Developer Tools

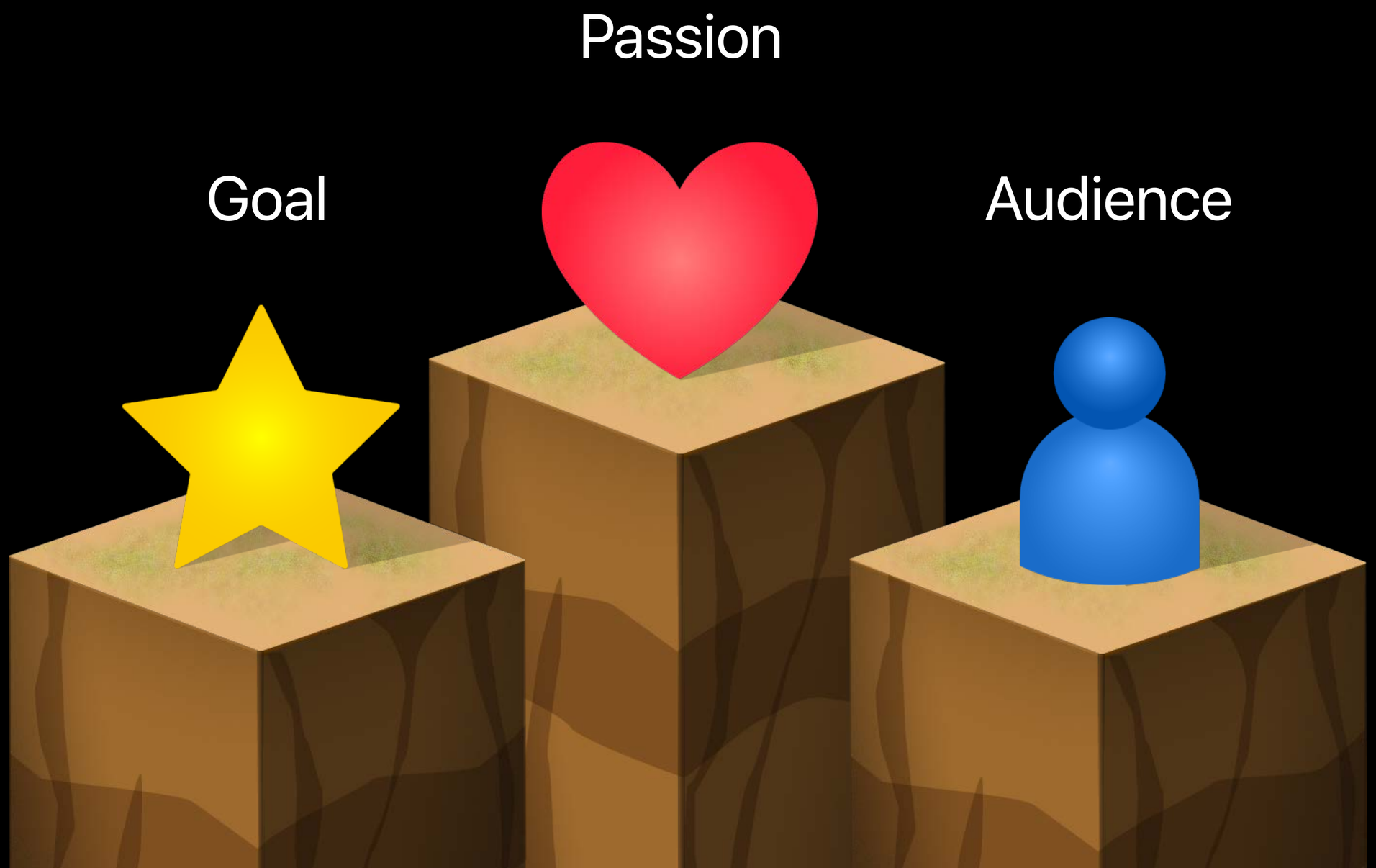
#WWDC17

Teaching with Swift Playgrounds

Session 416

Bill Dudney, Table Inverter
Elizabeth Salazar, Storyteller


CS
CS
CS
CS
CS
CS
CS
CS
CS
CS
CS
CS


Passion
Goal
Audience

Teaching

Goal

Goal

Mathematics

Mathematics

$$ax^2 + bx + c = 0$$

Polynomials

$$sin(x)$$

Trigonometry

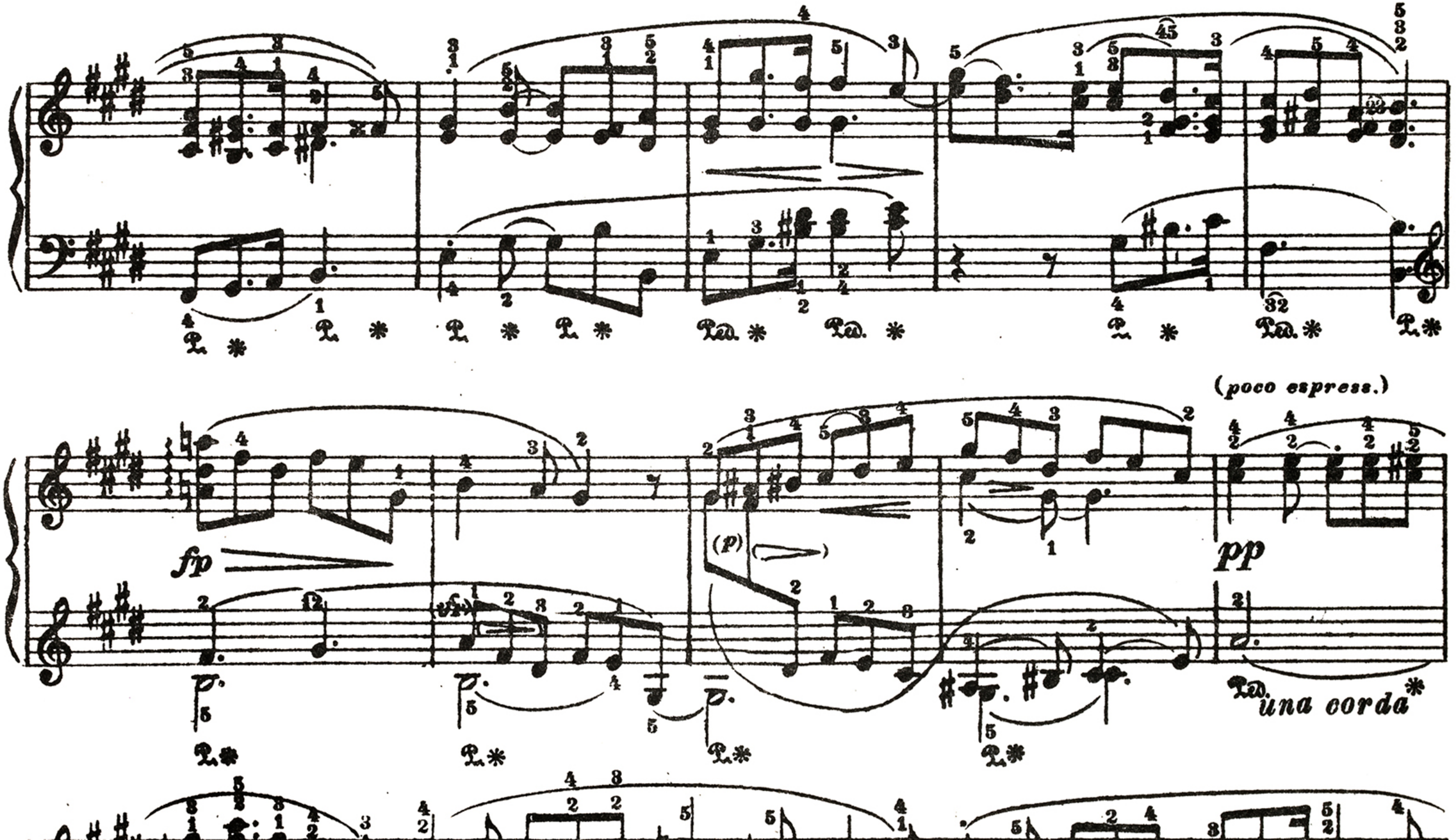
$$e^x$$

Exponentials

$$log_{10}(x)$$

Logarithms

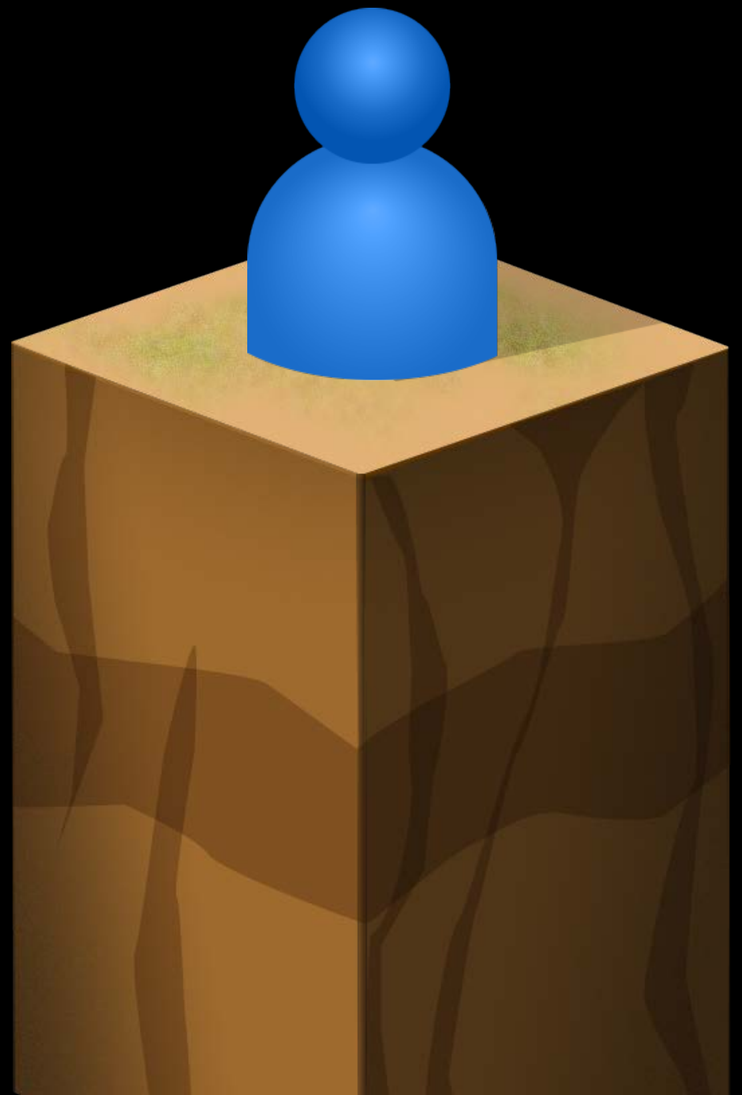
When will I use this?!


(poco espress.)
fp
(p)
pp
una corda

Start with a question.

Audience

Audience



Typical Teaching—Syntax

#function strings didSet case

operators #available false convenience

conditional class do loop declarations protocol

#selector subscript return nil continue #if else

try continue true #line as associatedtype

catch inout var willSet types static

Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a `for` loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```

Typical Teaching—
Algorithms and Data Structure

Fermat Primality Test Biconnectivity Dice’s Coefficient Shadow Volume

Interval Halving Laplacian Smoothing Simplex Noise Binary Search

Double Metaphone Soundex Heap Sort Axis Aligned Bounding Boxes

Quick Sort Gradient Descent Hashing Recursion Depth First Search

Gibbs Sampling Rasterisation Spatial Partitioning Gauss-Legendre

Gouraud Shading Hamming distance Newton’s Method

Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a `for` loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```

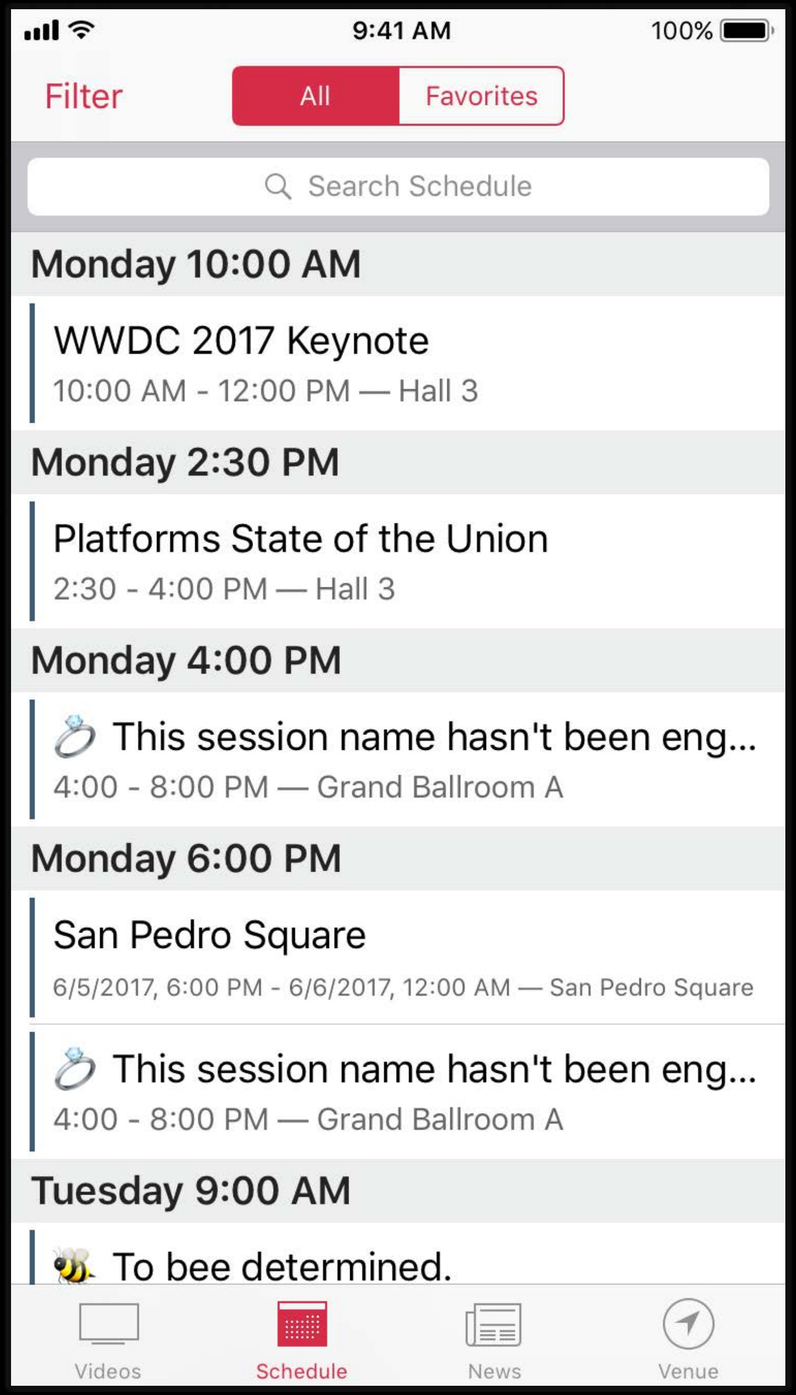
Challenge: Implement the most efficient algorithm to collect the gems and activate the switches.

For the last challenge of Learn to Code 1, you'll test your algorithm design skills. There are many different algorithms you could use to solve the puzzle, and many different ways to structure your code.

If you're not able to find a solution right away, that's okay! Coding often requires trying different solutions to a problem until you find the one that works best. When you're ready, you can move on to Learn to Code 2.

Tap to enter code

The easy parts.



