THE ALGEBRA INSIDE YOUR VIDEO GAME: COMPUTER SCIENCE IN MATH

Many students love video games, but how many really understand what it takes to make one? In Code.org’s Computer Science in Algebra, we pull back the veil on Computer Science and reveal to students the underlying fundamental math that makes everything tick. Using an easy to learn block based language, students will learn to create images, animations, and interactive video games that are driven by algebraic functions. Through the process of developing their video games, students will discover how computer programmers use the coordinate plane, algebraic functions, and logic to create the games that they love.

Code.org partnered with Bootstrap to develop an exciting and engaging curriculum that allows students to discover that is built on proven pedagogy and practices.

FUNCTIONAL PROGRAMMING IS MATH!

The Functional Programming paradigm used in this course will look very familiar to algebra teachers, because the language operates just like algebraic functions do! Students can write functions (both algebraic and programmatic) to solve for word problems, such as:

PROBLEM #1

A rocket blasts off, traveling at 7 meters per second. Write a function called rocket-height that takes in the number of seconds that have passed since the rocket took off, and which produces the height of the rocket at that time.

\[
\text{rocket-height(seconds)} = \text{seconds} \times 7
\]

IT’S MORE THAN JUST CODING

Algebra goes beyond just solving for \(x\), and Code.org CS in Algebra goes beyond this writing code. Through learning to program, students will also practice problem decomposition, clear communication through documentation, testing their own functions against example cases and input/output tables, and much more. Many of the practices employed by CS in Algebra directly correlate to key concepts in the Common Core math standards: data types teach domain and range, test cases introduce students to input/output tables, and our block based programming language allows students to write functions with manipulatives.

Code.org’s CS in Algebra curriculum aligns to the Common Core Math Standards and Practices, as well as the
*CSTA Computer Science Standards* and *ISTE* standards.

Across the entire CS in Algebra curriculum, students will develop the skills of a computer scientist through the development of *Computational Thinking Practices*, including Creativity, Collaboration, Communication, Persistence, and Problem Solving.

<table>
<thead>
<tr>
<th>Unit of Study</th>
<th>Description</th>
<th>Standards</th>
</tr>
</thead>
</table>
| **Videogames and Coordinate Planes**  | Students reverse-engineer a simple videogame into a series of coordinates, and explore Cartesian space. Once they are comfortable with coordinates, they brainstorm their own games and begin programming simple arithmetic expressions with Evaluation Blocks. | A-SSE.1.2  
N-Q  
6.NS.6  
6.NS.8 |
| **Contracts, Strings and Images**      | Students are introduced to a set-mapping representation for functions, in which the function translates points from a Domain to a Range. Students generalize their understanding of functions to include Strings and Images.                                                                 | F-IF.1-3  |
| **Defining Functions**                | Students define values of various types, as well as linear functions.                                                                                                                                   | A-SSE.1-2  
F-IF.1-3  
7.EE.4.a  
F-BF.1-2 |
| **The Design Recipe**                 | Students are introduced to the process for deriving functions from Word Problems using worked-through examples, called the Design Recipe.                                                                  | 8.F.1  
F-IF.1-3  
F-IF.4-6 |
| **Game Animation**                    | Students solve Word Problems that describe animation, and define functions that map character positions in their game from one frame to the next.                                                            | F-LE.5  
F-IF.7-9 |
| **Boundary Detection**                | Students discover Boolean types and inequalities in the plane, and use them to create programs that test locations of a character on the screen. They then solve Word Problems that deal with Boundary-Detection, writing code to detect when a character has gone off-screen. | 6.EE.b.5  
7.EE.3  
8.IF.1  
A-CED.1-4 |
| **Piecewise Functions**               | Students use geometry and conditional branching to move their player characters in response to key-presses. The Word Problem for key-events describes a function that behaves differently under different sub-domains, requiring students to learn about Piecewise Functions. | A-SSE.1-2  |
| **Collision Detection**               | Students derive, discuss, and prove the Pythagorean theorem, then use this theorem—in conjunction with Booleans—to detect collisions in their games.                                                        | 8.G.6  
F-IF.1-3  
F-IF.4-6 |