Adding Levels

Students parameterize other parts of their game, so that the experience changes as the score increases. This track delves deeper into conditionals and abstraction, offering students a chance to customize their games further while applying those concepts.

Product Outcomes	• Students add a second level (with a different background image) to NinjaCat • test
Materials	• Slides are not yet available for this lesson • Printable Lesson Plan (a PDF of this web page)
Prerequisites	• Simple Data Types • Key Events • Contracts • Simple Inequalities • Compound Inequalities: Solutions & Non-Solutions • Piecewise Functions and Conditionals • Introduction to Data Structures • Structures, Reactors, and Animations • Build Your Own Animation

Glossary

helper function :: a small function that handles a specific part of another computation, and gets called from other functions

Adding Levels *45 minutes*

Overview

This activity introduces a programming pattern to add *levels* to students' games. For now, the only thing a level does is change the background - but students can easily extend this to change other aspects of the game.

Launch

You can add depth to your game by adding *levels*. In this lesson, we'll cover making new levels based on the game's score. To start, we want our Ninja Cat game to have a different background image when the player progresses to the next level. We'll say that **the player reaches level two when his or her score is greater than 250**.

Where do you define the `BACKGROUND-IMG` image in your game? Keep your original background for the first level, but define a new variable, `BACKGROUND2-IMG`, that will be used for level 2. For the best results, use an image that is the same size as your original background.

Once you have your second background image, it should be drawn into the game — but only when a certain condition is met. Think back to the ***helper function*** we wrote to change the color of the sunset animation in Unit 4], and we need to do the same thing here!

- What must be true for the player to progress to level 2?
- Write a function draw-bg, which consumes the score and produces the appropriate background image.

Now that we have our helper function, we can use it to draw of that one part of the animation. Instead of blindly putting `BACKGROUND-IMG` into our function, now we'll use `draw-bg(g.score)`:

```
fun draw-state(g):
put-image(text(
string-append("NinjaCat! Score: ", num-to-string(g.score)),
          20, "white"),
    310, 470,
    put-image(
    text("Use arrow keys to move. Jump on the dog & catch the ruby!",
          12, "white"),
      320, 450,
      put-image(PLAYER-IMG, g.playerx, g.playery,
        put-image(CLOUD-IMG, 150, 350,
          put-image(RUBY-IMG, g.targetx, g.targety,
            put-image(DOG-IMG, g.dangerx, g.dangery,
              draw-bg(g.score)))))))
...
```

Investigate

Now our Ninja Cat game has a level 2! You can add more conditions to `draw-bg` to have multiple levels. You can use this same technique in lots of ways:

- Write `draw-player` and change `draw-state` so that have the Player transform if the score is above 250.
- Change your animation functions so that your characters move faster if the score is above 250.
- Add a special key (jumping? firing? warping?) that is only unlocked if the score is above 250.