Session Table View Controller

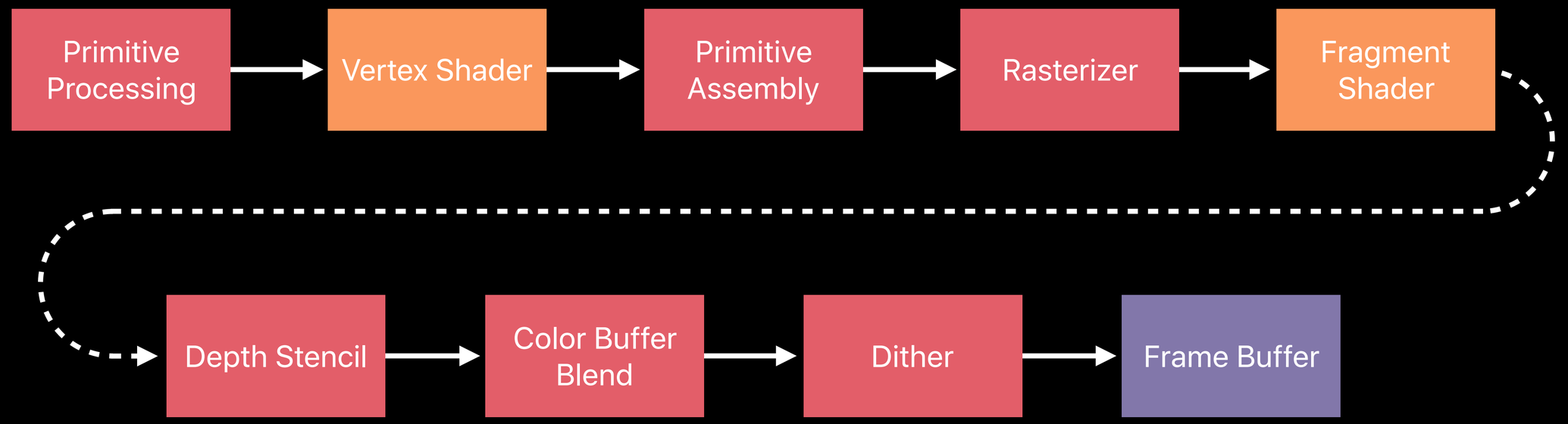
Session Table View Cell

Session

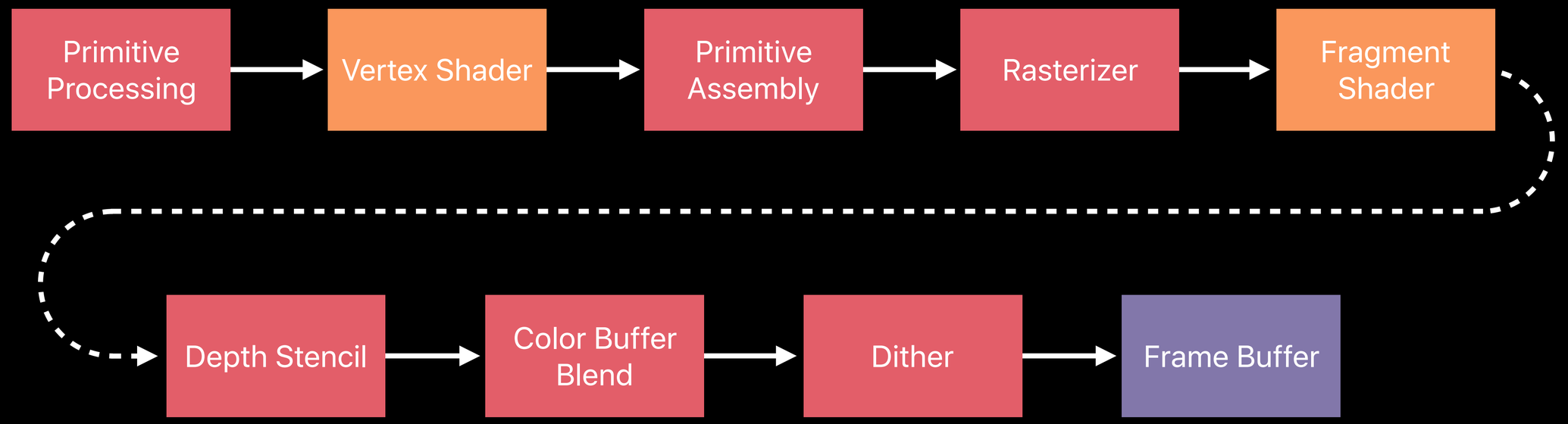
Details

Open GL Pipeline

Open GL Pipeline



Open GL Pipeline

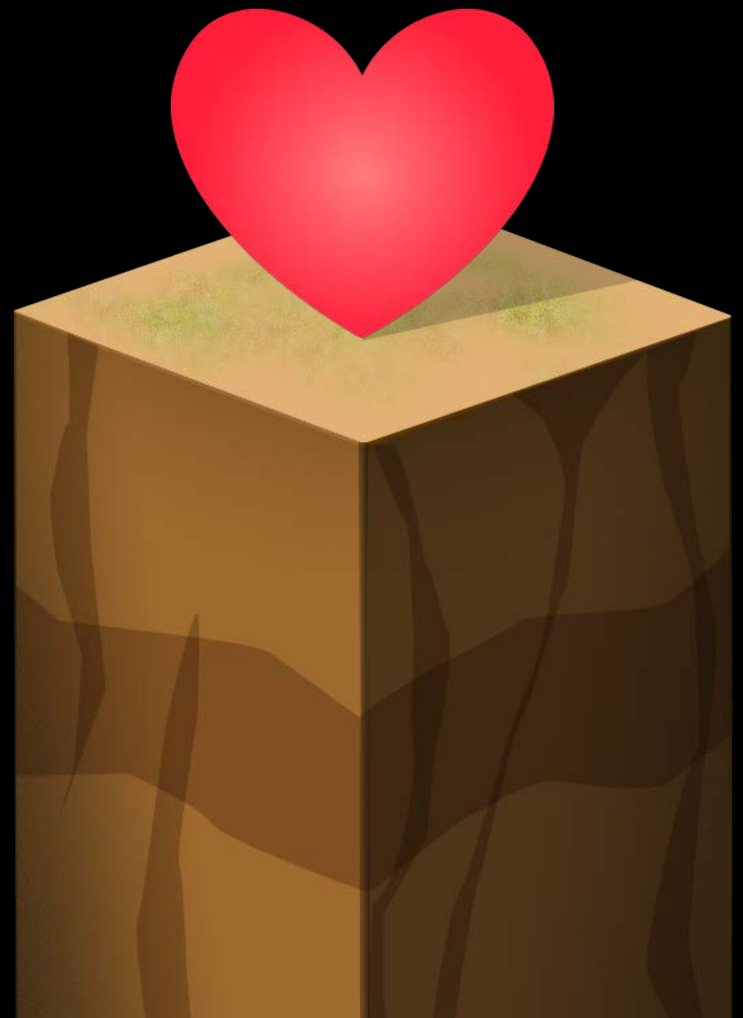


Fun

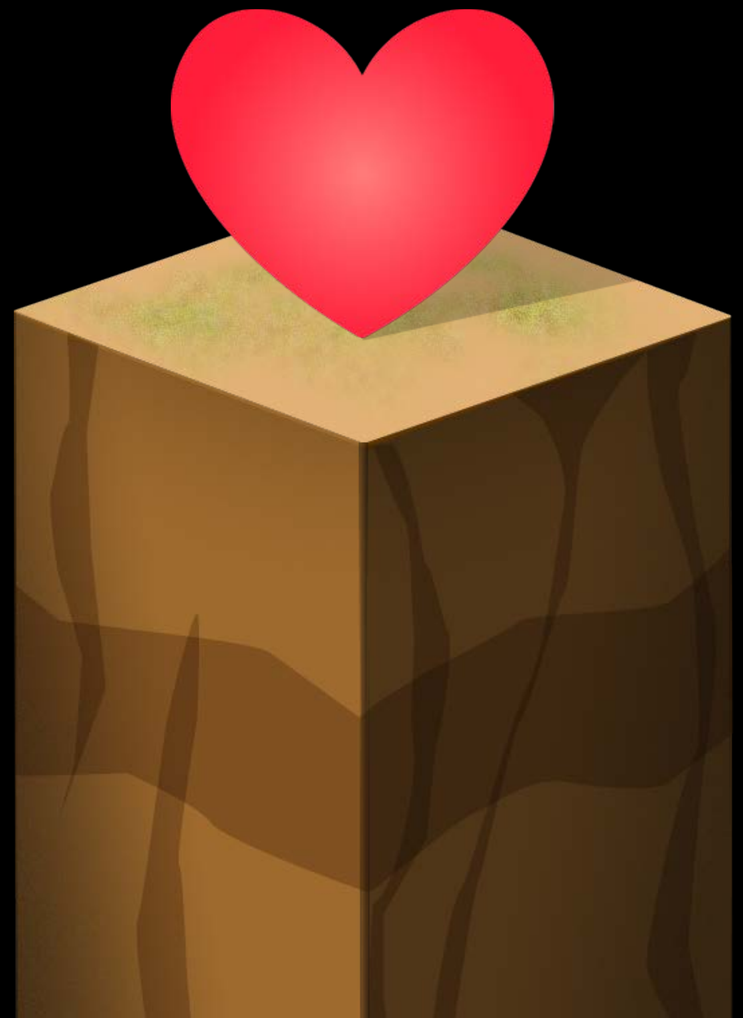
Fun

Passion

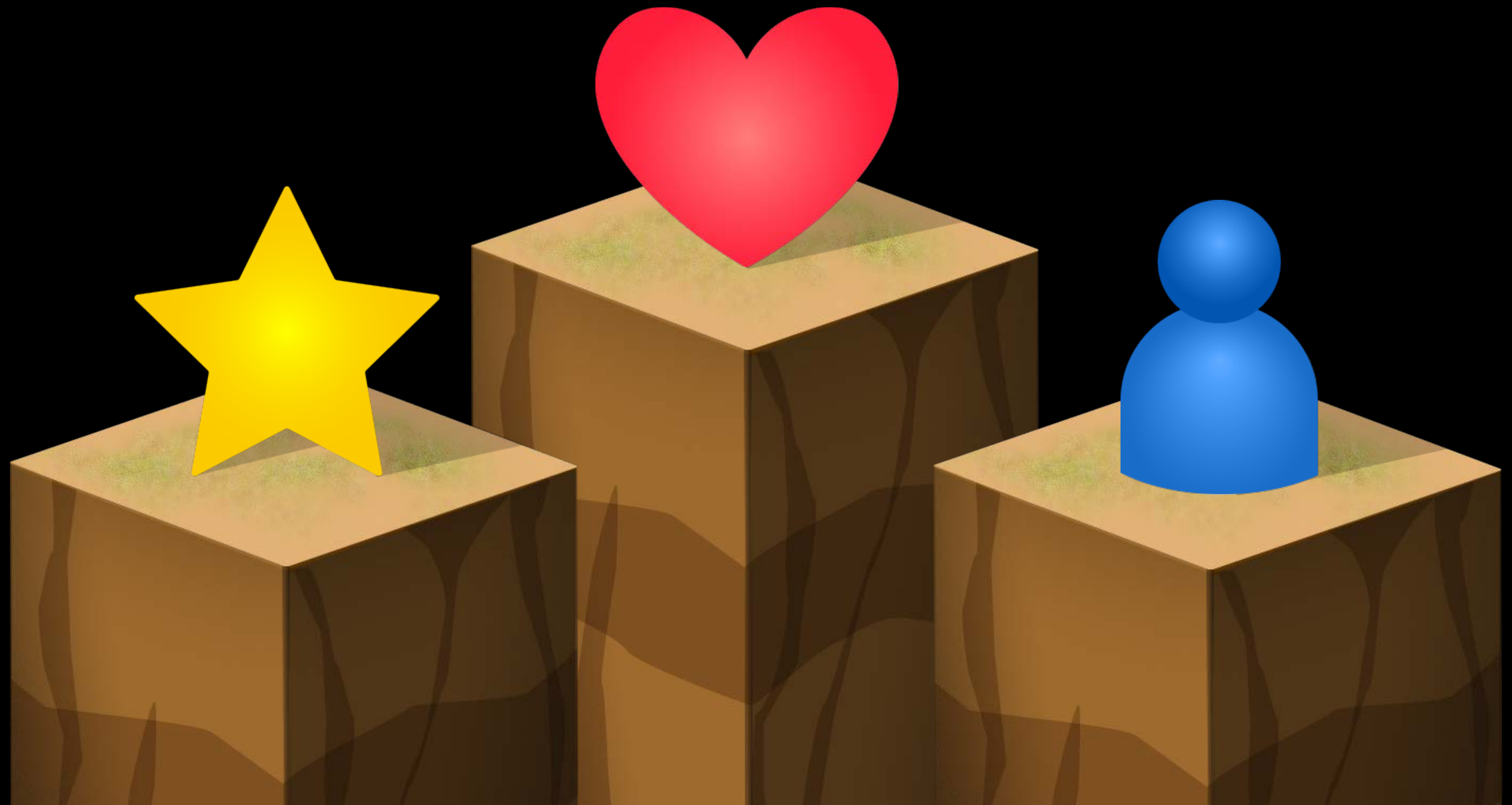
Passion



Teaching with Swift Playgrounds



Teaching with Swift Playgrounds



Teaching

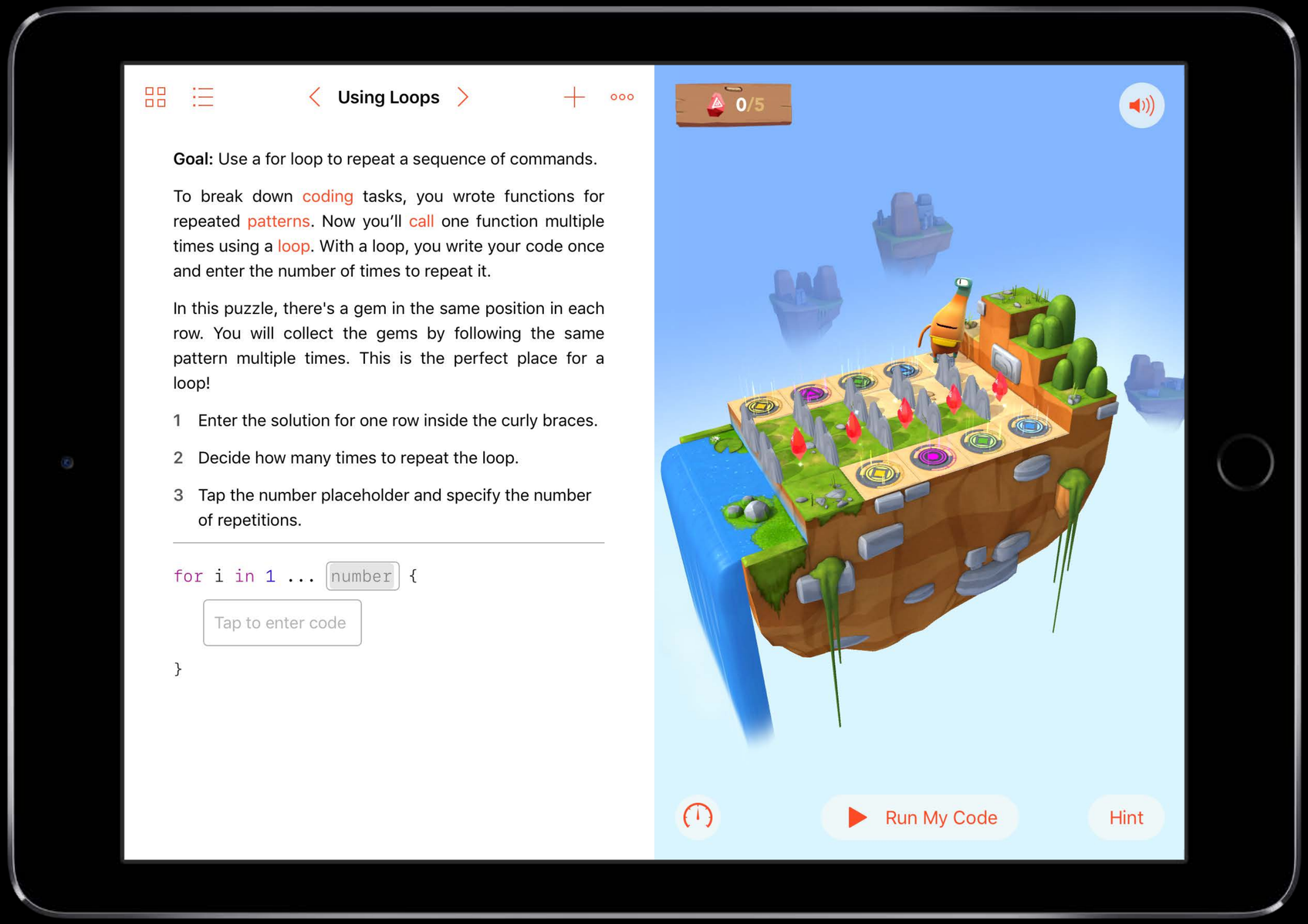
Designing

```
conditional conditional
#selector return #selector return
class #line false class #line false
operators static operators static
protocol do continue protocol do continue
default #if subscript default #if subscript
#sourceLocation self #sourceLocation self
continue #function continue #function
strings nil didSet strings nil didSet
#available try #available try
static as
types true
self else
true case
nil loop
class types
didSet public
```

