Collision Detection - Distance and Inequality

(Also available in Pyret)

Students use function composition and the distance formula to detect when characters in their games collide.

Lesson Goals	Students will be able to: • recognize another application for simple inequalities • compose their `distance` function with an inequality to determine when their game characters collide
Student-Facing Lesson Goals	• Let's write a function that recognizes when our characters collide!
Prerequisites	• Simple Data Types • Contracts • Functions: Contracts, Examples & Definitions • Solving Word Problems with the Design Recipe • Simple Inequalities • Problem Decomposition • Piecewise Functions and Conditionals
Materials	• PDF of all Handouts and Page • Lesson Slides • Printable Lesson Plan (a PDF of this web page)
Supplemental Materials	• *Flag of Trinidad and Tobago Starter File*

Problem Decomposition Returns! *20 minutes*

Overview

Students revisit problem decomposition - the idea of breaking down a complex problem into simpler pieces, solving those pieces separately, and then composing the solutions to solve the original.

Launch

Knowing how far apart our characters are is the first step. But in order to keep track of the score, we need to know when they are close enough to collide!

"Problem Decomposition" is a powerful tool, which lets us break apart complex problems into simpler ones that we can solve, test, and then glue together into a complex solution.

Students may remember that there are two strategies for doing this:

1. **Top-Down:** Describe the problem at a high level, then fill in the details later
2. **Bottom-Up:** Focus on the smaller parts that you’re sure of, then build them together to get the big picture

Investigate

For the following complex word problem, have students **first** decide which strategy they want to use, and then apply the Design Recipe to build the functions they need.

- Before we focus on our game code, let’s practice breaking apart a different complex word problem.
- A retractable flag pole starts out 24 inches tall, and can grow at a rate of 0.6in/sec. An elastic is tied to the top of the pole and anchored 200 inches from the base, forming a right triangle. Write a function that computes the area of the triangle as a function of time.
- This is easier to think about as two functions:
    - one that computes the height of the pole, based on the seconds
    - another that computes the area of the triangle, based on the height
- Does one function depend on (or "sit on top of") the other? If so, which one?
    - *Yes -* `area` *depends on* `height`.
- Which strategy will YOU use: bottom-up (independent first) or top-down (dependent first)?

    - *Students answers will vary! They can define either function first.*
  - Complete Top Down / Bottom Up, using your chosen strategy for Problem Decomposition!

Synthesize

Note: Defining the `height` first is bottom-up, and solving `area` first is top-down.

- Which strategy did students use?
- Did they try starting with one function, and then switch to another?
  - *Invite students to share. Oftentimes, responses are not only intriguing but can highlight the value of each approach. Explicitly point out that the* `area` *function _uses* `height`, allowing us to break a big problem down into two smaller ones._

Collision Detection *20 minutes*

Overview

Students once again see function composition at work, as they compose a simple inequality with the `distance` function they've created.

Launch

Knowing how far apart our characters are is the first step. We still need the computer to be asking: "True or False: is there a collision?"

Investigate

- This is easier to think about as two functions:
  - one that computes the distance between a Player and a Character, based on their coordinates
  - another that checks if those same coordinates are less than 50 pixels apart, based on the distance
- Does one function rely on the other? If so, which one?
- Complete Word Problem: collision?.
- When you've finished, open your saved game file and fix the `collision?` function in your game file, and click "Run"!

Synthesize

- In our flag-pole exercise, we had to write two functions and could start with one or the other. In our game, however, we began already having written the `distance` function! Is this **Top-Down** or **Bottom-Up** decomposition?
- Explicitly point out that the `collision?` function *uses* `height`, allowing us to break a big problem down into two smaller ones.
- Connect this back to `profit` (from Problem Decomposition), which relied on `revenue` and `cost`. Which function(s) would a top-down strategy address first?
- Connect this back to `onscreen?` (from Sam the Butterfly), which relied on `safe-left?` and `safe-right?`. Which function(s) would a bottom-up strategy address first?

Additional Exercises

For teachers who've already introduced your class to flags, Flag of Trinidad and Tobago Starter File makes use of Pythagorean Theorem and could make for an interesting connection to this lesson.