Lindenwold Public Schools

Unit 5: Building Apps
January-March

Targeted Standards: K12 Computer Science Standards

- Computing Systems - Devices, Hardware and Software, Troubleshooting
- Data and Analysis - Collection, Storage, Visualization and Transformation, Inference and Models
- Algorithms and Programming - Algorithms, Variables, Control, Modularity, Program Development
- Impacts of Computing - Culture, Social Interactions, Safety, Law, and Ethics

(also reference CSTA K-12 Computer Science Standards)

Rationale and Transfer Goals :

This unit continues to develop students' ability to program in the JavaScript language, using Code.org's App Lab environment to create a series of small applications (apps) that live on the web, each highlighting a core concept of programming. In this unit students transition to creating event-driven apps. The unit assumes that students have learned the concepts and skills from Unit 3, namely: writing and using functions, using simple repeat loops, being able to read documentation, collaborating, and using the Code Studio environment with App Lab.

Enduring Understandings:

- Creative development can be an essential process for creating computational artifacts.
- Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.
- Computing can extend traditional forms of human expression and experience.
- Multiple levels of abstraction are used to write programs or create other computational artifacts
- Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.
- Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (to help people, organizations, or society).
- People write programs to execute algorithms.
- Programming is facilitated by appropriate abstractions.

Lindenwold Public Schools

- Programs are developed, maintained, and used by people for different purposes.
- Programming uses mathematical and logical concepts.
- Computing enhances communication, interaction, and cognition.
- Models and simulations use abstraction to generate new understanding and knowledge.
- People use computer programs to process information to gain insight and knowledge.
- Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.
- Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (to help people, organizations, or society).

Essential Questions:

- How do you program apps to respond to user "events"?
- How do you write programs to make decisions?
- How do programs keep track of information?
- How creative is programming?
- How do people develop, test, and debug programs?
- How are real world phenomena modeled and simulated on a computer?
- How do you write programs to store and retrieve lots of information?
- What are "data structures" in a program and when do you need them?
- How are algorithms evaluated for "speed"?

Content/Objectives		Instructional Actions	
Content ***What students will know***	**Skills** ***What students will be able to do***	**Activities/Strategies** ***How we teach content and skills***	**Evidence (Assessments)** ***How we know students have learned***
• Creative development can be an essential process for creating computational artifacts.	• Use Design Mode to user interface (UI) elements to a screen. • Create a simple event-driven program by	• Programming • Conditionals • App Lab • Unplugged • App Creation	• Use a Gallery Walk, Pair-Share, or other strategy to allow students to share their Chaser Games with each other.

• Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem. • Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (to help people, organizations, or society). • Programs are developed, maintained, and used by people for different purposes. • People write programs to execute algorithms. • Programming is facilitated by appropriate abstractions. • Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.	creating user-interface elements with unique IDs and attaching event handlers to them. • Recognize debugging and responding to error messages as an important step in developing a program. • Debug simple issues related to event-driven programming • Write a simple event-driven program that has multiple screens. • Recognize debugging as an important step in developing a program. • Use console.log to debug simple issues related to event-driven programming. • Develop and design a plan for multi-screen application • Collaborate with a "thought partner" during the implementation of a project • Create a multi-screen application in App Lab		Encourage students to note design features they would want to include in future applications they create. • Elements in your app are required to have unique IDs. Given what you now know about how event handlers work, why is it important for the IDs of page elements to be unique? • Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development; the second

• Programming uses mathematical and logical concepts. • People use computer programs to process information to gain insight and knowledge. • Models and simulations use abstraction to generate new understanding and knowledge. • Algorithms can solve many but not all computational problems. • Multiple levels of abstraction are used to write programs or create other computational artifacts • Computing can extend traditional forms of human expression and experience.	using simple UI elements and event handling • Use variables in a program to store numeric values. • Store the value returned by a function (randomNumber, promptNum) in a variable for use in a program. • Debug problems related to variable re-assignment. • Write arithmetic expressions that involve variables. • Reason about multi-line segments of code in which variables are re-assigned multiple times. • Use global variables to track numeric data in an app. • Give a high-level explanation of what “variable scope” means. • Debug problems related to variable scoping issues. • Modify existing programs to add and update variables to track information.		could refer to either collaborative or independent program development. (Approximately 200 words) • What made it hard was that you needed to check more than one condition at the same time. You needed to say "it's saturday OR sunday". That's more than one condition to check. • In your own words, describe how a while loop works. Explain two things to pay attention to when creating while loops. In your response, justify why the name "while loop" accurately describes the behavior of this new programming construct • In general, when do you think you should you store information in an array, and when should you use a variable? • In today’s activity, we needed to make some changes to our programs

	• Create a multi screen "clicker" game from scratch • Identify strings as a unique data type which contains a sequence of ASCII characters. • Describe characteristics of the string data type. • Accept string input in a program. • Manipulate user-generated string input to generate dynamic output. • Reason about if-statements by tracing pseudocode programs by hand • Write a short program in pseudocode that uses if statements • Explain the purpose of if-statements in programs • Write and test conditional expressions using comparison operations • Given an English description write code (if statements) to create desired program logic		in order to incorporate new functionality. Sometimes this meant we needed to make changes to our old code as well. • Why would we prefer to write a function that returns a value to using the strategy shown above? How might return values make our function more generally useful? How might they make our code easier to reason about? • We've seen a few ways to process our array of events over the course of this lesson, but there are many other effects we could produce. How else could we use the information we stored in our array? What other effects do you think we could make?