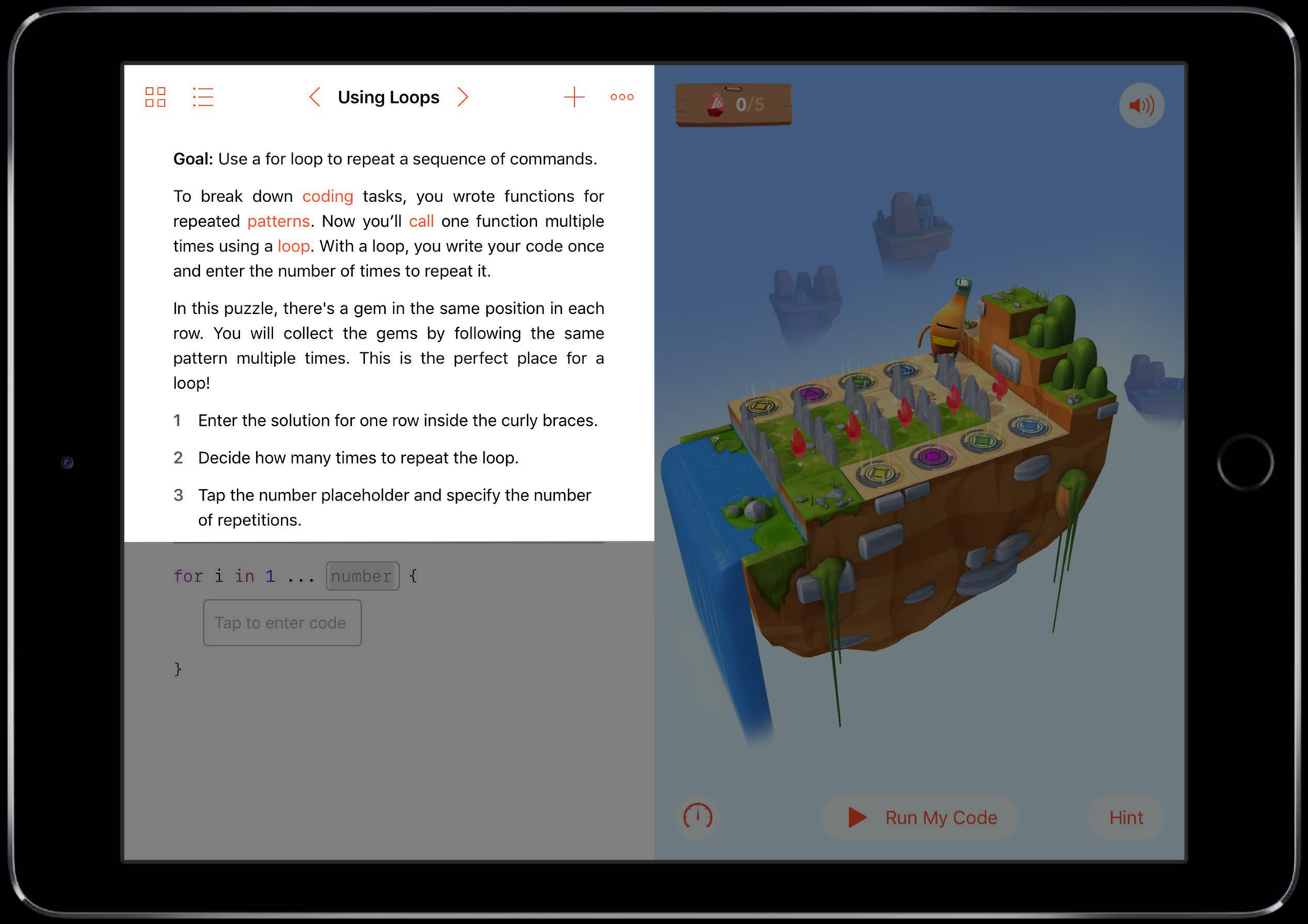
Designing Swift Playgrounds

Elizabeth Salazar, Storyteller

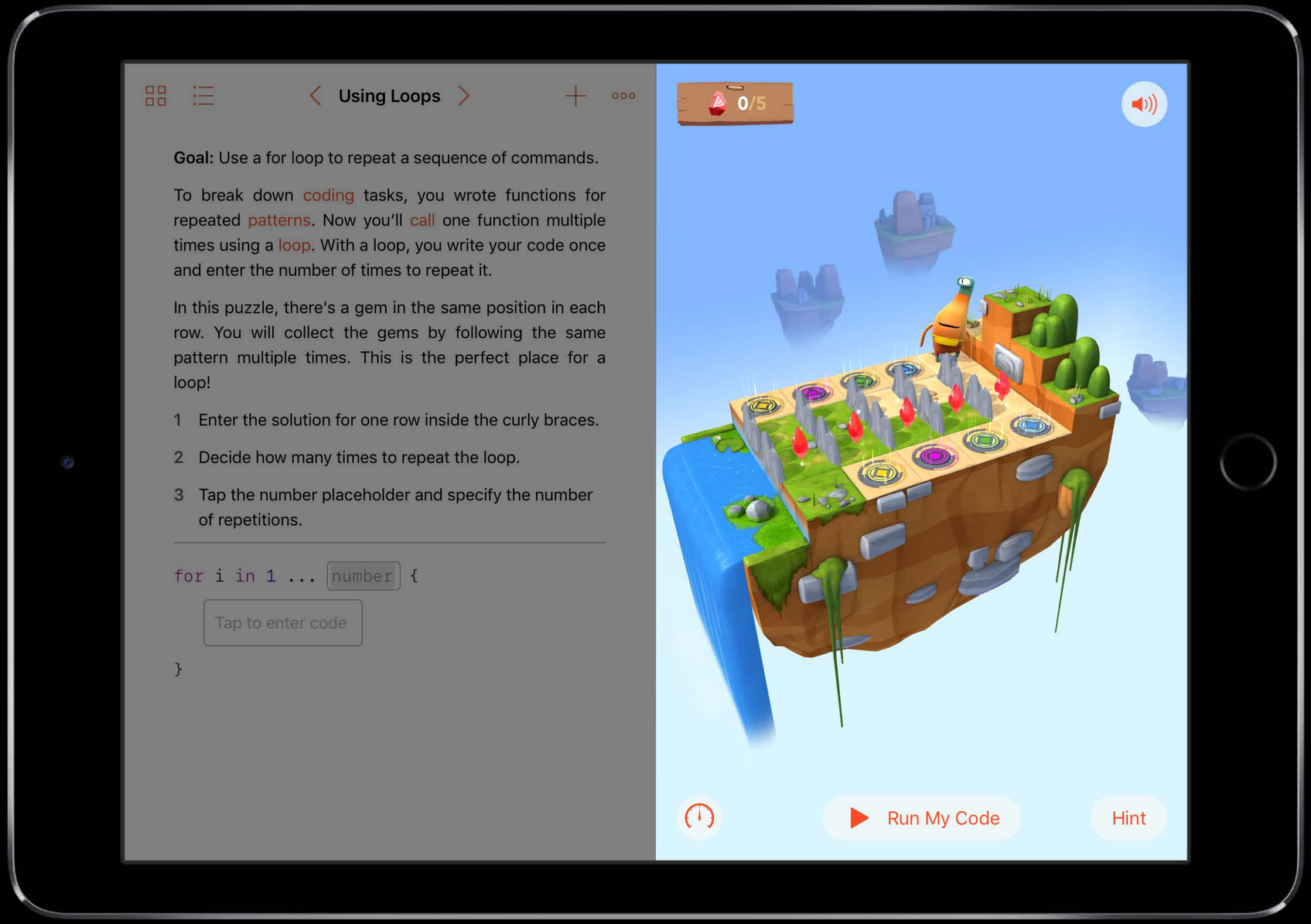
What is a Swift Playground?

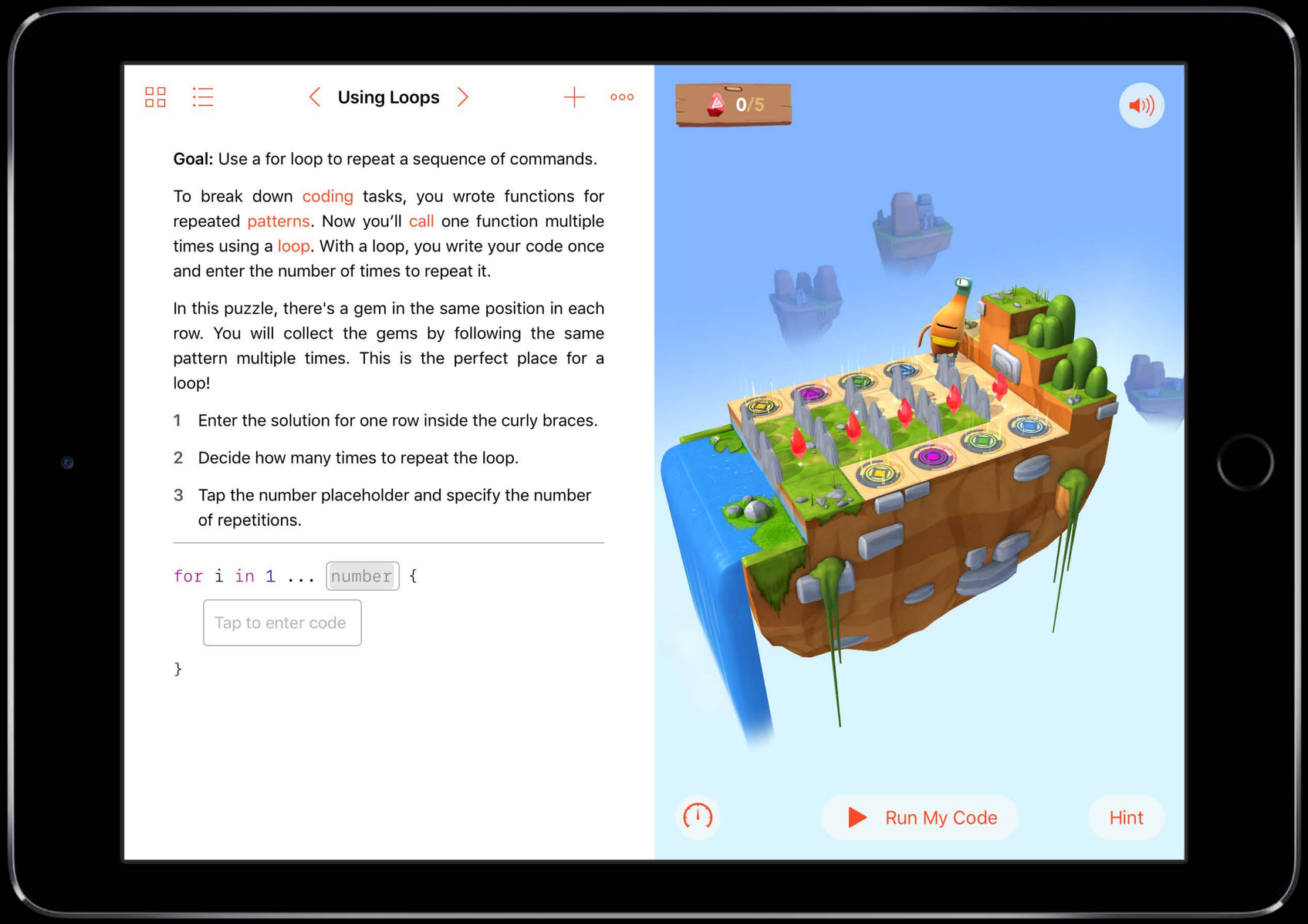

Using Loops
Goal: Use a for loop to repeat a sequence of commands.
To break down coding tasks, you wrote functions for repeated patterns. Now you'll call one function multiple times using a loop. With a loop, you write your code once and enter the number of times to repeat it.
In this puzzle, there's a gem in the same position in each row. You will collect the gems by following the same pattern multiple times. This is the perfect place for a loop!
1 Enter the solution for one row inside the curly braces.
2 Decide how many times to repeat the loop.
3 Tap the number placeholder and specify the number of repetitions.
for i in 1 ... number {
Tap to enter code
}
0/5
Run My Code
Hint


Using Loops
Goal: Use a for loop to repeat a sequence of commands.
To break down coding tasks, you wrote functions for repeated patterns. Now you'll call one function multiple times using a loop. With a loop, you write your code once and enter the number of times to repeat it.
In this puzzle, there's a gem in the same position in each row. You will collect the gems by following the same pattern multiple times. This is the perfect place for a loop!
1 Enter the solution for one row inside the curly braces.
2 Decide how many times to repeat the loop.
3 Tap the number placeholder and specify the number of repetitions.
for i in 1 ... number {
Tap to enter code
}
0/5
Run My Code
Hint

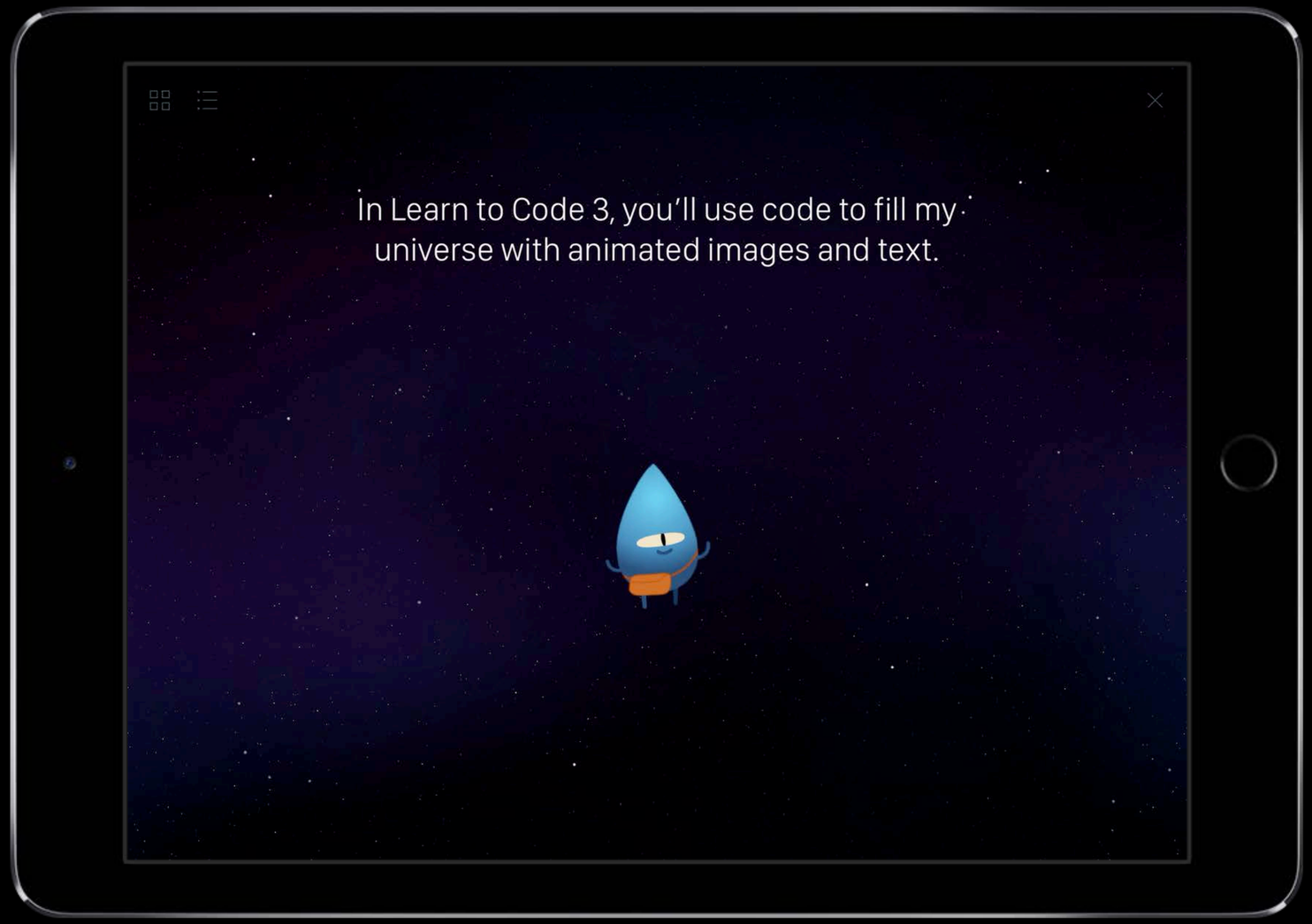

Using Loops
Goal: Use a for loop to repeat a sequence of commands.
To break down coding tasks, you wrote functions for repeated patterns. Now you'll call one function multiple times using a loop. With a loop, you write your code once and enter the number of times to repeat it.
In this puzzle, there's a gem in the same position in each row. You will collect the gems by following the same pattern multiple times. This is the perfect place for a loop!
1 Enter the solution for one row inside the curly braces.
2 Decide how many times to repeat the loop.
3 Tap the number placeholder and specify the number of repetitions.
for i in 1 ... number {
Tap to enter code
}
0/5
Run My Code
Hint


