises, coding IDE, and quizzes.

Grade(s)	9, 10, 11, 12 Honors
Unit Title and Purpose	**Unit 7: Functions** Students learn to write reusable code with functions, parameters, and return values, and explore the impact of variable scopes
Timeframe	4-8 hours

Vision of the Graduate

Problem Solver: In this unit, students will problem-solve code with various kinds of functions such as functions with and without parameters, and functions with and without return values.

Unit Priority Standards

CSTA Standards

3A-AP-13. Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

3A-AP-10. Use flowcharts and/or pseudocode to address complex problems as algorithms.

3A-AP-11. Create clearly named variables that represent different data types and perform operations on their values.

Unit Supporting Standards

Common Core Standards

CCSS.ELA-Literacy.RST.9-10.3. Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text.

CCSS.ELA-Literacy.RST.9-10.9. Compare and contrast findings presented in a text to those from other sources (including their own experiments), noting when the findings support or contradict previous explanations or accounts.

CCSS.ELA-Literacy.WHST.9-10.1. Write arguments focused on discipline-specific content.

Essential Questions

How do functions, parameters, and return values help us create reusable and modular code?

How does the concept of variable scope affect the behavior and efficiency of a program?

How can we design functions that handle different inputs and return useful outputs to solve specific problems?

Performance Expectations: Skills	Performance Expectations: Essential Knowledge/Concepts
1. Create functions that take in parameters as input. 2. Use parameters to generalize functions and reduce repeated code. 3. Create functions that return values. 4. Create programs that call functions with return values and store the result for later use. 5. Set default values for their parameters. 6. Properly set the order of parameters and the default values.	1. Explain the use of parameters and arguments. 2. Explain the purpose of returning a value from a function. 3. Understand the role default values can have in a function. 4. Identify the scope of a variable. 5. Identify which variables are in scope at a given point in a program.
Student Learning Tasks & Resources	**Suggested Teacher Materials & Resources**
● CodeHS tasks to build understanding of parameters; return values; default parameter values; and variable scopes.	● CodeHS platform containing videos, slides handouts, exercises, coding IDE, and quizzes.

● 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 ● 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.	

Unit Title and Purpose	**Unit 8: Functions Challenges** Students use what they have learned in the Functions module to solve new challenges.
Timeframe	2-4 hours

Vision of the Graduate

Communicator, **Problem Solver**, and **Collaborator**: In this unit, students will collaborate, communicate and problem-solve with their classmates on three challenges using functions to tie everything learned in the Functions module together.

Unit Priority Standards

CSTA Standards

3A-AP-13. Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

3A-AP-17. Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

3A-AP-23. Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.

3B-AP-12. Compare and contrast fundamental data structures and their uses.

3B-AP-14. Construct solutions to problems using student-created components, such as procedures, modules and/or objects.

3B-AP-16. Demonstrate code reuse by creating programming solutions using libraries and APIs.

Unit Supporting Standards

Common Core State Standards

CCSS.ELA-Literacy.RST.9-10.4. Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to grades 9–10 texts and topics.

CCSS.ELA-Literacy.RST.9-10.5. Analyze the structure of the relationships among concepts in a text, including relationships among key terms (e.g., force, friction, reaction force, energy).

CCSS.ELA-Literacy.RST.9-10.7. Translate quantitative or technical information expressed in words in a text into visual form (e.g., a table or chart) and translate information expressed visually or mathematically (e.g., in an equation) into words.

CCSS.ELA-Literacy.WHST.9-10.2. Write informative/explanatory texts, including the narration of scientific procedures/experiments, or technical processes.

Essential Questions

How can breaking a complex problem into smaller, manageable parts using top-down design help us create efficient and effective solutions?

How do we use functions, parameters, and return values to create reusable, modular code that can be applied to different challenges?

What role does debugging play in solving programming problems, and how can we systematically find and fix errors in large programs?

Performance Expectations: Skills	Performance Expectations: Essential Knowledge/Concepts
1. Synthesize the skills and concepts from The Canvas and Graphics, Control Structures, and Functions modules to solve increasingly difficult programming	1. Explain how breaking down a complex problem can make code more efficient. 2. Recognize how functions, parameters and return

challenges. 2. Break down a large problem into smaller parts using Top Down Design, and solve each of these smaller parts using functions 3. Create helpful comments with preconditions and postconditions to help the reader understand the code. 4. Find and fix bugs in large programs.	values can be reused and applied to other challenges. 3. Define how debugging plays an important role in solving programming problems. 4. Explain the use of parameters and arguments.
Student Learning Tasks & Resources	**Suggested Teacher Materials & Resources**
● CodeHS tasks to build understanding of: Solving large and more complex problems using functions. ● Labs: 3 challenges using functions to synthesize learning from the Functions module. ● Exercise: Balloons. 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`.	● CodeHS platform containing videos, slides handouts, exercises, coding IDE, and quizzes.

Unit Title and Purpose	**Unit 9: Animation and Games** Students learn how to make objects move around the screen and let users interact using the mouse.
Timeframe	8-11 hours

Vision of the Graduate

Communicator, **Problem Solver**, and **Collaborator**: In this unit, students create and problem-solve a game that incorporates basic animation techniques and input events. They collaborate with a classmate to test their product and communicate any improvements with each other.

Unit Priority Standards

CSTA Standards

3A-AP-17. Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

3A-AP-18. Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

3A-AP-23. Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.

3B-AP-16. Demonstrate code reuse by creating programming solutions using libraries and APIs.

3B-AP-19. Develop programs for multiple computing platforms.

Unit Supporting Standards

Common Core Standards

CCSS.ELA-Literacy.RST.9-10.3. Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text.

CCSS.ELA-Literacy.RST.9-10.4. Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to grades 9–10 texts and topics.

CCSS.ELA-Literacy.RST.9-10.7. Translate quantitative or technical information expressed in words in a text into visual form (e.g., a table or chart) and translate information expressed visually or mathematically (e.g., in an equation) into words.

CCSS.ELA-Literacy.RST.9-10.9. Compare and contrast findings presented in a text to those from other sources (including their own experiments), noting when the findings support or contradict previous explanations or accounts.

Essential Questions

How do timers and callback functions work together to create smooth, interactive animations in programming?

What are the key steps involved in transforming static graphics into animated visuals, and how does each step contribute to the final result?

How can clear and effective documentation, such as commenting and explaining the code, help others (and yourself) understand and improve your animated programs?

Performance Expectations: Skills	Performance Expectations: Essential Knowledge/Concepts
1. Create animation in programs using the setTimer function. 2. Create programs with timers to create increasingly challenging animations. 3. Stop animation timers when a condition is met using	1. Explain how animation works. 2. Explain what a callback function is. 3. Understand when objects “collide” with the canvas walls and other objects. 4. Explain and describe how events are different

stopTimer() function. 4. Write collision detection logic. 5. Use mouse click events to create programs that respond to user clicks. 6. Create interactive programs that use events to respond to the mouse moving. 7. Create interactive programs that use events to respond to the keyboard input.	from timers.
Student Learning Tasks & Resources	**Suggested Teacher Materials & Resources**
● CodeHS tasks to deepen understanding of Timers; Randomizing Games; Mouse Events; and Keyboard Events: ● Labs: 19 animations programming exercises in total. ● Students develop a simple game that incorporates basic animation techniques and input events. ● Students use timers to add randomizations to graphical programs. ● Students use mouse events for interactive programs and keyboard events for interactive programs.	● CodeHS platform containing videos, slides handouts, exercises, coding IDE, and quizzes.