UNPLUGGED

Video Games and Coordinate Planes

Lesson time: 30-60 Minutes

LESSON OVERVIEW

Students discuss the components of their favorite video games and discover that they can be reduced to a series of coordinates. They then explore coordinates in Cartesian space, identifying the coordinates for the characters in a game at various points in time. Once they are comfortable with coordinates, they brainstorm their own games and create sample coordinate lists for different points in time in their own game.

LESSON OBJECTIVES

Students will:

- Create a data model that describes a simple video game.
- Describe the movements of videogame characters by their change in coordinates.

ANCHOR STANDARD

Common Core Math Standards

- **6.NS.8**: Solve real-world and mathematical problems by graphing points in all four quadrants of the coordinate plane. Include use of coordinates and absolute value to find distances between points with the same first coordinate or the same second coordinate.

Additional standards alignment can be found at the end of this lesson

TEACHING SUMMARY

Getting Started

1) Vocabulary
2) Learning a Language

Activity: Video Games and the Coordinate Plane

3) Reverse Engineer a Demo
4) Coordinate Planes

Wrap-up

5) Brainstorming a Game

TEACHING GUIDE

MATERIALS, RESOURCES, AND PREP

For the Student

- Reverse Engineering Table (in the student workbook)
- Videogame Design Template (in the student workbook)

For the Teacher

- Lesson slide deck
- Example Game
- Printed cutouts of the Ninja, Dragon, and Unicorn

GETTING STARTED

1) Vocabulary

This lesson has three new and important words:

- **Apply** - use a given function on some inputs
- **Reverse Engineer** - to extract knowledge or design information from an existing product
- **Sprite** - a graphic character on the screen. Sometimes called a bitmap or an image.

2) Learning a Language

Welcome to Code.org CS in Algebra! In this course you'll be learning a new programming language - a way to tell computers exactly what you want them to do. Just like English, Spanish or French, a programming language has its own vocabulary and grammar that you'll have to learn. Fortunately, the language you'll be using here has a lot in common with the simple math that you already know!

Connect this material with things students already know:

- What makes a language?
- Does anyone speak a second (or third) language? Do you speak a different language than your parents/grandparents?
- Are there languages that share features, such as a common root (Romance, Germanic) or a similar alphabet (Latin, Cyrillic, Arabic, Kanji)?
- Are there languages that are designed for specific purposes or within certain constraints (sign language, Esperanto)?
- Math is a language, just like English, Spanish, or any other language!
  - We use nouns, like "bread", "tomato", "mustard" and "cheese" to describe physical objects. Math has values, like the numbers 1, 2 or 3, to describe quantities.
  - We also use verbs like "toast", "slice", "spread" and "melt" to describe operations on these nouns. Mathematics has functions like addition and subtraction, which are operations performed on numbers.
  - Just as you can "slice piece of bread", a person can also "add four and five".

A mathematical expression is like a sentence: it's an instruction for doing something. The expression 4+5 tells us to add 4 and 5. To evaluate an expression, we follow the instructions in the expression. The expression 4+5 evaluates to 9.

ACTIVITIES:

3) Reverse Engineer a Demo

Let's begin by exploring a simple video game, and then figuring out how it works. Open this link to play the game, and spend a minute or two exploring it. You can use the arrow keys to move the up and down - try to catch the unicorn and avoid the dragon!

This game is made up of characters, each of which has its own behavior. The unicorn moves from the left to the right, while the dragon moves in the opposite direction. The ninja only moves when you hit the arrow keys, and can move up and down. We can figure out how the game works by first understanding how each character works.

Directions:

1) Divide students into groups of 2-4.

2) Provide each student with a copy of the reverse-engineering table.

3) As students demo the game, ask them to fill in the "Thing in the game..." column with every object they see in the game.

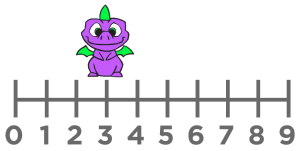
4) Discuss with the whole group which things they came up with. Characters? Background? Score?

5) Next, for each of the things in the game, fill in the column describing what changes. Size? Movement? Value?

6) Ask students to share back with the whole group. Note how students described changes - how detailed were they? What words did they use to describe movement?

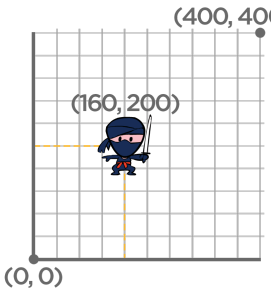
4) Coordinate Planes

Computers use numbers to represent a character's position on screen, using number lines as rulers to measure the distance from the bottom-left corner of the screen. For our video game, we will place the number line so that the screen runs from 0 (on the left) to 400 (on the right). We can take the image of the Dragon, stick it anywhere on the line, and measure the distance back to the left hand edge. Anyone else who knows about our number line will be able to duplicate the exact position of the Dragon, knowing only the number. What is the coordinate of the Dragon on the right hand side of the screen? The center? What coordinate would place the Dragon beyond the left hand edge of the screen?



LESSON TIP

The key point for students here is precision and objectivity. There are many possible correct answers, but students should understand why any solution should be accurate and unambiguous. This requires students to propose solutions that share a common "zero" (the starting point of their number line) and direction (literally, the direction from which a character's position is measured).



By adding a second number line, we can locate a character anywhere on the screen in either dimension. The first line is called the x-axis, which runs from left to right. The second line, which runs up and down, is called the y-axis. A 2-dimensional coordinate consists of both the x- and y-locations on the axes. Suppose we wanted to locate the Ninja's position on the screen. We can find the x-coordinate by dropping a line down from the Ninja and read the position on the number line. The y-coordinate is found by running a line to the y-axis.

A coordinate represents a single point, and an image is (by definition) many points. Some students will ask whether a character's coordinate refers to the center of the image, or one of the corners. In this particular program, the center serves as the coordinate - but other programs may use another location. The important point in discussion with students is that there is flexibility here, as long as the convention is used consistently.

When we write down these coordinates, we always put the x before the y (just like in the alphabet!). Most of the time, you'll see coordinates written like this: (200, 50) meaning that the x-coordinate is 200 and the y-coordinate is 50.

Depending on how a character moves, their position might change only along the x-axis, only along the y-axis, or both. Look back to the table you made. Can the Ninja move up and down in the game? Can he move left and right? So what's changing: his x-coordinate, his y-coordinate, or both? What about the clouds? Do they move up and down? Left and right? Both?

OPTIONAL ACTIVITY: Depending on timing and the background of your students, having one student place a character on a large graph and another student stating the coordinates is excellent practice. Students often need extra practice remembering which

coordinate comes first. Coordinates do not have to be exact but they should be in the correct order. Extending this to all four quadrants to include negative numbers is also excellent practice.

Fill in the rest of the reverse-engineering table, identifying what is changing for each of your characters.

WRAP-UP

5) Brainstorming for a Game

Use the game planning template to make your own game. Just like we made a list of everything in the Ninja game, we're going to start with a list of everything in your game. To start, your game will have four things in it:

- A Background, such as a forest, a city, space, etc.
- A Player, who can move when the user hits a key.
- A Target, which flies from the right to the left, and gives the player points for hitting it.
- A Danger, which flies from the right to the left, which the player must avoid.

LESSON TIP

The structure of your students' games will very closely resemble the demo they've just played. Many students will want to reach for the stars and design the next Halo. Remind them that major games like that take massive teams many years to build. Some of the most fun and enduring games are built on very simple mechanics (think Pacman, Tetris, or even Flappy Bird).