Using Loops
Goal: Use a for loop to repeat a sequence of commands.
To break down coding tasks, you wrote functions for repeated patterns. Now you'll call one function multiple times using a loop. With a loop, you write your code once and enter the number of times to repeat it.
In this puzzle, there's a gem in the same position in each row. You will collect the gems by following the same pattern multiple times. This is the perfect place for a loop!
1 Enter the solution for one row inside the curly braces.
2 Decide how many times to repeat the loop.
3 Tap the number placeholder and specify the number of repetitions.
for i in 1 ... number {
Tap to enter code
}
0/5
Run My Code
Hint

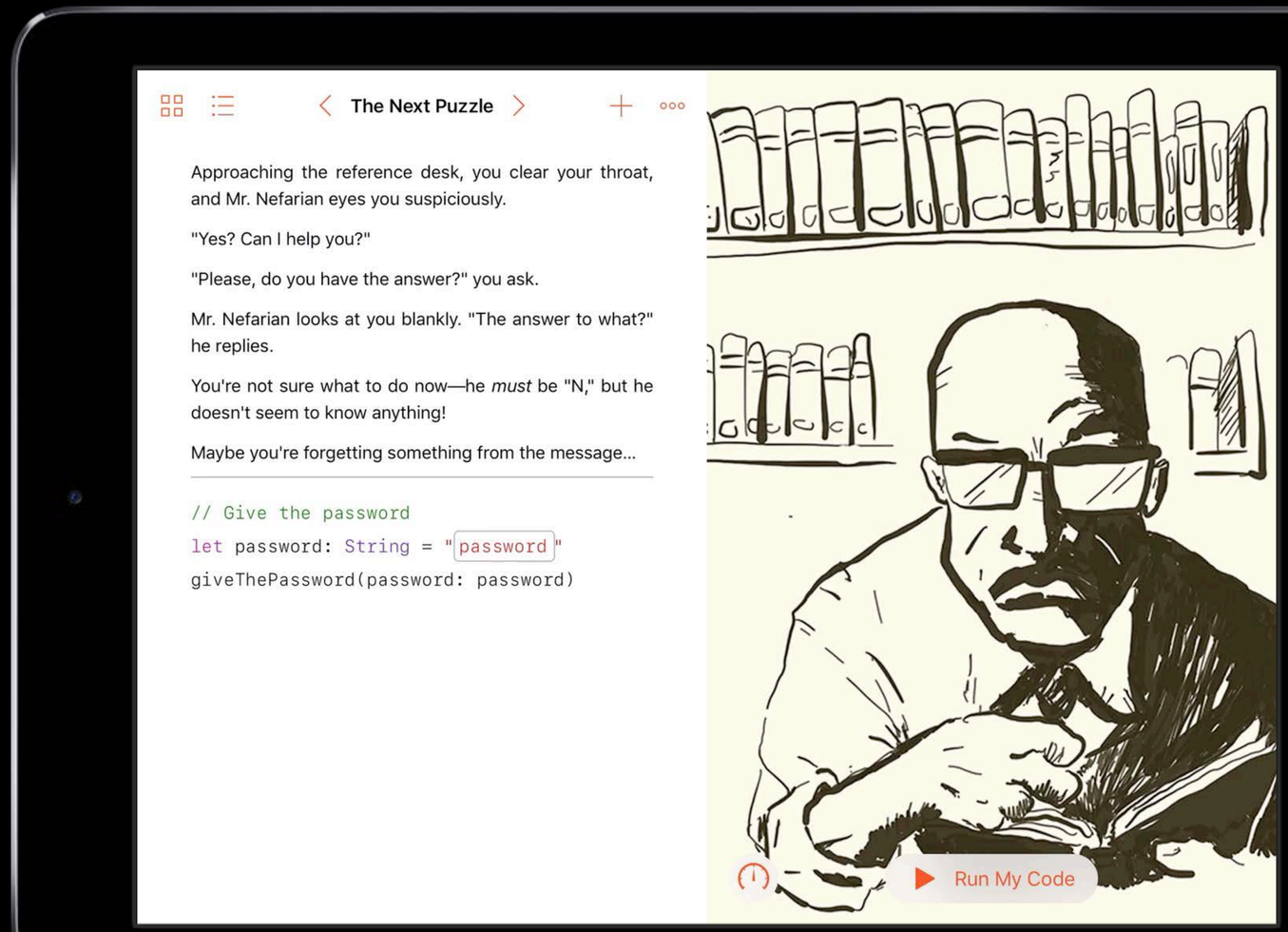

Using Loops
Goal: Use a for loop to repeat a sequence of commands.
To break down coding tasks, you wrote functions for repeated patterns. Now you'll call one function multiple times using a loop. With a loop, you write your code once and enter the number of times to repeat it.
In this puzzle, there's a gem in the same position in each row. You will collect the gems by following the same pattern multiple times. This is the perfect place for a loop!
1 Enter the solution for one row inside the curly braces.
2 Decide how many times to repeat the loop.
3 Tap the number placeholder and specify the number of repetitions.
for i in 1 ... number {
Tap to enter code
}
0/5
Run My Code
Hint


Using Loops
Goal: Use a for loop to repeat a sequence of commands.
To break down coding tasks, you wrote functions for repeated patterns. Now you'll call one function multiple times using a loop. With a loop, you write your code once and enter the number of times to repeat it.
In this puzzle, there's a gem in the same position in each row. You will collect the gems by following the same pattern multiple times. This is the perfect place for a loop!
1 Enter the solution for one row inside the curly braces.
2 Decide how many times to repeat the loop.
3 Tap the number placeholder and specify the number of repetitions.
for i in 1 ... number {
Tap to enter code
}
0/5
Run My Code
Hint

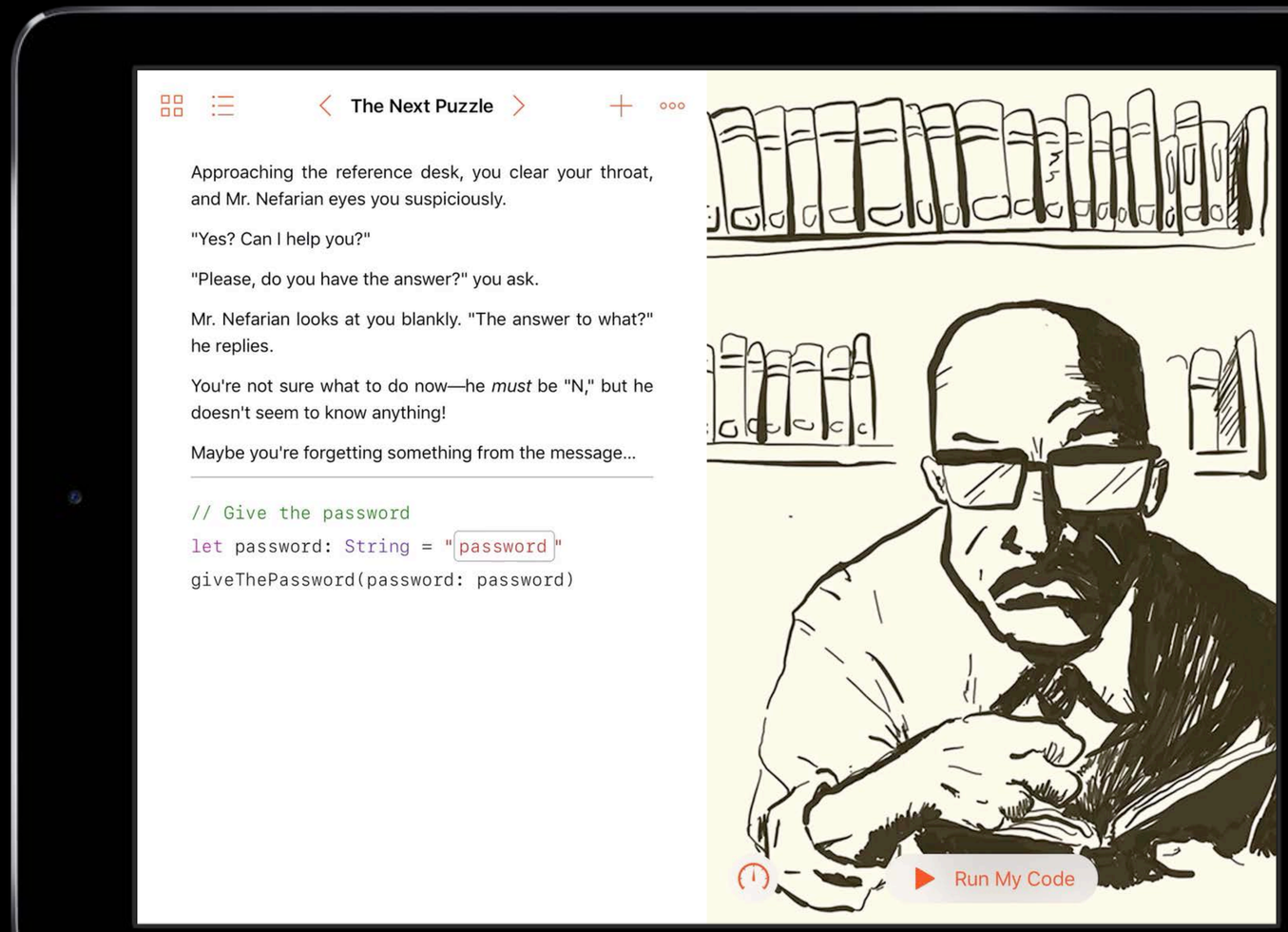
A few other features...


In Learn to Code 3, you'll use code to fill my universe with animated images and text.

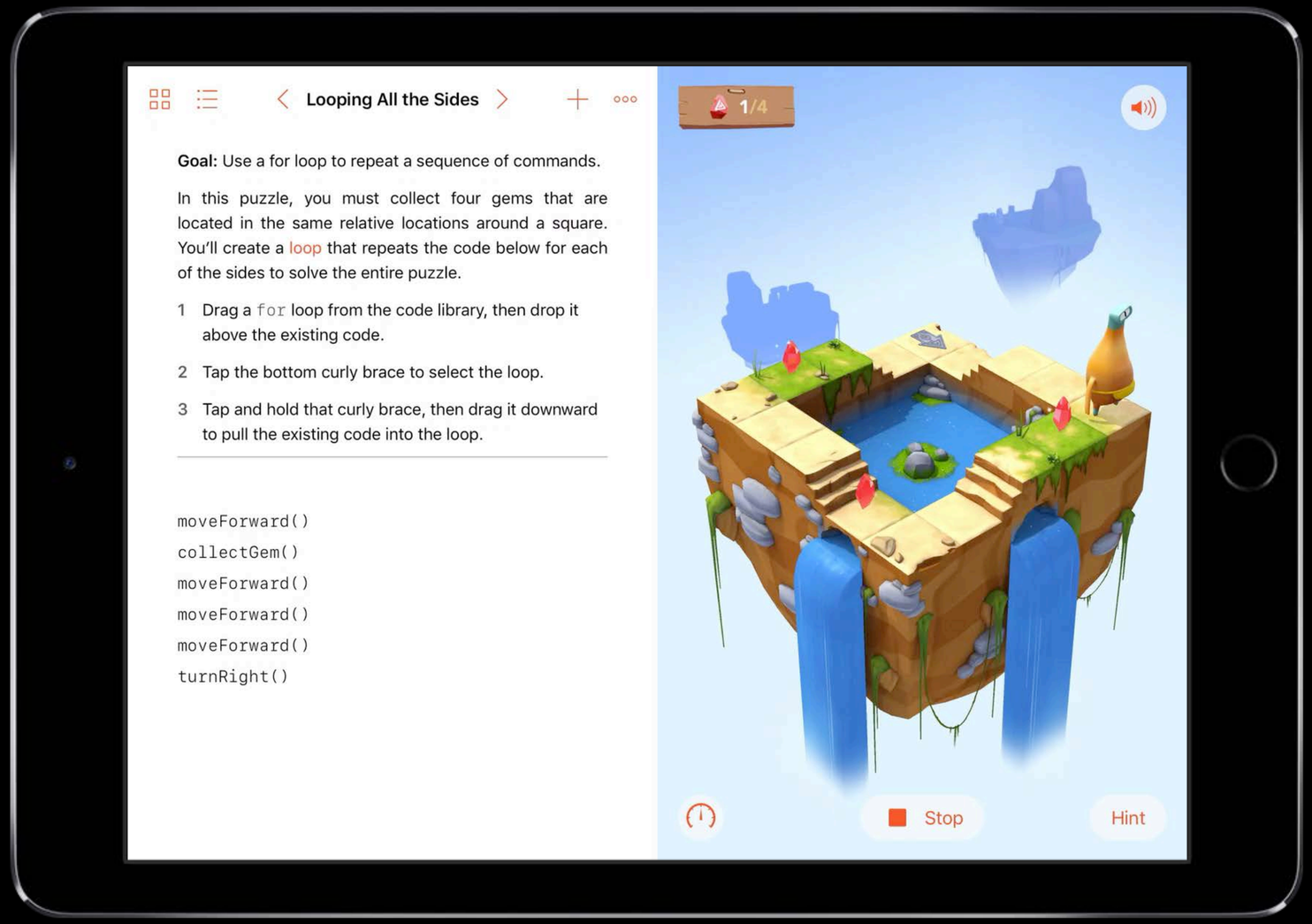
Cutscenes


The Next Puzzle
Approaching the reference desk, you clear your throat, and Mr. Nefarian eyes you suspiciously.
"Yes? Can I help you?"
"Please, do you have the answer?" you ask.
Mr. Nefarian looks at you blankly. "The answer to what?" he replies.
You're not sure what to do now—he *must* be "N," but he doesn't seem to know anything!
Maybe you're forgetting something from the message...
// Give the password
let password: String = "password"
giveThePassword(password: password)
Run My Code

