Platformer Lesson Plans
Table of Contents

January 2022

2345 Yale St., 1st floor
Palo Alto, CA 94306
info@codemonkey.com
www.playcodemonkey.com

TEACHER's GUIDE

LESSON Nº 1

Let's move

Exercises 1-5

PART 1

Introduction

20 min.

Platformer introduces basic programming concepts to your students through the process of building a version of the game Super Mario™.

In the game, the goal is for the player to receive the highest score.

Each banana that the monkey catches adds points to the score. There are also bonus points for getting all of the bananas before the time ends.

“Eating” a chocolate bar reduces points from the total score.

There are also obstacles and power-ups to make the game more entertaining.

The monkey is controlled by pressing the keys (specifically, the arrows keys).

Objectives:

Within this lesson, students will:

- Get introduced to the game
- Write event handlers
- Write “rules” for different values of parameters (i.e. write if statements)

Components:

- “onKey”, “step”, “if”
- tilemaps

Part 1 - Introduction

Watch 5 min.

- Open the following link and show your students the game that they will create at the end of the course.
- Ask a volunteer to play the game.
- Ask for 2 more volunteers to play the game.
- Ask your students to think of the different sprites they see in the game.

Part 1 - Introduction

Discussion 3 min.

Ask your students what sprites they saw in the game.
Make a list on the board.

The list should include the following sprites:

- Monkey
- Banana
- Chocolate
- Tiger
- Power up

Part 1 - Introduction

Discussion 3 min.

Ask your students to think of properties that all the sprites poses in the context of the game.

Make a list on the board.

The list should include the following:

Properties:

- Position
- Direction
- Speed
- Size

Note:

Students may find some other properties too such as color or proportion between width and length and so on.

The properties listed here are properties that your students can manipulate.

Part 1 - Introduction

Activity #1 5 min.

The following activity facilitates the understanding of an event and an event handler.

Ask for 3 volunteers:

- Nominate one volunteer to be the “events causer”
- Nominate the other two volunteers to respond to an event (be the “event handler”)

Agree with the “event causer” on 2 different events. For example:

1. Raising hands
2. Jumping

Agree with each of the “event handler”s that they are “programmed” to respond to one of the events and see what their response is.

Part 1 - Introduction

Activity #1 (con't) 5 min.

For example:

1. "event handler" #1:
    - event - jumping
    - response - walks 5 steps
2. "event handler" #2:
    - event - raising hands
    - reponse - turn around

Ask the "event causer" to start doing the events and ask the "event handlers" to respond to their events.

For example:

- when the "event causer" jumps, "event handler" #1 walks 5 steps
- when the "event causer" raise hands, "event handler" #2 turns around

Part 1 - Introduction

Explain 2 min.

Events are actions triggered by the user that take place while the code is running. As a result, events affect what is happening.

For example, computer events are:

- Pressing a key on the keyboard
- Clicking on the mouse
- Scrolling the mouse

When we press a key on the keyboard, we trigger the keyboard event and the onKey function is called. Such functions are also called event handlers.

The onKey function receives an argument which is the key we pressed. The code in the onKey function will run only when we press the keyboard.

Part 1 - Introduction

Activity #2 5 min.

This activity is a preparation for an if statement and parameter handling.

This activity is based on the previous one.

Add a "parameter" to the events. For example, use the following parameters:

1. Raising hands - right or left depending on the hand that was raised
2. Jumping - the number of legs (1 or 2)

The “event handlers” will respond differently according to the parameter.

Part 1 - Introduction

Activity #2 (con't) 5 min.

1. Raising hands:
   - **if** the left hand was raised
     - turn left
   - **if** the right hand was raised
     - turn right
2. Jumping:
   - **if** jumping on 1 leg
     - walk 5 steps forward
   - **if** jumping on both legs
     - walk 5 steps backwards

Part 1 - Introduction

Activity #2 (con't) 5 min.

Ask the "events causer" to cause some more events including raising your left and right hands and jumping on one and both legs.

Ask the "event handlers" to respond according to the events agreed on and based on the "parameters".

PART 2

Let’s Go!

20 min.

Course Structure:
The Course is divided into exercises. Each exercise includes tasks that need to be completed. Upon successful completion of an exercise, the next exercise becomes available.

Each task ends with clicking the “check” button, after making the changes in the code, as required in the task. If the code is correct, the next task becomes available, if not, the task is marked as “failed” and retry is needed.

When the Check button is not available, the code should be tested by using the Run button. Usually when running the code, the player is asked to get the monkey to the banana.

Bonus Tasks

Encourage your students to complete these as well. Bonus tasks are more complicated and most of them are not step by step instructions like . in the tasks within the exercises.

Some bonus tasks include topics that were covered in previous exercises.

Part 2 – Let's Go!

