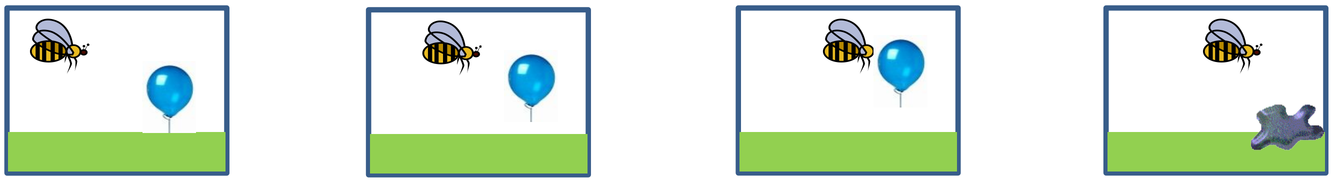
BOOTSTRAP
www.bootstrapworld.org

In Partnership with Code.org

The Algebra Inside Your Videogame: Computer Science in Math

Think of a videogame as a sequence of frames, like pages of a flip-book animation. Game elements change between frames, either on their own or in response to an event (a key-press, a mouse-click, etc). In Bootstrap, each student develops his or her own game around three such elements: a player (the user's avatar), a target (something the player wants) and a danger (something the player must avoid).

Let's look at a sample game in which the player (a bee), which is trying to hit the target (a balloon). Between frames the balloon's *y*-coordinate changes and the bee's *x*-coordinate changes:



Simple algebraic functions describe how the positions of the balloon and bee change between frames:

$$new\_balloonY(currY) = currY + 5 \qquad new\_beeX(currX, key) = \begin{cases} currX + 2, key = "right" \\ currX - 2, key = "left" \end{cases}$$

Students write functions like these (in a programming language) to update the positions of objects, to detect when their characters have gone off-screen (inequalities in the plane), and to detect when two objects are close enough to collide (Pythagorean theorem). Students provide these functions and image files for their characters, while our software handles the complexities of events, polling, etc. Our standard module focuses on functions, conditionals, and testing: a small set of concepts for both students and teachers. We also offer a follow-on module that covers loops, data structures, events, and other typical topics for a full introductory programming course.

Algebra goes beyond just *solving for x*, and Bootstrap goes beyond just *writing code*. We also teach students to document data types and, more importantly, how to test that their functions produce expected answers. Types and tests help students develop a program step-by-step, switching between concrete and abstract ways of thinking about functions. Better still, **each of these corresponds to a key concept about functions in the Common Core math standards**: Types teach domain and range, while test cases correspond to input/output tables. In fact, several standards emphasize teaching students how these three representations (in addition to graphs) talk about the same objects (functions).

Bootstrap naturally (and explicitly) reinforces these concepts by teaching program design, allowing students to use algebra and programming to solve problems that are concrete, relevant and engaging.

BOOTSTRAP www.bootstrapworld.org

In Partnership with Code.org

Bootstrap:1 Unit	Description	Objectives	Standards
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 tinkering with the programming language and the Circles of Evaluation.	Students will be able to identify points on the Cartesian plane, write simple arithmetic programs, and correctly solve equations using Order of Operations.	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.	Students will be able to identify different data types (Strings, Images, Numbers, etc.), and to analyze the Domain and Range of a function.	F-IF.1-3
Defining Functions	Students define values of various types, as well as linear functions. They are also introduced to the process for deriving functions from Word Problems using worked-through examples, called the *Design Recipe*.	Students will be able to model relationships in context using linear functions, and interpret a function's behavior using Domain and Range.	A-SSE.1-2 F-IF.1-3 7.EE.4.a F-BF.1-2
The Design Recipe	Students continue to practice the Design Recipe by applying it to a diverse range of Word Problems.	Students will be able to solve word problems by breaking them down into a series of structured steps.	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.	Students will be able to apply their knowledge of linear functions to a real-world problem.	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 offscreen.	Students will be able to express geometric regions as Cartesian inequalities, and interpret logical statements involving "and" / "or".	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.	Students will be able to describe the need for piecewise functions, and identify their behavior using function notation.	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.	Students will be able to demonstrate a geometric proof of the Pythagorean Theorem, and apply it to a real-world problem.	8.G.6 F-IF.1-3 F-IF.4-6
Optional Activities	*Have students use composition and coordinates to create a collage, or to build flags for their favorite country. Have them use randomness and trigonometric functions for more sophisticated motion, or introduce data structures for more sophisticated games.*	*These activities offer connections to additional Geometry, Programming and Trigonometry concepts.*	*Various*

* A ninth unit ("Preparing for Launch") is used as review. As such, it is not listed here alongside the units that *introduce* content.