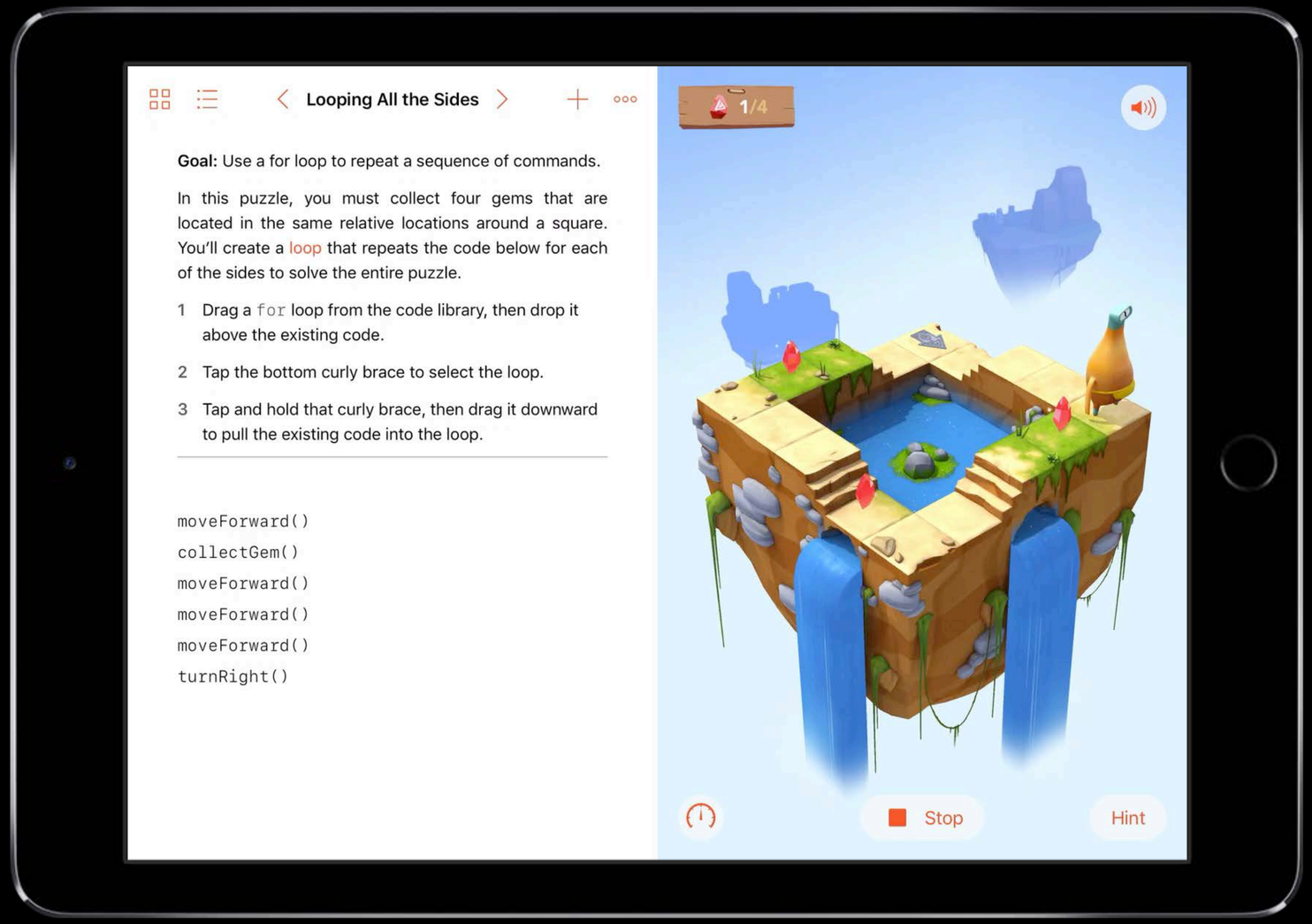
Hints


The Next Puzzle
Approaching the reference desk, you clear your throat, and Mr. Nefarian eyes you suspiciously.
"Yes? Can I help you?"
"Please, do you have the answer?" you ask.
Mr. Nefarian looks at you blankly. "The answer to what?" he replies.
You're not sure what to do now—he *must* be "N," but he doesn't seem to know anything!
Maybe you're forgetting something from the message...
// Give the password
let password: String = "password"
giveThePassword(password: password)
Run My Code

Hints


Looping All the Sides
Goal: Use a for loop to repeat a sequence of commands.
In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.
1 Drag a for loop from the code library, then drop it above the existing code.
2 Tap the bottom curly brace to select the loop.
3 Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.
moveForward()
collectGem()
moveForward()
moveForward()
moveForward()
turnRight()
1/4
Stop
Hint

Solutions


Looping All the Sides
Goal: Use a for loop to repeat a sequence of commands.
In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.
1 Drag a for loop from the code library, then drop it above the existing code.
2 Tap the bottom curly brace to select the loop.
3 Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.
moveForward()
collectGem()
moveForward()
moveForward()
moveForward()
turnRight()
1/4
Stop
Hint

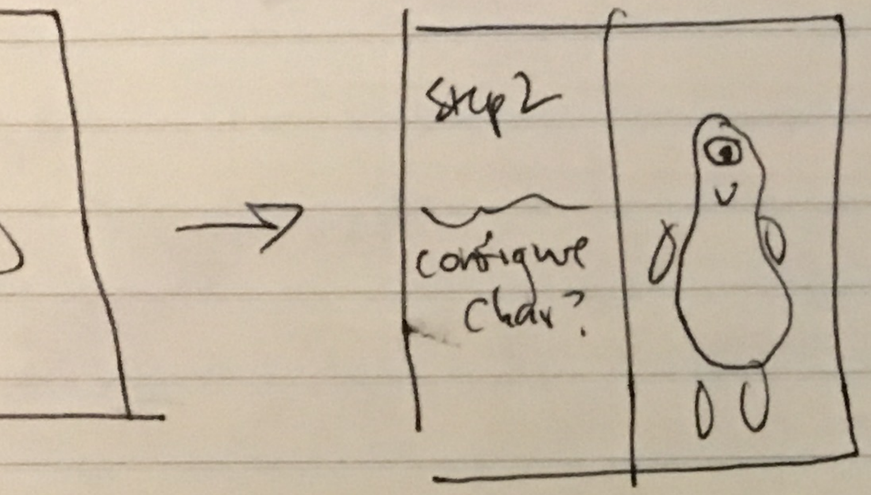
Solutions

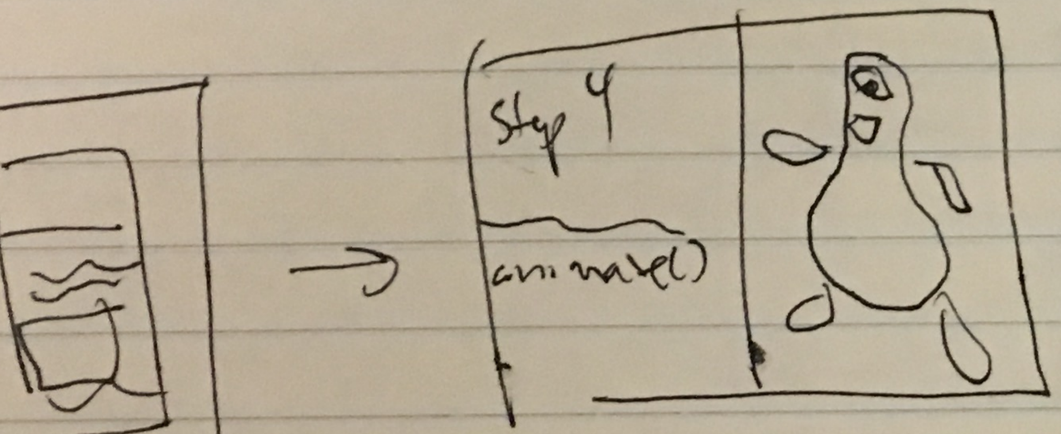
Assessments

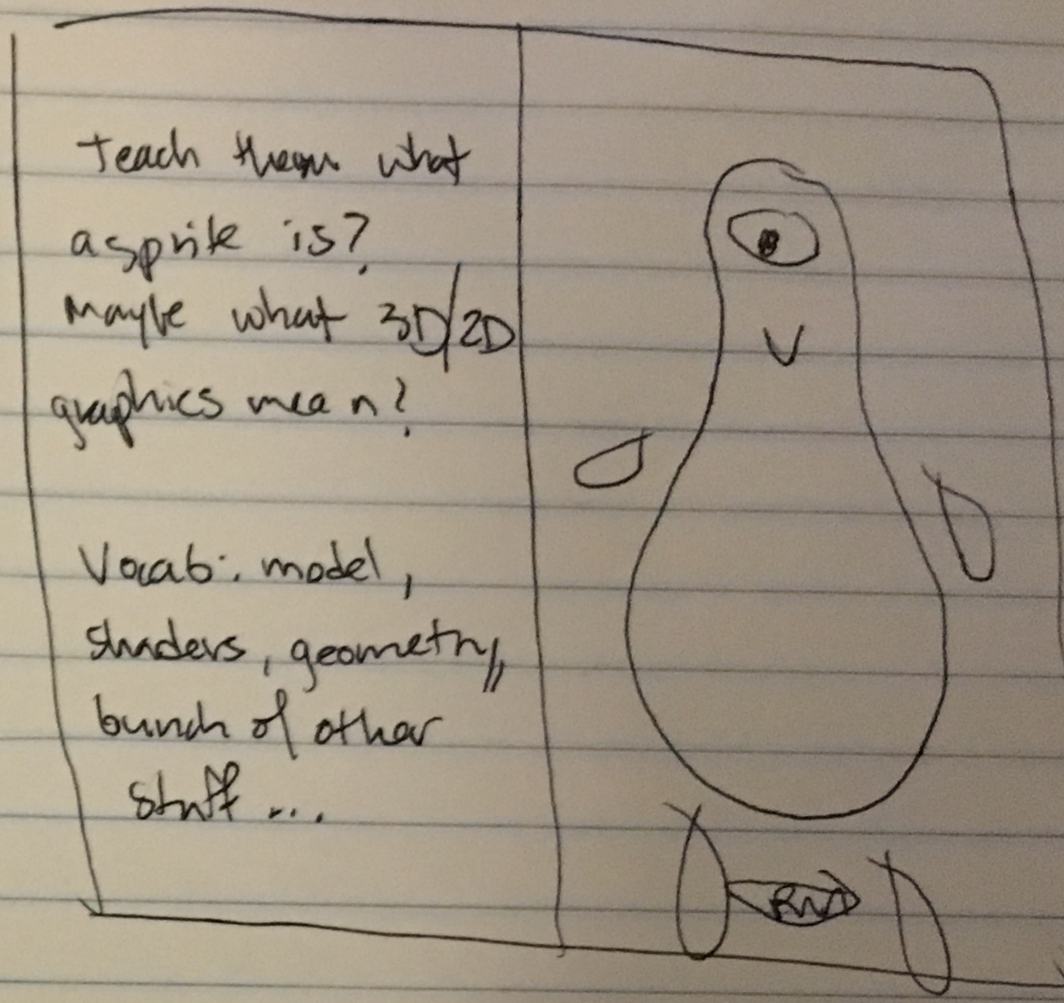
Assessments

Coding time?

To the drawing board! 🎨


Step 2
configure char?


Step 4
animate()


Teach them what a sprite is? maybe what 3D/2D graphics mean?
Vocab: model, shaders, geometry, bunch of other stuff...

What is the learner's goal?

What is the learner's goal?

Learner Goals

Learner Goals

Complete a task

"Collect 6 gems and
flip all the switches."

Learner Goals

Complete a task

Experiment

"Try changing the radius to a negative number—see what happens!"

Learner Goals

Complete a task

Experiment

Practice a new skill

"Remember for loops?
Use one now for the fastest solution!"

Learner Goals

Complete a task

Experiment

Practice a new skill

Think about a concept

"Cryptography is the science of studying hidden writing."

Learner Goals

Complete a task

Experiment

Practice a new skill

Think about a concept

Create

"Now you have all the skills you need: create your own work of art in the LiveView!"

Prose Tools

Prose Tools—Glossary Entries

Prose Tools—Glossary Entries

You'll need to write an algorithm that lets you keep your character moving efficiently around the puzzle world, picking up gems that appear.

Prose Tools—Glossary Entries

You'll need to write an algorithm that lets you keep your characte the puzzle world, picking

algorithm

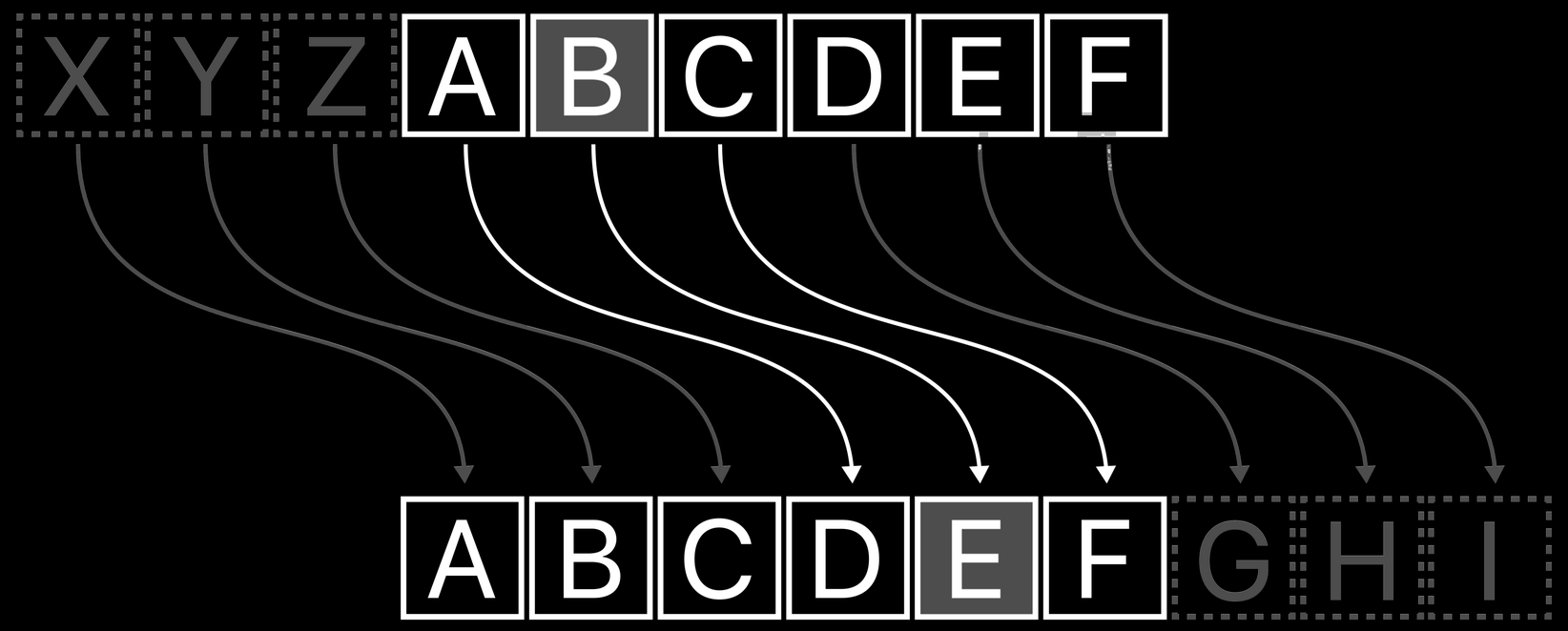
A step-by-step set of instructions or rule for solving a problem.

Prose Tools—Diagrams

"A Substitution Cipher is one in which each letter of the message is substituted (or exchanged) for a different letter."

Prose Tools—Diagrams

"A Substitution Cipher is one in which each letter of the message is substituted (or exchanged) for a different letter."



Prose Tools—Callouts

Here's a lot of prose, lots and lots of it oh yes I could go on like this forever. First I'll start making a point and giving you a bit of color and explanation.

Then I'll start the second point and etc etc etc.

> Hello There
>
> Here's the important bit!
>
> `newAwesomeFunction()`

And now for the wrap-up and a bit more prose, yes, on we go. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Prose Tools—Callouts

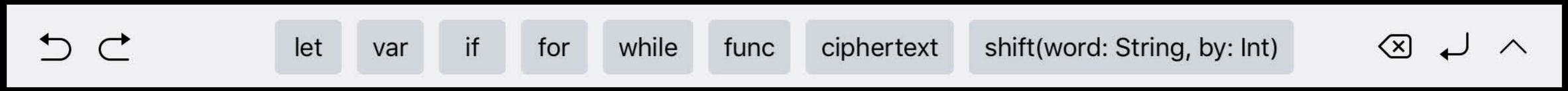
Here's a lot of prose, lots and lots of it oh yes I could go on like this forever. First I'll start making a point and giving you a bit of color and explanation.

Then I'll start the second point and etc etc etc.