Play Time 10 min.

Ask your students to complete exercises 1 - 5.

Encourage your students to also read the instructions in each exercise, especially if they do not understand what they need to do.

Use your classroom's dashboard to keep track of students' achievements.

Part 2 – Let's Go!

Walkthrough 5 min.

Open Exercise #1:

Ask one of your students to solve it. Your code should be:

```
@onKey = (key) =>

    @step 1
```

Run the game, press on any key and show the monkey is moving.

We do not check the argument key - therefore the monkey moves just by pressing on any key.

For example - press on the key ‘s’, ‘t’ and the right arrow.

Part 2 – Let's Go!

Walkthrough (con't) 5 min.

Open Exercise #2:

```
@onKey = (key) =>

    if key == keyboard.left

        @step -1

    if key == keyboard.right

        @step 1
```

Run the game, press on any key other than the right or left arrow and show that the monkey does not move. Explain that this is because the step function is called only when the right or left arrow key is pressed.

Press the right arrow key and show that the monkey steps to the right.

Part 2 – Let's Go!

Play Time 5 min.

Students should complete up to exercise 5.

Use this time to pass between your students and help them.

PART 3

Debriefing

5 min.

Part 3 – Debriefing

Discussion 5 min.

Discuss with your students what we covered in this lesson:

1. Name a few sprites that we will be using in this game
   - monkey, banana, chocolate, tiger
2. Name a few properties of the sprites:
   - position, direction, size, speed
3. What is an event handling function?
   - A function that is called when the event happens and then does something
   - An example is the onKey function that is called when we press the keyboard
4. What happens if we do not write an event handling function?
   - The event will still occur but nothing will happen
5. What are parameters for?
   - We can decide to do different things based on the parameter
   - For example - decide which way to move the sprite

TEACHER's GUIDE

LESSON Nº 2

Collide with me

Exercises 6-9

PART 1

Introduction

20 min.

In this lesson, your students will get a better understanding of a sprite's code.

This lesson introduces a new event handling - when two sprites collide.

Your students will learn how to add a new sprite.

Objectives:

Within this lesson, students will:

- Use the concept @ vs. sprite name when calling a function
- Learn to use the collide event
- Write an event handler for the collide event.
- Add a new sprite

Components:

- @ vs. sprite name
- "onCollide", "destroy"
- Add new sprite

Part 1 - Introduction

Activity #1 5 min.

The following activity facilitates the understanding of using a @ vs. sprite name when calling a function.

Ask for 2 volunteers, let us assume their names are Johnny and Emily.

Let Emily go first and let her write instructions on the board for Johnny and herself.

For example:

```
I step 4

I turn left

Johnny turn right

Johnny step backwards 2
```

Let Emily read the instructions out loud and both of them perform them.

Emily takes 4 steps and turns left and then Johnny turns right and steps back 2 steps.

Part 1 - Introduction

Activity #1 (con't) 5 min.

Now, let Johnny write his instructions on the board for Emily and himself.

For example:

```
I step 4

I turn left

Emily turn right

Emily step backwards 2
```

Let Johnny read the instructions out loud and have both Emily and Johnny perform them.

This time, Johnny will take 4 steps and turns left and then Emily will turn right and take 2 steps back.

Part 1 - Introduction

Explain 5 min.

Open Exercise #4

In this exercise there are two sprites: monkey and banana.

Show your students that when clicking on a sprite (either on the game itself or under the Sprites tab), the editor in the middle changes.

This editor is the sprite's own code. The sprite's name is displayed at the top. Switch between the sprites to show that the code changes accordingly.

When a function is called, the name of the sprite needs to be added to know which sprite to apply the function on.

For example, in this exercise, there are 3 functions in the monkey's code: onKey, step and jump.

Before each function the @ symbol is added.

Part 1 - Introduction

Explain (con't) 5 min.

The @ symbol refers to the sprite in which the code is written. Writing the sprite's name (in the sprite's code) and @ is equivalent.

Copy the code from the monkey's sprite to the banana's sprite (delete the code in the monkey's sprite). Ask your students what they think will happen now?

- the banana will move

Run the game and show them that when the arrows keys are pressed the banana moves and not the monkey.

Ask your students what should be changed in the banana's code in order to move the monkey?

- monkey. should replace the @
  - make sure to add a dot after the sprite's name

Part 1 - Introduction

Explain (con't) 5 min.

The code should be:

```
@onKey = (key) =>
    if key == keyboard.left
        monkey.step -1
    if key == keyboard.right
        monkey.step 1
    if key == keyboard.up
        monkey.jump()
```

Run the game again to see that now the monkey moves when the arrow keys are pressed.

Part 1 - Introduction

Explain (#2) 10 min.

