Unit 4
The Design Recipe

Unit Overview

Agenda
15min Introduction
25min Design Recipe Intro
20min Design Recipe Practice
20min Even More Practice
10min Closing

Students are introduced to the Design Recipe and apply it to simple problems.

Learning Objectives
Students will:
- Become familiar with the steps of the Design Recipe.
- Practice Racket syntax and the Circles of Evaluation.

Product Outcomes
- Students, in pairs, will write functions to solve simple problems by using the Design Recipe.

State Standards
See Bootstrap Standards Matrix provided as part of the Bootstrap curriculum.

Length: 90min

Materials and Equipment
- ☐ Student workbook folders - in pairs! - with names on covers.
- ☐ Pens/pencils for the students, fresh whiteboard markers for teachers
- ☐ Class posters (List of rules, basic skills, course calendar)
- ☐ Language Table (see below)

Preparation
- ☐ Write agenda on board
- ☐ Display Class posters, Language Table, Design Recipe
- ☐ "Rocket" [DrRacket file + teachpack | WeScheme] preloaded on students' machines
- ☐ Seating arrangements: ideally clusters of desks/tables
- ☐ Load pre-class assessment on student machines

Pre-Class Assessment: Review of Unit 3
- *This data helps us track student performance. All data collected is anonymous, and participation is optional.*
- If you would like us to see the assessment results for your class, contact us.

Language Table

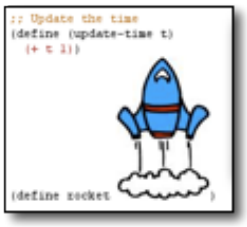
Types	Functions
Number	+ - * / sq sqrt expt
String	string-append string-length
Image	rectangle circle triangle ellipse radial-star scale rotate put-image

Introduction — Time: 15 minutes

- *Have students begin by taking the pre-class assessment on their computers.*

Introducing the Design Recipe — Time: 25 minutes

- Now it's time to define functions to make your characters move!
- You already know most of the steps from the last lesson, but we're going to do a few practice exercises before turning you loose on your own games.
- How many of you know what a flip-book animation is? Explain if necessary.
- When you make a flip-book, you draw each page a little differently from the page before it. We can use functions to do just that! Suppose we had a function that drew a rocket ship a little higher up on each page of a flip-book. What would it look like if you moved through the pages quickly? It would look like the rocket was flying! You've already learned how to draw one frame of your videogame with the characters at certain coordinates, and now we'll learn how a program can change those coordinates. **Today, you're going to learn how to animate a rocket**, then next time you will use what you learned today to animate the characters in your games!
- Believe it or not, you're 90% of the way towards being able to write an animation function! All the steps you took to write those functions before are actually part of a process that real programmers use, called the Design Recipe.
- Engineers all over the world use the Design Recipe to write functions. It's a way of thinking through each step of programming and making sure that you're on the right path before you even touch the keyboard.
- We're going to try using the Design Recipe to animate a rocket, and for this you'll be working in pairs.
- Turn to **page 10** in your workbooks. Have a student read the problem statement aloud.
- Just by reading this Problem Statement, we can tell a lot about the function we're going to write. For example, what is the function's name? Have one person in each pair underline the name on your paper.
- What does the function take in as an input? What does it give back as an output? Now the other person in each pair Circle these on your paper.
- Who would like to volunteer to *act out* this function?
- What is your **name**? `rocket-height`. What is your **Domain**? `Number`. What is your **Range**? `Number`.
- When I give you the number of seconds since blast-off, what do you do? *multiply it by seven*. Let's try it out: "rocket-height three!" "rocket-height zero!" *etc*.
- **Step 1: Contract**



- *Copy these answers on the board:*

```
;; rocket-height : Number -> Number
```

- Copy the contract down into your Design Recipe page!
- **Step 2: Give Examples**
- We already tried some examples with our volunteer. When we said "rocket-height three" (*write the accompanying code* `(EXAMPLE (rocket-height 3) ...)` *on the board*), what did we get back? 21! But *how* did we get that? By multiplying 3 and 7. Let's write that code into our example:

```
(EXAMPLE (rocket-height 3)
  (* 3 7))
```

- In your notebooks, I want you to come up with another one.
- Now we need to circle what's different between these two examples - what's changeable? What does that number represent? Is it the number of astronauts? No, it's the *time* since blastoff.

.