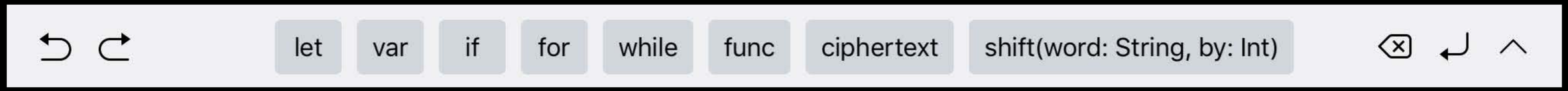
> ## Hello There
>
> Here's the important bit!
>
> `newAwesomeFunction()`

And now for the wrap-up and a bit more prose, yes, on we go. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Prose Tools—Shortcut Bar

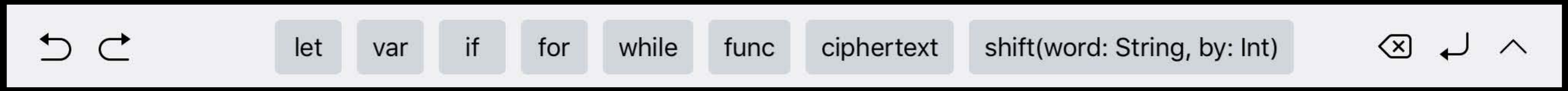


Prose Tools—Shortcut Bar



Only include relevant choices

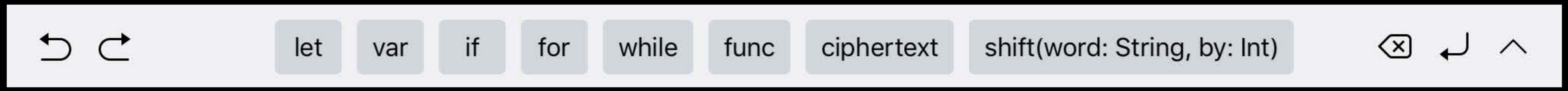
Prose Tools—Shortcut Bar



Only include relevant choices

Use concise method names

Prose Tools—Shortcut Bar

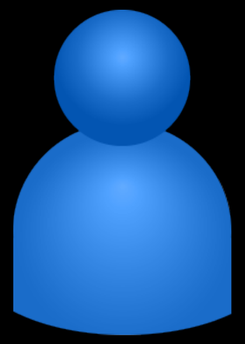


Only include relevant choices

Use concise method names

Provide common keywords

Who is your audience?



Who is your audience?

Complete a Task—Beginner

Adding a Background

Goal: Set a background image for the scene.

Think of the scene as a bit like a scene in a movie, with graphics as actors. And just like in a movie, a scene can have a background. You can set a background image for the scene.

> Setting the background image
>
> `scene.backgroundImage =`

1. Write `scene.backgroundImage =` in the code, and then choose from the shortcut bar.
2. Choose an image for your background.

```
// Set the background image.
```

Tap to enter code

Complete a Task—Beginner

Adding a Background

Goal: Set a background image for the scene.

Think of the scene as a bit like a scene in a movie, with graphics as actors. And just like in a movie, a scene can have a background. You can set a background image for the scene.

> Setting the background image
>
> `scene.backgroundImage =`

1. Write `scene.backgroundImage =` in the code, and then choose from the shortcut bar.
2. Choose an image for your background.

```
// Set the background image.
```

Tap to enter code

Complete a Task—Beginner

Adding a Background

Goal: Set a background image for the scene.

Think of the scene as a bit like a scene in a movie, with graphics as actors. And just like in a movie, a scene can have a background. You can set a background image for the scene.

Setting the background image

```
scene.backgroundImage =
```

1 Write `scene.backgroundImage =` in the code, and then choose from the shortcut bar.

2 Choose an image for your background.

```
// Set the background image.
```

Tap to enter code

Complete a Task—Beginner

Adding a Background

Goal: Set a background image for the scene.

Think of the scene as a bit like a scene in a movie, with graphics as actors. And just like in a movie, a scene can have a background. You can set a background image for the scene.

> Setting the background image
>
> `scene.backgroundImage =`

1. Write `scene.backgroundImage =` in the code, and then choose from the shortcut bar.
2. Choose an image for your background.

```
// Set the background image.
```

Tap to enter code

Complete a Task—Beginner

Adding a Background

Goal: Set a background image for the scene.

Think of the scene as a bit like a scene in a movie, with graphics as actors. And just like in a movie, a scene can have a background. You can set a background image for the scene.

> Setting the background image
>
> `scene.backgroundImage =`

1. Write `scene.backgroundImage =` in the code, and then choose from the shortcut bar.
2. Choose an image for your background.

```
// Set the background image.
```

Tap to enter code

Complete a Task—Advanced

Breadth-First Search

Goal: Use a breadth-first search algorithm to escape the maze.

Breadth-first search is a type of pathfinding algorithm. In the name of this algorithm, *breadth* refers to the number of things it checks at each step in the search. When the algorithm looks at a tile, it also checks all four of its neighbors, searching a broad area to find a path.

1. Start by adding all neighbors of the starting tile to a queue.
2. Pop tiles from the queue to see what kind of tiles they are.
3. Each time you pop a tile from the queue, add *all* of that tile's neighbors to the queue.

```
func findPath(in maze: Maze) {
    // Use the types, methods, and properties from the
     introduction page to write your own maze-solving code.
}
```

Complete a Task—Advanced

Breadth-First Search

Goal: Use a breadth-first search algorithm to escape the maze.

Breadth-first search is a type of pathfinding algorithm. In the name of this algorithm, *breadth* refers to the number of things it checks at each step in the search. When the algorithm looks at a tile, it also checks all four of its neighbors, searching a broad area to find a path.

1. Start by adding all neighbors of the starting tile to a queue.
2. Pop tiles from the queue to see what kind of tiles they are.
3. Each time you pop a tile from the queue, add *all* of that tile's neighbors to the queue.

```
func findPath(in maze: Maze) {
    // Use the types, methods, and properties from the
     introduction page to write your own maze-solving code.
}
```

Complete a Task—Advanced

Breadth-First Search

Goal: Use a breadth-first search algorithm to escape the maze.

Breadth-first search is a type of pathfinding algorithm. In the name of this algorithm, *breadth* refers to the number of things it checks at each step in the search. When the algorithm looks at a tile, it also checks all four of its neighbors, searching a broad area to find a path.

1. Start by adding all neighbors of the starting tile to a queue.
2. Pop tiles from the queue to see what kind of tiles they are.
3. Each time you pop a tile from the queue, add *all* of that tile's neighbors to the queue.

```
func findPath(in maze: Maze) {
    // Use the types, methods, and properties from the
     introduction page to write your own maze-solving code.
}
```

Complete a Task—Advanced

Breadth-First Search

Goal: Use a breadth-first search algorithm to escape the maze.

Breadth-first search is a type of pathfinding algorithm. In the name of this algorithm, *breadth* refers to the number of things it checks at each step in the search. When the algorithm looks at a tile, it also checks all four of its neighbors, searching a broad area to find a path.

1. Start by adding all neighbors of the starting tile to a queue.
2. Pop tiles from the queue to see what kind of tiles they are.
3. Each time you pop a tile from the queue, add *all* of that tile's neighbors to the queue.

```
func findPath(in maze: Maze) {
    // Use the types, methods, and properties from the
     introduction page to write your own maze-solving code.
}
```

Think About a Concept

Think About a Concept

Introduction

Blink: A Cell Simulator

Blink is a simulation that explores how a living cell reproduces or dies given a certain set of rules. Your goal is to understand the algorithms that run the simulation so that you can create your own version, with your own rules.

This playground is running a modified version of Conway's Game of Life, which presents cells reproducing and dying based upon the status of the 8 neighboring cells. You will see this simulation in the **live view** when you run the code.

The rules for this simulation are:

- Any living cell with fewer than two living neighbors dies.
- Any living cell with two or three living neighbors lives on.
- Any living cell with more than three living neighbors dies.
- Any dead cell with exactly three living neighbors becomes a living cell.

The cell simulator uses a loop to evaluate all cells on the grid. For each iteration of the loop, the rules are applied and a new generation of cells is created. Experiment with stepping through the simulation to watch this happen. On the next page, you'll explore modifying this algorithm.

Think About a Concept

Introduction

Blink: A Cell Simulator

Blink is a simulation that explores how a living cell reproduces or dies given a certain set of rules. Your goal is to understand the algorithms that run the simulation so that you can create your own version, with your own rules.

This playground is running a modified version of Conway's Game of Life, which presents cells reproducing and dying based upon the status of the 8 neighboring cells. You will see this simulation in the **live view** when you run the code.

The rules for this simulation are:

- Any living cell with fewer than two living neighbors dies.
- Any living cell with two or three living neighbors lives on.
- Any living cell with more than three living neighbors dies.
- Any dead cell with exactly three living neighbors becomes a living cell.

The cell simulator uses a loop to evaluate all cells on the grid. For each iteration of the loop, the rules are applied and a new generation of cells is created.

Think About a Concept

Introduction

Blink: A Cell Simulator

Blink is a simulation that explores how a living cell reproduces or dies given a certain set of rules. Your goal is to understand the algorithms that run the simulation so that you can create your own version, with your own rules.

This playground is running a modified version of Conway's Game of Life, which presents cells reproducing and dying based upon the status of the 8 neighboring cells. You will see this simulation in the **live view** when you run the code.

The rules for this simulation are:

- Any living cell with fewer than two living neighbors dies.
- Any living cell with two or three living neighbors lives on.
- Any living cell with more than three living neighbors dies.
- Any dead cell with exactly three living neighbors becomes a living cell.

The cell simulator uses a loop to evaluate all cells on the grid. For each iteration of the loop, the rules are applied and a new generation of cells is created.

Think About a Concept

Introduction

Blink: A Cell Simulator

Blink is a simulation that explores how a living cell reproduces or dies given a certain set of rules. Your goal is to understand the algorithms that run the simulation so that you can create your own version, with your own rules.

This playground is running a modified version of Conway's Game of Life, which presents cells reproducing and dying based upon the status of the 8 neighboring cells. You will see this simulation in the **live view** when you run the code.

The rules for this simulation are:

- Any living cell with fewer than two living neighbors dies.
- Any living cell with two or three living neighbors lives on.
- Any living cell with more than three living neighbors dies.
- Any dead cell with exactly three living neighbors becomes a living cell.

The cell simulator uses a loop to evaluate all cells on the grid. For each iteration of the loop, the rules are applied and a new generation of cells is created.

Think About a Concept

Introduction

Blink: A Cell Simulator

Blink is a simulation that explores how a living cell reproduces or dies given a certain set of rules. Your goal is to understand the algorithms that run the simulation so that you can create your own version, with your own rules.

This playground is running a modified version of Conway's Game of Life, which presents cells reproducing and dying based upon the status of the 8 neighboring cells. You will see this simulation in the **live view** when you run the code.

The rules for this simulation are:

- Any living cell with fewer than two living neighbors dies.
- Any living cell with two or three living neighbors lives on.
- Any living cell with more than three living neighbors dies.
- Any dead cell with exactly three living neighbors becomes a living cell.

The cell simulator uses a loop to evaluate all cells on the grid. For each iteration of the loop, the rules are applied and a new generation of cells is created.

Think About a Concept

This playground is running a modified version of Conway's Game of Life, which presents cells reproducing and dying based upon the status of the 8 neighboring cells. You will see this simulation in the **live view** when you run the code.

The rules for this simulation are:

- Any living cell with fewer than two living neighbors dies.
- Any living cell with two or three living neighbors lives on.
- Any living cell with more than three living neighbors dies.
- Any dead cell with exactly three living neighbors becomes a living cell.

The cell simulator uses a loop to evaluate all cells on the grid. For each iteration of the loop, the rules are applied and a new generation of cells is created. Experiment with stepping through the simulation to watch this happen. On the next page, you'll explore modifying this algorithm.

Think About a Concept

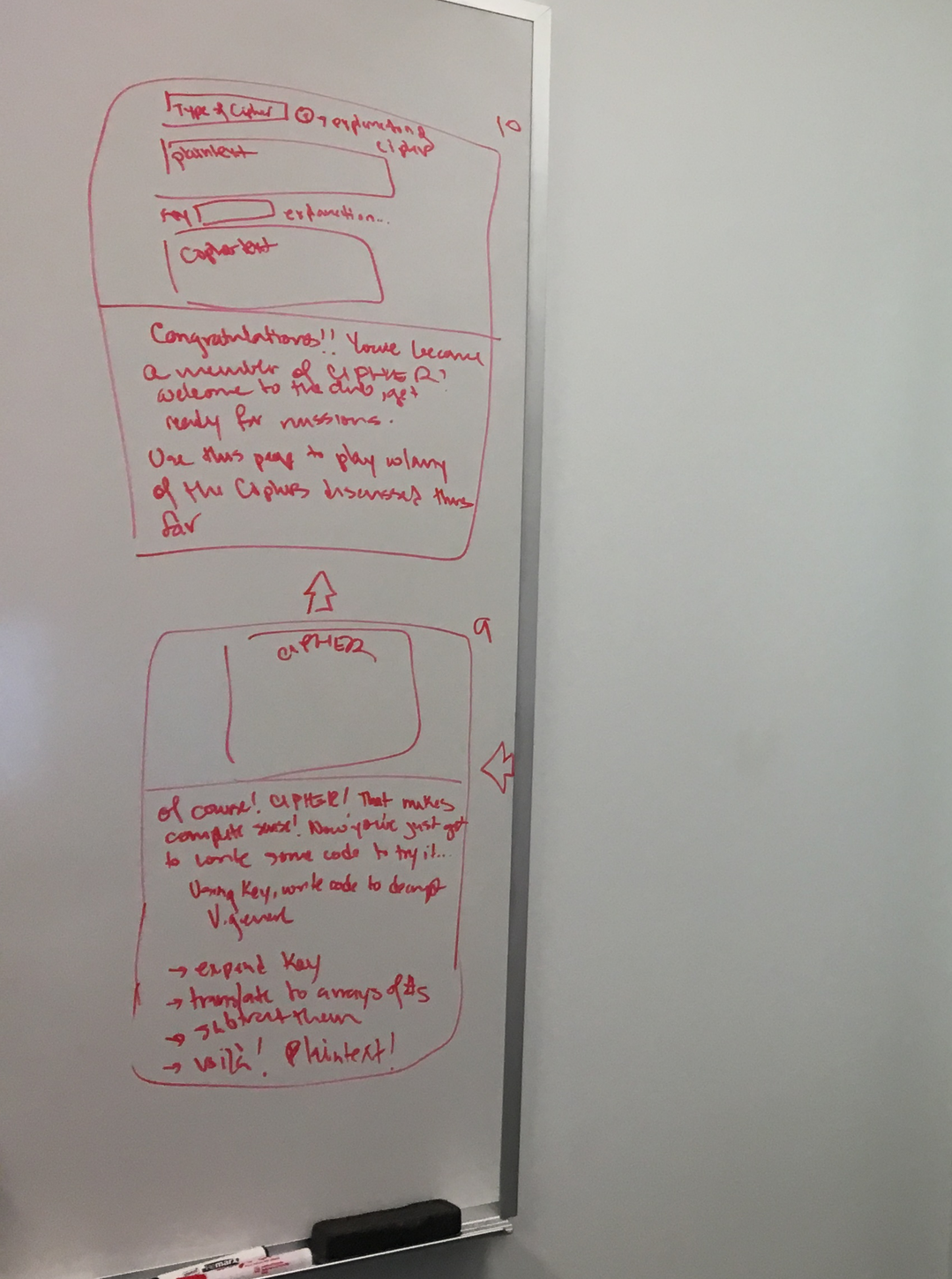
This playground is running a modified version of Conway's Game of Life, which presents cells reproducing and dying based upon the status of the 8 neighboring cells. You will see this simulation in the **live view** when you run the code.

