Unit 5
Building your World

Unit Overview

After thinking about their World, students practice building, drawing and animating it.

Agenda

5 min	Introduction
40 min	Drawing the World
40 min	Updating the World
5 min	Closing

Learning Objectives:

- Learn algebraic, World-style programming
- Understand the concept of "events", such as on-draw and on-tick

Product Outcomes:

- Students will define draw-world, and hook it up to an event handler
- Students will define a simple update-world function, and hook it up to on-tick

State Standards See our Common Core Standards Table provided as part of the Bootstrap curriculum.
Length: 90 minutes

Materials and Equipment:

- *Blank Game Template from source-files.zip, with student images included. Hint: name image DANGER instead of dog, etc., for ease of reference while writing draw-world*
- *Student workbooks*
- *Clear plastic sheet protectors: put pages 20 & 21 at the front of the workbook for ease of reference*
- *Design Recipe Sign*
- *The Ninja World 3 file [NW3.rkt from source-files.zip | WeScheme] preloaded on students' machines*

Preparation:

- *Language Table Posted*
- *Seating arrangements: ideally clusters of desks/tables*

Introduction (Time 5 minutes)

- Monitors on.
- *Students should open Ninja World, to see the dog and ruby flying across the screen*
- Let's look back at our Ninja world for a moment.
- How many things are in this world? 2. What are they? `dogX` and `rubyX`.
- What does `dogX` represent? `rubyX`?
- What function do we use to make a world? `make-world`
- What function updates the world? `update-world`
- What function draws it? `draw-world`
- How fast is the dog moving from left to right? How fast is the ruby moving right to left across the screen?
- Excellent! Now turn to page 21. It's in a sheet protector, so that you can use it as a reference from now on. What are the things in your world? If you can tell me, I'll ask your partner what datatype each thing is.
  Go through the pairs, to make sure that everyone can list the names and types, as well as answer some questions about accessor functions
- Go ahead and open your games. As you can see, they have your images in them, but nothing else. I want you to take two minutes to define your world struct at the top, and type in your example world. Name it START.

Drawing the World (Time 40 minutes)

- Now that we've got our world structure, we need to know how to draw it. Turn to Page 23, and fill in your `START` world at the bottom. Once you have that done, do a simple sketch of your `START` world. According to your world struct, where should everything be when the game starts? Take one minute and sketch that now.
  Be sure to count down!
- *5, 4, 3, 2, 1, pencils down.*
  The next thing we need to decide is what order our images need to go in. We know we have to stack images, so we're going to have to use `put-image`. Raise your hand if you can tell me: What is the contract for `put-image`? What does the first image represent? The numbers? The last image?
- In the chart on Page 23, figure out which image goes on top, which goes second, and so on. Make a list from top to bottom in the left column, and then write each image's coordinates in the right column. Take two minutes to do so.
- *Excellent!*
  Let's set up one more example, so that when we get into writing our function that draws the world, we're ready to go. On Page 24 there's a nearly identical page. You've already written a `START` world, which has everything where it will be when the game starts. Now do the same for a world called `NEXT`. This world is where everything is ONE SECOND after the game starts. Again, you have one minute to fill in a world struct, and sketch the NEXT world. Go!
- *As soon as you're done with that,*
  put the images in the same order (we don't want them to be switching around in the middle of the game!) and write the NEW coordinates beside them. Take another minute: go!
- Which function do we use to draw a world? It takes in a world and draws us a picture. What is the name of this function? `draw-world`. Just like `draw-auto`, and the `draw-world` for Ninja World, this function takes in a struct and produces an Image. What is the Domain of this function? The Range?
- On Page 25, there's a place for you to write the contract. Below that, we see a sort of staircase pattern using `put-image`, just like we did in Ninja World.
- First, start out on the bottom of this 'staircase' by putting one of our images onto the background. Do you remember how to use `put-image`? It needs the coordinates of where to put our image.
- If we wanted the images to be centered on the scene, what are the x- and y-coordinates I'll need? 320 and 240!
- But you probably don't want your image to be at the center of the screen. Look back at your `START` world picture a couple pages back...you made a note of which coordinates you want that image to be on, above the background!
- *Show students the 'staircase' pattern that forms when they put the coordinates and each image on their own line. For example:*
-

```
(put-image IMAGE
           320 240
           BACKGROUND)
```

- Now try adding another one of the images from your world. Remember, you're placing another image on top of the one that this staircase has already created! Keep adding to it, until you have a stack of all of the images in your game.
- *Work with small groups to complete this section. When students finish writing draw-world, have them type their NEXT*

world and draw-world into their games, in the GRAPHICS section.

- *If they type (draw-world START) into the interactions window, they can see a screenshot of their game.*

Updating the World (Time 40 minutes)

- Now scroll down to where it says "UPDATING FUNCTIONS." This code is responsible for changing the World. What function do you see here? `update-world`. What's in its Domain? World! Range? World!
- That's right - `update-world` takes a world, and then returns a new one that's been updated. Think of this function as the one that generates the next page of a flipbook.
- Look back at the difference between your `START` and `NEXT` worlds - what has changed?
- On Page 26, make a list of what changed and how it changed as a problem statement for writing `update-world`, using the design recipe on the next page.
- *Work with small groups to complete this section as needed. Upon completion, have students type update-world into their games.*

Closing (Time 5 minutes)

- *Have students show each other their their animated games!*