Player Animation

(Also available for WeScheme)

Students apply their knowledge of piecewise functions to write a function to move the player in their game.

Prerequisites	Piecewise Functions
Relevant Standards OK CC-Math	*Select one or more standards from the menu on the left (⌘-click on Mac, Ctrl-click elsewhere).*
Lesson Goals	Students will be able to: • Apply previous knowledge of *piecewise function*s to a new problem situation.
Student-Facing Lesson Goals	• I can write a function using conditionals to move my player.
Materials	• Lesson slides (Google Slides) • Word Problem: update-player (Page 51)
Preparation	• Make sure all materials have been gathered • Decide how students will be grouped in pairs
Supplemental Resources	• Domain & Range of Piecewise Functions (Desmos Activity)
Key Points for the Facilitator	• Encourage students to challenge themselves when creating update-player by completing one of the extension activities. • The update-player function is one of the main places where students can set their game apart and make it theirs. Encourage exploration and experimentation! • Adding comments to code - if you have to ask a student "What are you trying to do there?", then they probably need more comments!

For a textbook-like version of materials similar to these, you may wish to see the prior unit-based version

Glossary

contract :: a statement of the name, domain, and range of a function

debug :: to find and fix errors in one's code

function :: a mathematical object that consumes inputs and produces an output

piecewise function :: a function that computes different expressions based on its input

---

Warmup

Students should have their computer, workbook, contracts page, and pencil and be logged in to code.pyret.org and have their workbooks with a pen or pencil.

Defining Piecewise Functions   30 minutes

Overview

Students *define* a piecewise function. This is a challenging task, which is motivated by introducing key events in their video game.

Launch

You've already defined functions to move your `DANGER` and `TARGET`. Take a moment to look at your code or workbook, and refresh your memory on how they work.

- What controlled the speed of your characters?
- What controlled the *direction* of your characters?

If we wanted our `PLAYER` to go up all the time, we would already know how to do that. If we wanted our `PLAYER` to go *down* all the time, we would already know how to do that. But we want the player to go up *only* when the "up" arrow is pressed, and down when the "down" arrow is pressed. Do we know how to make a function behave differently, based on its input?

Investigate

Students open their **Game Project file** and look for `update-player`, then figure out what the contract represents.

> ### Strategies for English Language Learners
>
> MLR 6 - Three Reads: Have students read through the problem statement three times, looking for different information. What is the problem asking me? What is the *contract* for this *function*? What information do I need to create that function?

- **What is the contract for `update-player`?**

   The Name is `update-player`, the Domain consists of a Number and String, and Range is a Number.

- **What does each part of the domain and range represent?**

   Domain: the Number is the y-coordinate of `PLAYER`, the String is the key that the user pressed; Range: the Number is the new y-coordinate of `PLAYER`

- **What should happen mathematically to the y-coordinate of `PLAYER` when the user presses the "up" key?**

   It should increase, the program should add something to it
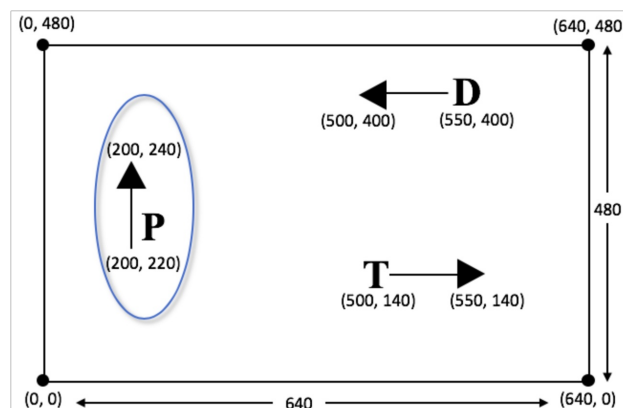
Students complete Word Problem: update-player (Page 51) with a partner, then type their code into their **Game Project file** and test.

Possible Misconceptions

- Students often think of this function as returning a *relative distance* (e.g. "it adds 20"), instead of an absolute coordinate (e.g. "the new y-coordinate is the old y plus 20")

Synthesize

Synthesize

- How is this function similar to the piecewise functions you've seen before? How is it different?

- How could we change this function so that the "W" key makes the player go up, instead of the arrow key?

- How could we change this function so that the "W" key makes the player go up, *in addition to* the arrow key?

- What happens if your little brother or sister walks by and hits a random key that doesn't have a meaning in your function? What *should* happen?

Cheat Codes and Customizations   flexible

Overview
Students choose one or more features to make their game more unique. These features can be quite simple, such as adding another key that does the same thing that "up" or "down" does. But they can also be extremely sophisticated, requiring students to exploit properties of the number line in conjuntion with function composition and compound inequalities!

Launch
Right now, all of your games allow the player to move up and down at a constant speed. But what if we wanted to add a special key that made the player warp to the top of the screen, or move down twice as fast? What if we wanted the player to *wrap*, so going off one side of the screen would make it re-appear on the other?

Investigate
Now is your time to customize your game! Try implementing some of the following features, or make your own!

- Warping - program one key to "warp" the player to a set location, such as the center of the screen
- Boundaries - change `update-player` such that `PLAYER` cannot move off the top or bottom of the screen
- Wrapping - add code to `update-player` such that when `PLAYER` moves to the top of the screen, it reappears at the bottom, and vice versa
- Hiding - add a key that will make `PLAYER` seem to disappear, and reappear when the same key is pressed again

Reminder: Use `#` to add comments to code!

Adding useful comments to code is an important part of programming. It lets us leave messages for other programmers, leave notes for ourselves, or "turn off" pieces of code that we don't want or need to *debug* later.

Have students complete at least one of the Challenges for update-player (Page 52) before turning to their computers.

Synthesize
Have students share back what they implemented. Sharing solutions is encouraged!

Question: What would it take to make the player move left and right? Why can't we do this without changing the contract?

> ### Pedagogy Note
>
> It's likely that once they hear other students' ideas, they will want more time to try them out. If time allows, give students additional *slices* of "hacking time", bringing them back to share each other's ideas and solutions before sending them off to program some more. This dramatically ramps up the creativity and engagement in the classroom, giving better results than having one long stretch of programming time.