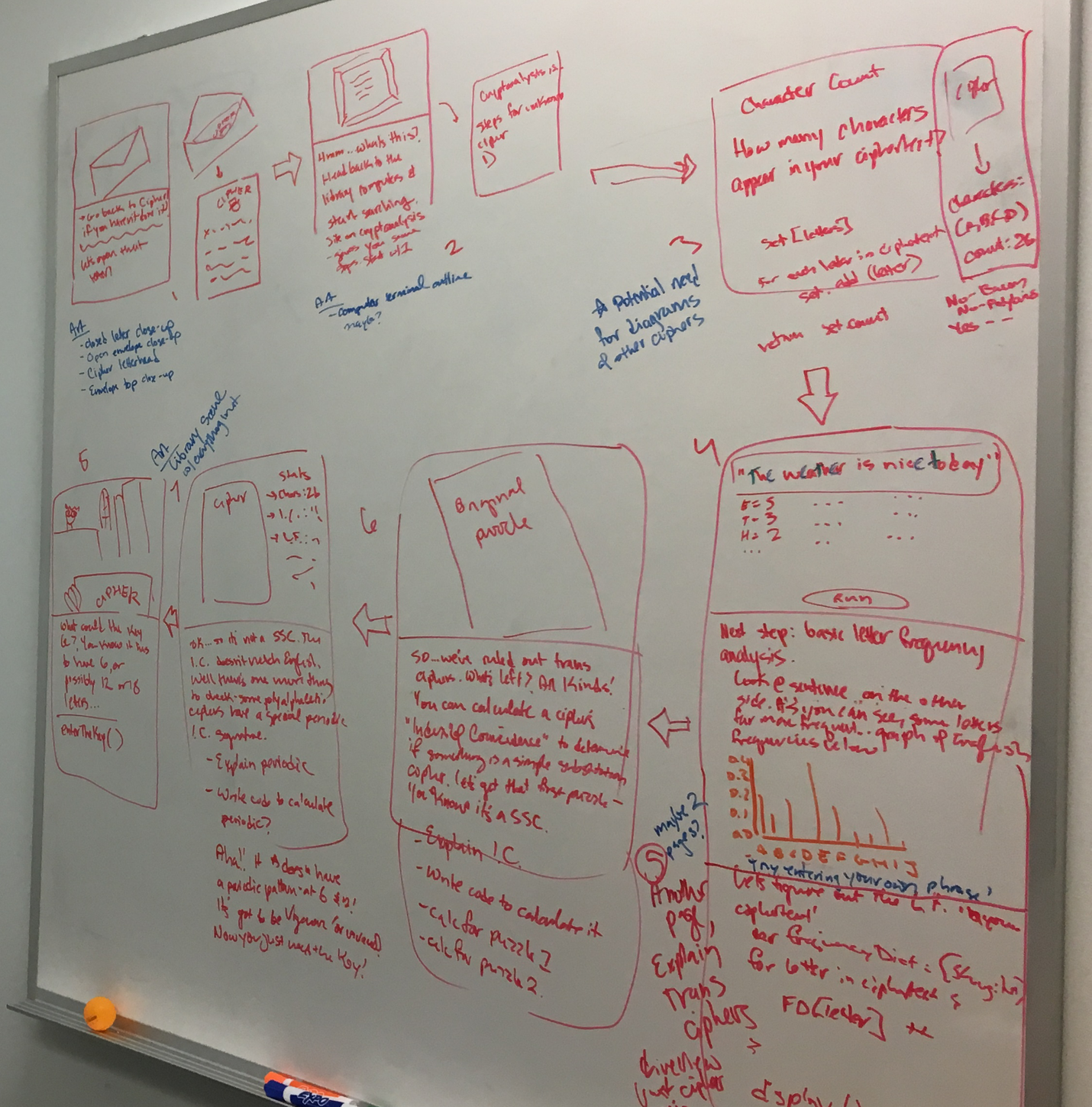
The rules for this simulation are:

- Any living cell with fewer than two living neighbors dies.
- Any living cell with two or three living neighbors lives on.
- Any living cell with more than three living neighbors dies.
- Any dead cell with exactly three living neighbors becomes a living cell.

The cell simulator uses a loop to evaluate all cells on the grid. For each iteration of the loop, the rules are applied and a new generation of cells is created. Experiment with stepping through the simulation to watch this happen. On the next page, you'll explore modifying this algorithm.

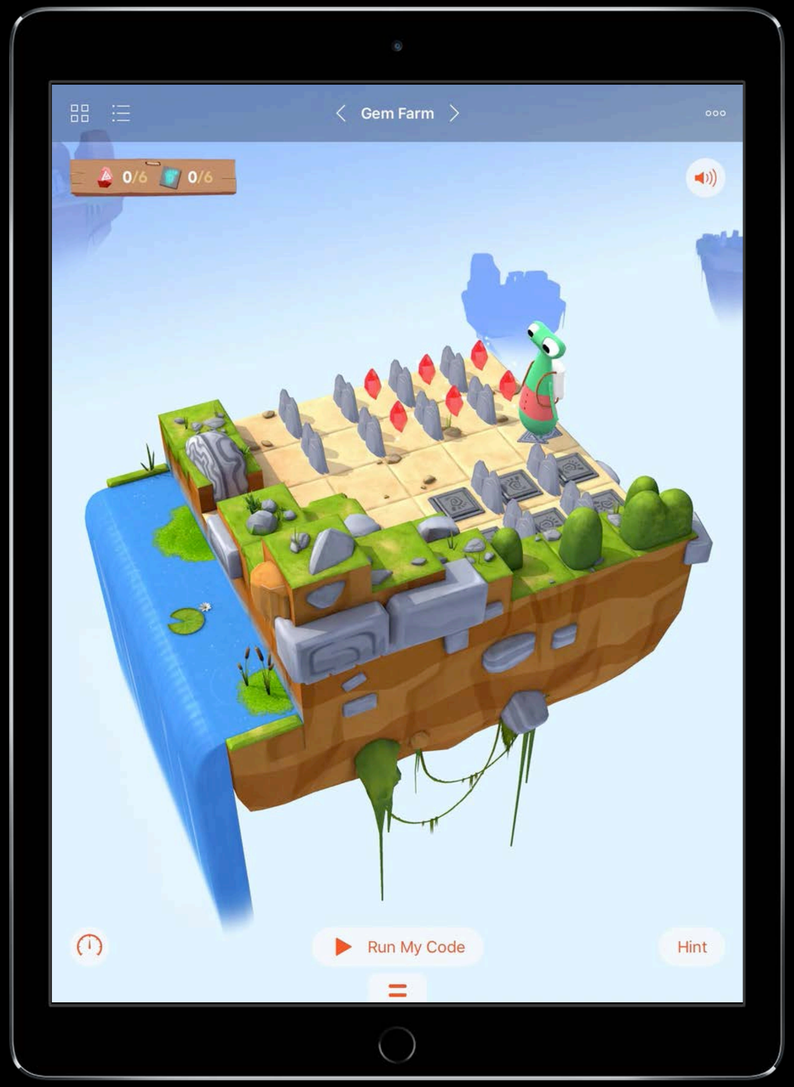
Is it coding time yet?

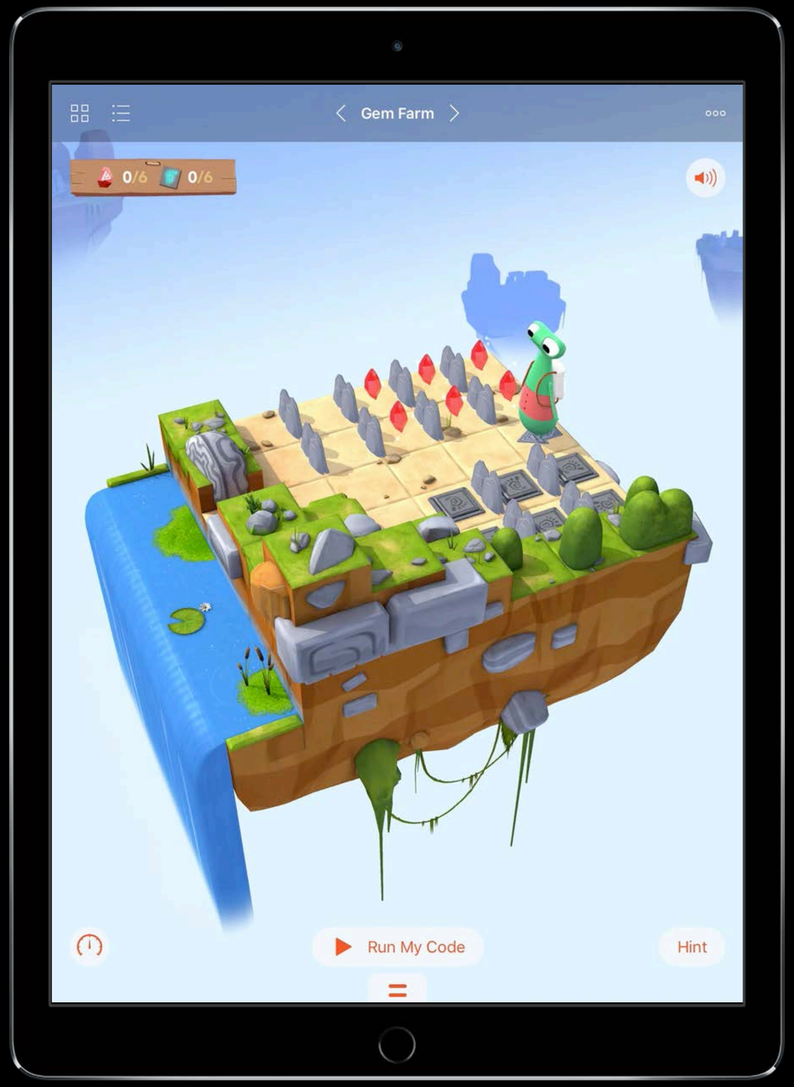

Type & Cipher
plaintext
key
ciphertext
10
Congratulations!! You've become a member of CIPHER! welcome to the club, get ready for missions.
Use this page to play w/ any of the ciphers discussed thus far
CIPHER
9
of course! CIPHER! That makes complete sense! Now you've just got to write some code to try it...
Using Key, write code to decrypt Vigenere
→ expand Key
→ translate to arrays of #s
→ subtract them
→ voilà! Plaintext!

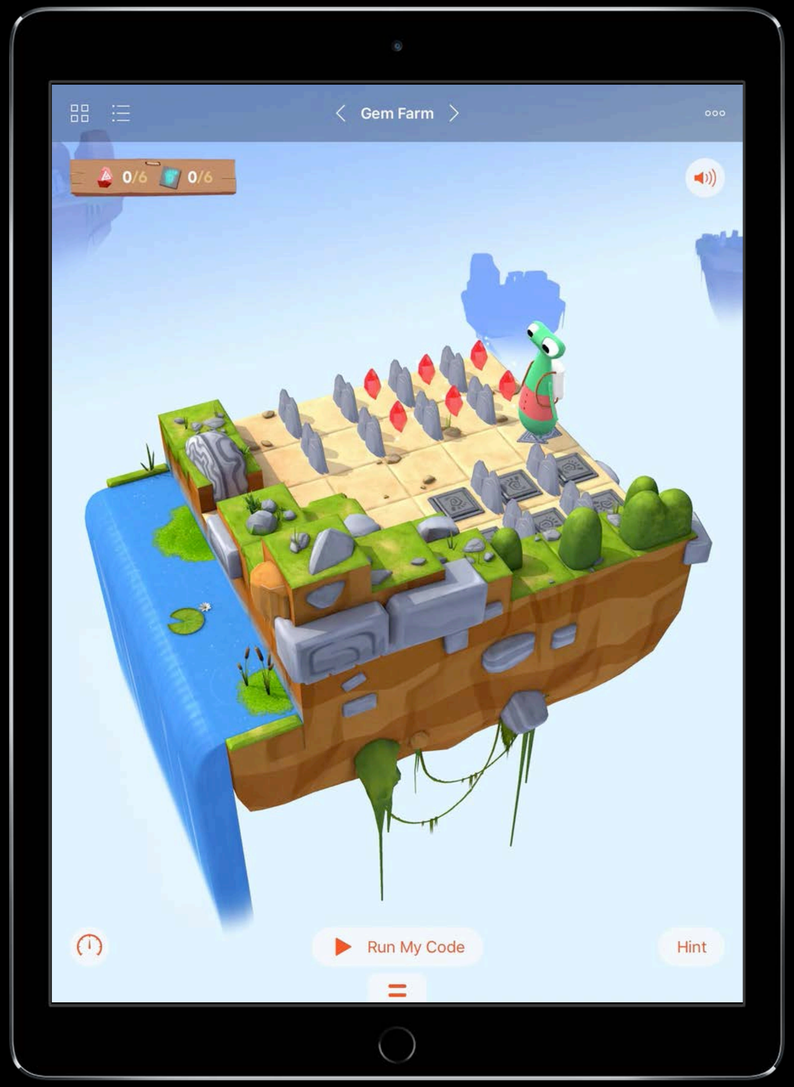

Hmm... what's this? Head back to the library computer & start searching.
Cryptanalysis is steps for cracking a cipher
2
Character Count
How many characters appear in your ciphertext?
set [letters]
for each letter in ciphertext
set.add (letter)
return set.count
cipher
characters: (A,B,C,D)
count: 26
A potential need for diagrams & other ciphers
4
"The weather is nice today"
E = 5
T = 3
H = 2
RUN
Next step: basic letter frequency analysis.
Look @ sentence on the other side. As you can see, some letters are more frequent... graph of English frequencies below
Try entering your own phrase
Let's figure out this L.F. in your ciphertext
char FrequencyDict = {char:int}
for letter in ciphertext:
FD[letter] +=
display()
maybe 2 pages?
5
Another page, Explain trans ciphers
Give new text cipher
6
Original puzzle
So... we've ruled out trans ciphers. What's left? Aha! Kinda!
You can calculate a cipher's "Index of Coincidence" to determine if something is a simple substitution cipher. Let's get that first puzzle - we know it's a SSC.
- Explain I.C.
- Write code to calculate it
- calc for puzzle 1
- calc for puzzle 2
stats
→ Chars: 26
→ I.C.: ...
→ L.F.: ...
cipher
ok... so it's not a SSC. The I.C. doesn't match English. Well there's one more thing to check - some polyalphabetic ciphers have a special periodic I.C. signature.
- Explain periodic
- Write code to calculate periodic?
Aha!! It does have a periodic pattern at 6 #s! It's got to be Vigenere (or variant). Now you just need the Key!
7
Library scene w/ computing unit
CIPHER
What could the key be? You know it has to have 6, or possibly 12 or 18 letters...
enterTheKey()
8

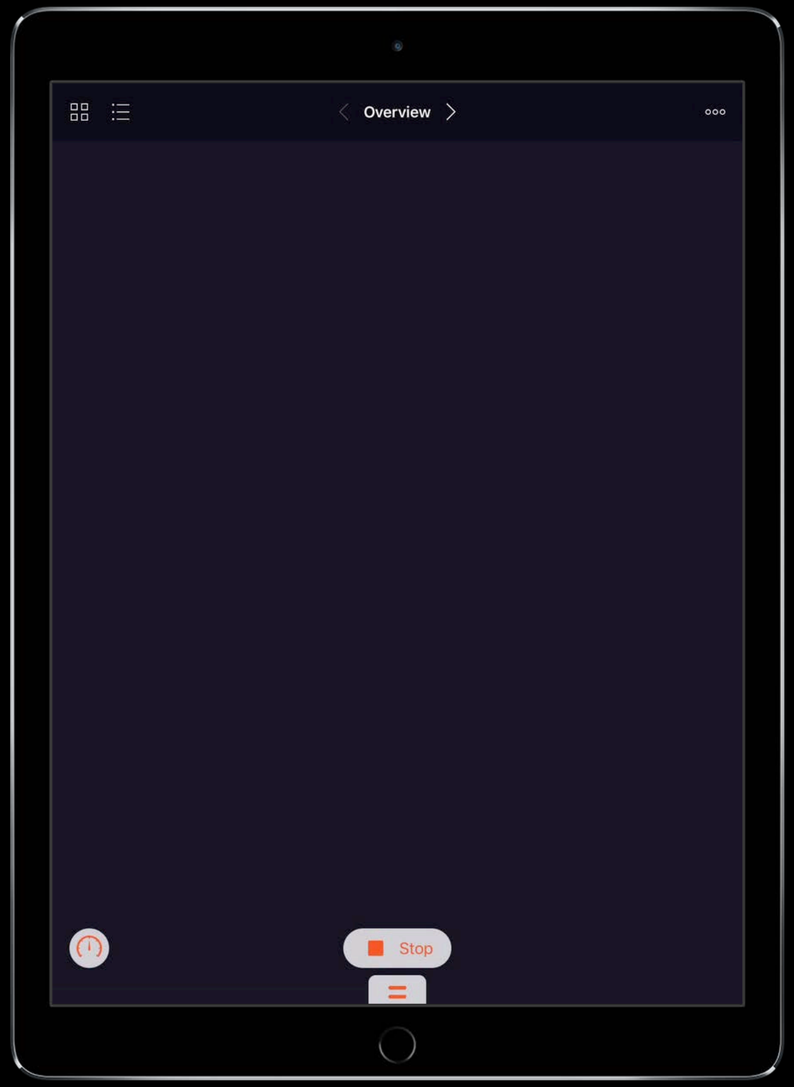
Bring your passion.