	• Use the comparison operators (<, >, <=, >=, ==, !=) to implement decision logic in a program. • When given starting code add if, if-else, or nested if statements to express desired program logic • Write and test conditional expressions using Boolean operators AND (&&) OR (||) and NOT (!) • Given an English description write compound conditional expressions to create desired program logic • Use a "chain" of if-else-if statements to implement desired program logic • When given starting code add if-else-if statements or compound boolean expression to express desired program logic • Write code to implement solutions to problems from pseudocode or description • Follow the iterative development process of a		

	collaboratively created program ● Develop and write code for conditional expressions to incorporate into an existing program ● Write a large program from scratch when given directions for each step ● Explain that a while loop continues to run while a boolean condition remains true. ● Translate a real-life activity with repeated components into a form that could be represented by a while loop. ● Analyze a while loop to determine if the initial condition will be met, how many times the loop will run, and if the loop will ever terminate. ● Write programs that use while loops in a variety of contexts. ● Use a while loop in a program to repeatedly call a block of code.		

	● Use variables, iteration, and conditional logic within a loop to record the results of a repeated process. ● Identify instances where a simulation might be useful to learn more about real-world phenomena. ● Develop a simulation of a simple real-world phenomenon. ● Identify an array as a data structure used to store lists of information in programs. ● Create arrays and access information stored within them using an index. ● Manipulate an array using the append, insert, and remove operations. ● Account for the fact that JavaScript arrays are zero-indexed when using them in a program. ● Use an array to maintain a collection of data in a program.		

	● Create apps that allow user interaction through key events. ● Refactor code in order to appropriately incorporate new functionality while maintaining readability and consistency. ● Use a for loop in a program to implement an algorithm that processes all elements of an array. ● Write code that implements a linear search on an unsorted array of numbers. ● Write code to find the minimum value in an unsorted list of numbers. ● Explain how binary search is more efficient than linear search but can only be used on sorted lists. ● Use the return command to design functions. ● Identify instances when a function with a return value can be used to contain frequently used computations within a program.		

	• Design functions that return values to perform frequently needed computations within a program. • Programmatically control the canvas element in response to user interactions. • Maintain a dynamically generated array through the running of a program in order to record and reuse user input. • Use nested loops within a program to repeat a command on the same array index multiple times. • Perform variable arithmetic within an array index to access items in an array by their relative position. • Complete reflection questions in a format similar to those on the AP performance tasks. • Collaborate to give and receive feedback on program implementation		

	to improve program functionality. ● Update existing code to add new functionality to a program. ● Create a video demonstrating the functionality of a program.		

<u>Spiraling for Mastery</u>
Where does this unit spiral back to other units from this or previous years in order to ensure that students retain mastery of what they've learned?

Content or Skill for this Unit	Spiral Focus from Previous Unit	Instructional Activity
● Programmers consider tradeoffs related to implementation, readability, and program performance when selecting and combining control structures. ● Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and procedures; or independent, but interrelated, programs. Modules	● Programmers select and combine control structures, such as loops, event handlers, and conditionals, to create more complex program behavior. ● Programs use procedures to organize code, hide implementation details, and make code easier to reuse. Procedures can be repurposed in new programs. Defining parameters for procedures can generalize	● Controlling Memory with Variables ● User Input and Strings

allow for better management of complex tasks. ● Diverse teams can develop programs with a broad impact through careful review and by drawing on the strengths of members in different roles. Design decisions often involve tradeoffs. The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program. Systematic analysis is critical for identifying the effects of lingering bugs.	behavior and increase reusability. ● People design meaningful solutions for others by defining a problem's criteria and constraints, carefully considering the diverse needs and wants of the community, and testing whether criteria and constraints were met.	● Building an App: Color Sleuth

21st Century Skills: What are the 21st Century Skills that are a part of this unit, and where are they experienced?

- Global awareness
- Creativity and Innovation
- Critical Thinking and Problem Solving
- Communication and Collaboration
- Information Literacy
- Flexibility and Adaptability
- Initiative and Self Direction

These skills are experienced throughout unplugged and plugged activities that will involve individual, group, and whole class discussion.

Key resources: What are the resources that are essential for this unit (may also be listed in "Activities/Strategies")?

- Unit 5 - Code.org Computer Science Principles Curriculum
- Code Studio