- **Step 3: Code**
- This next part is easy: it's exactly what you did for fast functions! Can you figure out how to plug in the header? What about the body?
- Once you are done, you and your partner can type this function into your computers.
- Do you guys want to see the rocket fly? Type `(start rocket-height)` to see your code in action. We've slowed time down, so that a second passes each time you hit the spacebar.
- All right. Now let's have a little bonus round.

- One point to every team who can figure out how to make the rocket fly twice as fast. You'll have 1 minute. GO!
- One point to every team who can figure out how to make the rocket fly backwards. You'll have 1 minute. GO!
- One point to every team who can figure out how to make the rocket fly faster over time, so it gets faster and faster as it gets higher. You'll have 1 minute. GO!

Practicing the Design Recipe  Time: 20 minutes

- *Note: if space allows, this section can also be done on the board with multiple, simultaneous challenges. See the video links for an example.*
- I need another volunteer, to act out a new function called `red-square`.
- When I say "red-square fifty", I want you to make a solid, red square where each side is 50 pixels long. "red-square fifty!" "red-square ninety!" etc..
- I want to make a function called `red-square`, which does exactly that.
- When I say "go," you will turn to **page 11** in your workbooks. You'll have 2 minutes to read the problem statement and to underline the function's name, domain and range. GO!
- *Count down the last 10 seconds, and then collect answers from teams.*
- Now you'll have another minute to fill in the contract for your function. GO!
- *Count down the last 10 seconds then collect answers from teams. Keep score!*
- **Step 1: Contract**:
- What's the name of the function we're trying to write? What kind of Domain did we say it expects? Range? Fill these in under as the three parts of your contract.
- *Count down the last 10 seconds, and then collect answers from teams.*
- Now you have to give two examples. Again, you'll have one minute. GO!
- **Step 2: Give Examples**
- When given the number 20, what should red-square draw? A solid, red square with radius 20.
  - How do I write the Racket code for that? `(rectangle 20 20 "solid" "red")`
  - Fill in the boxes with the Racket code for the other examples, including the generalized pattern.
  - Now circle everything that changes between the examples, and label it.
- *Count down the last 10 seconds then collect answers from teams. Don't forget to keep score!*
- Now for the final step! This one is worth two points: one for filling in everything on the "define" line and one for filling in the body of the function. You'll need to fill in the name of your function and a name for the variable. Then you'll need to figure out what goes on the next line for the function's body. You will have two minutes for this. GO!
- **Step 3: Code**
  - As always, we start with the function header: "define", then the name of the function and the names of the inputs from our Domain. What is the name of our function here? What is the name of our input?
  - Now you need to fill in the function body. Which pattern can we use for this? The general one.
- *Have kids turn their monitors on, and type in the code. Do their tests work?*

Even More Practice (Yard Area) Time: 20 minutes

- Okay, now it's time for the final round. When I say go, teams will turn to **page 12**, and begin. This time, each time will have 5 minutes to read the problem statement, write the contract, and fill out the examples. One point for each section. Any questions? On your mark ... get set ... GO!
- *Count down the last 10 seconds, and then collect answers from teams. Keep score!*
- Now you'll have 5 more minutes to finish the Design Recipe, and to type in your function and test cases. On your mark ... get set ... GO!
- *Count down the last 10 seconds, and then collect answers from teams. Keep score!*
- *If time allows, pass out extra Design Recipe worksheets [PDF | OOo], and run another round: e.g., one of*

```
(define (years-ago year) (- 2011 year))
(define (plural word) (string-append word "s"))
(define (diamond size color) (rotate 45 (rectangle size size "solid" color)))
```

Closing — Time: 10 minutes

- What you have learned is powerful: a structured way to approach solving any problem. First you state the problem as clearly as you can, then think about the data. Then you come up with examples of how the finished product should look, and write a template using only what you know. Only then do you solve!
- Engineers use this when they design cars. Journalists use this when they write articles. Chefs use it when they make food. Everyone, in every profession, needs to solve problems. What you have learned is a way of approaching problems that will work in many different places - not just programming. To succeed in this class, you need to be masters of problem solving. If you only learn one thing, let it be these steps.
- Next week, you will use the Design Recipe to animate the characters in your game! Don't forget the recipe - it will save your life over the next few lessons!
- Who can tell us one thing we learned today?
- Who saw someone else in the class do something great?
- Cleanup, dismissal.