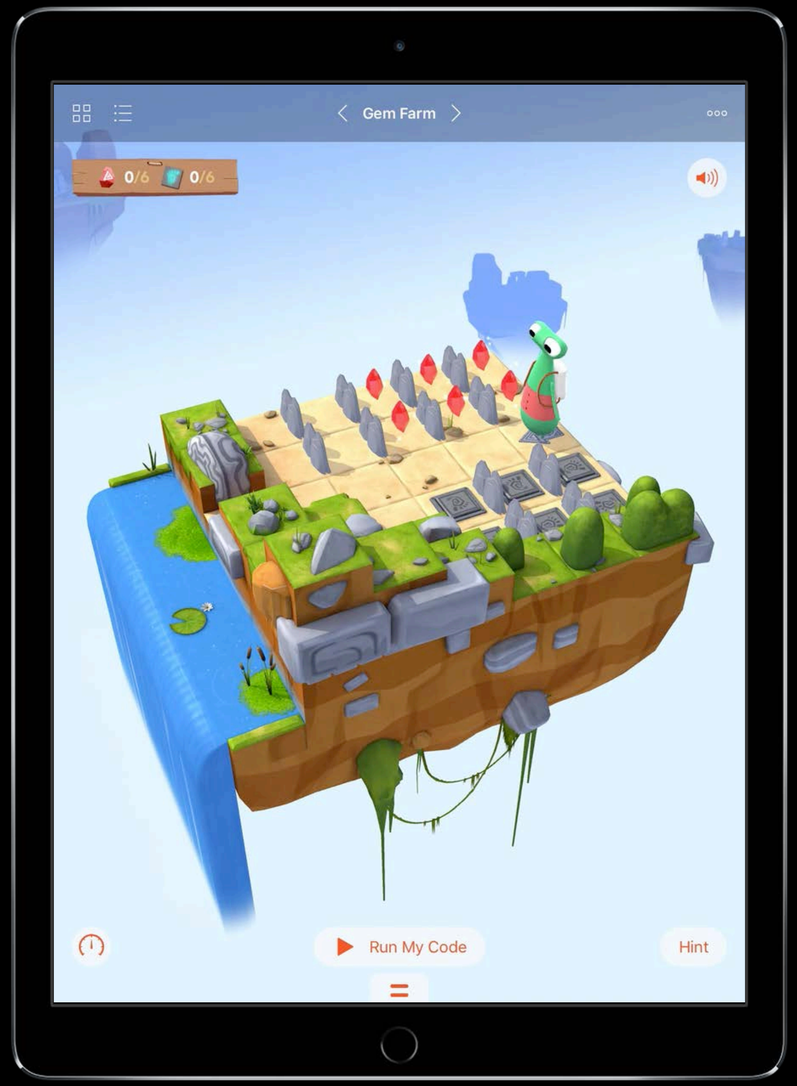
Bring your passion.

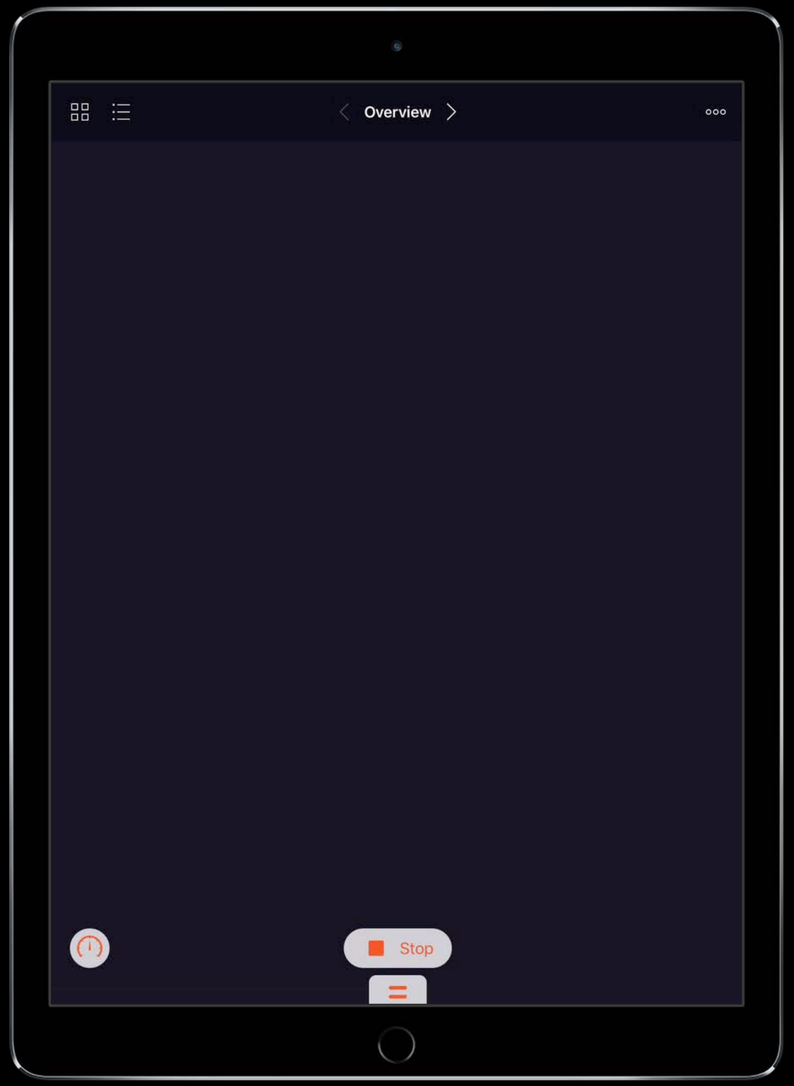

Gem Farm
0/6
0/6
Run My Code
Hint

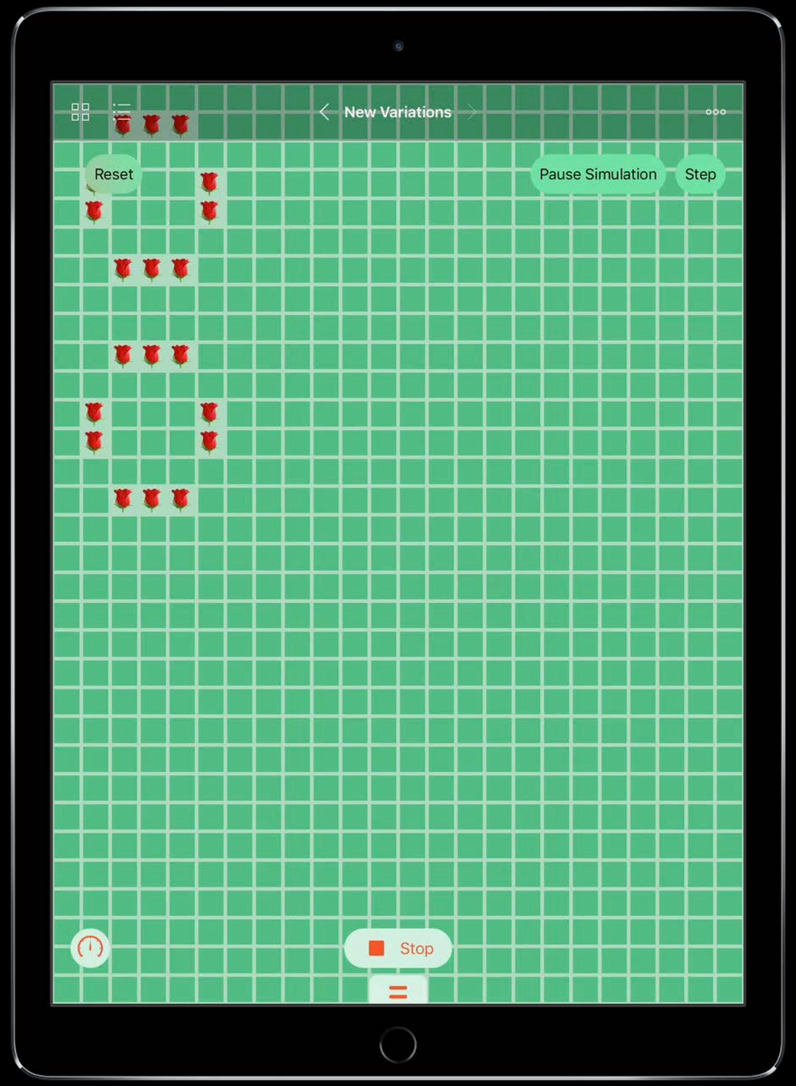

Gem Farm
0/6
0/6
Run My Code
Hint


Gem Farm
0/6
0/6
Run My Code
Hint


Overview
Stop

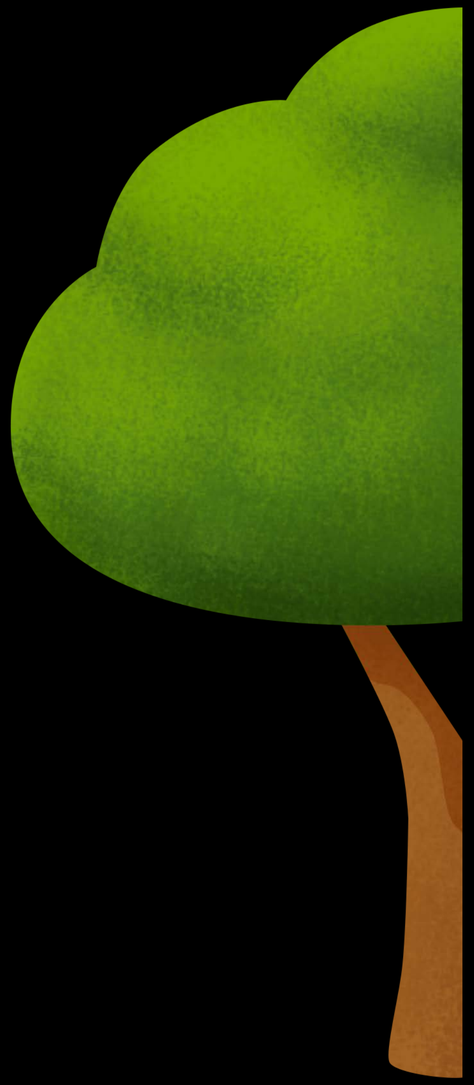

Gem Farm
0/6
0/6
Run My Code
Hint


Overview
Stop


New Variations
Reset
Pause Simulation
Step
Stop

conditional conditional
#selector return #selector return
class #line false class #line false
operators static operators static
protocol do continue protocol do continue
default #if subscript default #if subscript
#sourceLocation self #sourceLocation self
continue #function continue #function
strings nil didSet strings nil didSet
#available try #available try
static as
types true
self else
true case
nil loop
class types
didSet public

Teaching



```
conditional
#selector return
class #line false
operators static
protocol do continue
default #if subscript
#sourceLocation self
continue #function
strings nil didSet
#available try
    as
 true
else
case
loop
types
public
```

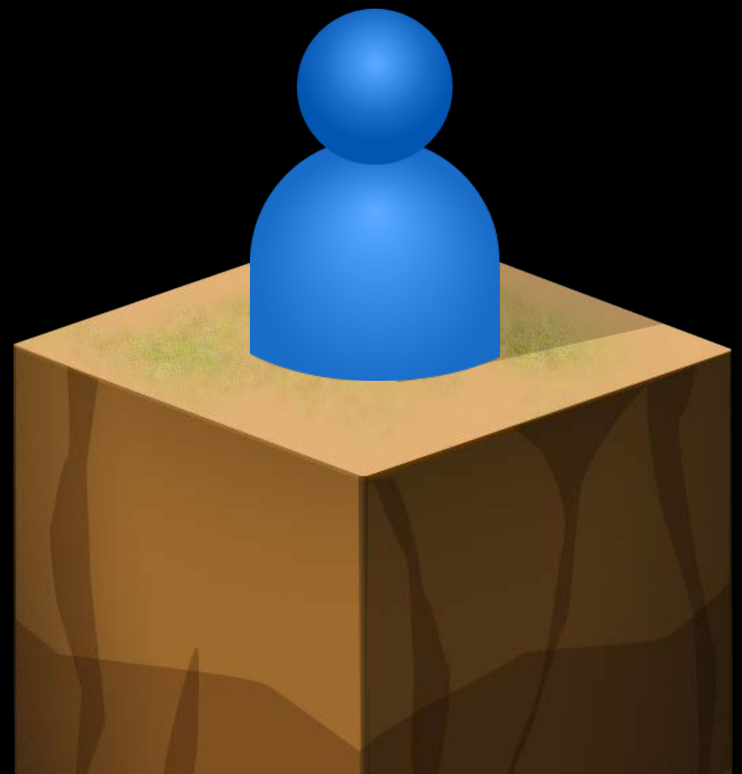
Designing

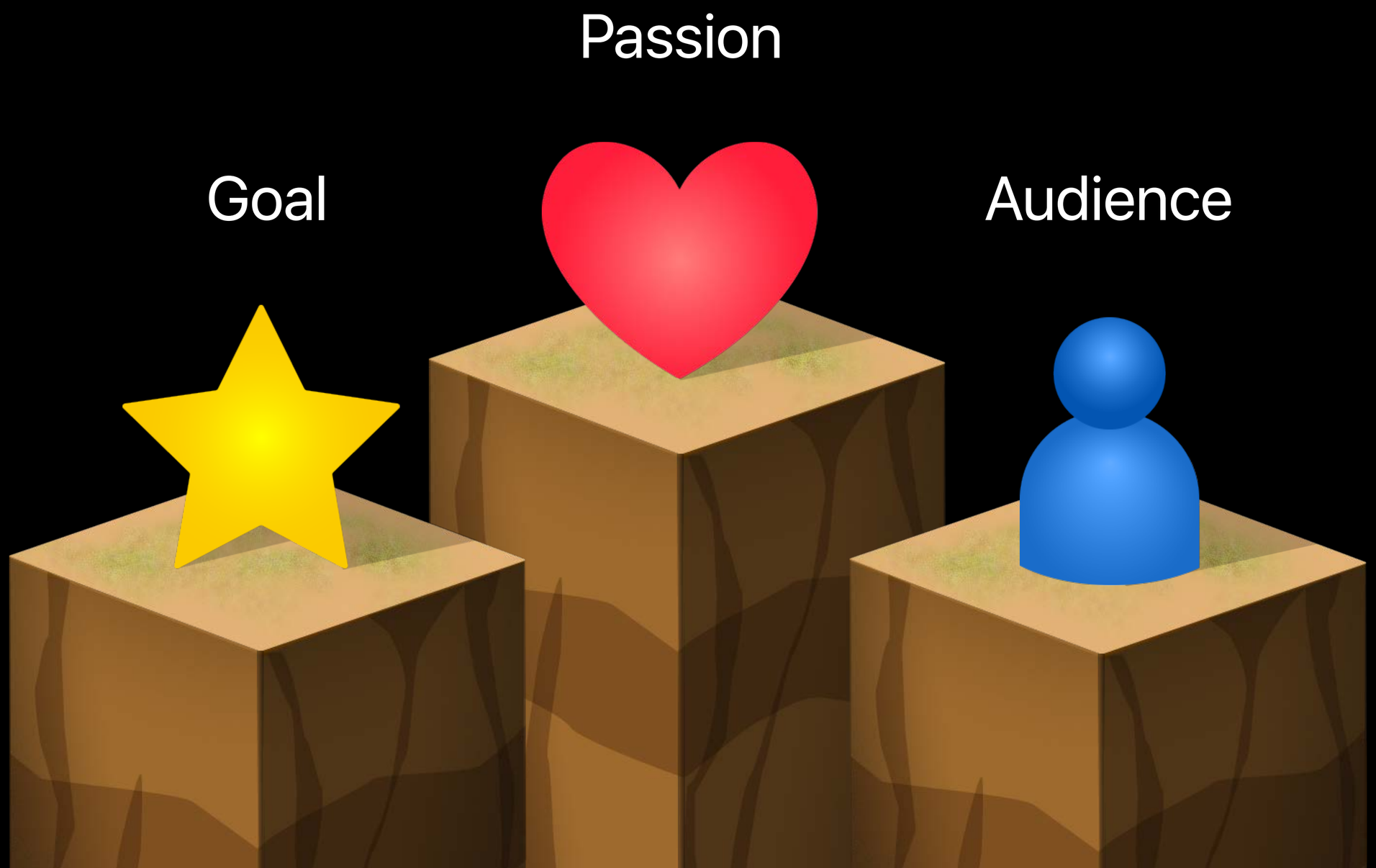
Finally, time to code.

Goal

Goal

Audience




Passion
Goal
Audience

Now go forth, and teach!

More Information

https://developer.apple.com/wwdc17/416

Related Sessions

SceneKit in Swift Playgrounds	WWDC 2017
What's New in Swift Playgrounds	WWDC 2017
Localizing Content for Swift Playgrounds	WWDC 2017

WWDC17