Emphasize event handling some more:

There are different events that can occur. Only events that have event handlers are handled. In other words, the event handlers trigger an action that is executed when the event happens. For example, if there is a keyboard event handler that moves the monkey, when a key is pressed, the monkey will step.

Part 1 - Introduction

Explain (con't) 10 min.

Open Exercise #3:

Show your solution, run the game and move the monkey to collide with the banana.

Ask your students:

- What happens in our game when the monkey and banana collide?
  - Answer: nothing
- Now ask: why?
  - Answer: because we did not write an event handler for that event.

The event where two sprites touch each other is the collision event.

Part 1 - Introduction

Explain (con't) 10 min.

The function onCollide is called when two sprites collide with each other, or, in other words, when a collision event occurs.

The function can be written in either of the sprites' code - the result will be the same.

The onCollide function, needs to get references to the 2 sprites:

```
@onCollide sprite, () =>
```

The first argument is the other sprite that our sprite collides with.

The second argument (), is the current sprite. The empty parentheses are how we can refer to the current sprite.

Part 1 - Introduction

Explain (con't) 10 min.

For example, if we want something to happen as a result of a collision between the monkey and **banana**, the following can written in the **banana**'s code:

```
@onCollide monkey, () =>
```

In this case:

- () is the **banana's** sprite
- monkey is the sprite colliding with the **banana's** sprite

Ask: What would we write if we would like to write the handler in the monkey's code?

```
@onCollide banana, () =>
```

Part 1 - Introduction

Explain (con't) 10 min.

In our game, when the monkey and banana collide, the monkey "eats" the banana so the banana is then destroyed.

The function destroy() destroys a sprite.

<u>Option 1:</u>

Add the following code to the banana:

```
@onCollide monkey, () =>

   @destroy()
```

Run the game. Now when the monkey and banana collide, the banana is destroyed.

Delete the code in the banana and continue onto option 2.

Part 1 - Introduction

Explain (con't) 10 min.

Option 2:

Write the following in the monkey's code:

```
@onCollide banana, () =>

    banana.destroy()
```

Can you see the difference between both options?

- The sprite's name (in bold) is different
- The name of the sprite when calling the destroy function
  - @ or banana

Part 1 - Introduction

Explain (con't) 10 min.

Ask your students what will happen if we write the following code (in the monkey's code):

```
@onCollide banana, () =>
    @destroy()
```

The monkey will be destroyed. Run the game and get the monkey to the banana.

Now ask your students how to destroy both sprites when they collide? Choose one of the following options and run the game.

In the **monkey**'s code:

```
@onCollide banana, () =>
    @destroy()
    banana.destroy()
```

In the **banana**'s code:

```
@onCollide monkey, () =>
    monkey.destroy()
    @destroy()
```

PART 2

Let’s Go!

20 min.

Course Structure:
The Course is divided into exercises. Each exercise includes tasks that need to be completed. Upon successful completion of an exercise, the next exercise becomes available.

Each task ends with clicking the “check” button, after making the changes in the code, as required in the task. If the code is correct, the next task becomes available, if not, the task is marked as “failed” and retry is needed.

When the Check button is not available, the code should be tested by using the Run button. Usually when running the code, the player is asked to get the monkey to the banana.

Bonus Tasks

Encourage your students to complete these as well. Bonus tasks are more complicated and most of them are not step by step instructions like . in the tasks within the exercises.

Some bonus tasks include topics that were covered in previous exercises.

Part 2 – Let's Go!

Play Time 10 min.

Ask your students to complete exercises 6 - 9.

Encourage the class to also read the instructions in each exercise, especially if they do not understand what needs to be done.

Use your classroom dashboard to keep track of your students' achievements.

Part 2 – Let's Go! Walkthrough 5 min.

Open Exercise #9:

This exercise explains how to add a sprite to the game.

Go over with your students the steps on how to add a sprite:

1. Click on the Sprites tab
    - Lower right side
2. Click on the green + button
    - The Add a Sprite screen is open
3. Choose a sprite to add to the game
    - In this exercise the banana sprite should be added
4. Click on the Add button

When a sprite is added, a default name is given - the same as the sprite (i.e. banana when adding a banana sprite). If there is already a sprite with that name, then a number will added as well (banana1, banana2 and so on).

Emphasize to your students that in the course when there is a task to add a sprite (also widgets or sounds later on), the name given in the task must be used. Otherwise the task will not be marked as completed.

Part 2 – Let's Go!

Walkthrough (con't) 5 min.

Add a hippo sprite, the default name is hippo.

Click on the sprite's settings icon (⚙)

There are various properties that can be changed from here (we will explore them in the next lesson). Point out that the word hippo is the name of the sprite.

Add another hippo sprite, the default name will be hippo1.

Make a note that the task was not marked as complete. This is because there is no sprite with the name banana.

Part 2 – Let's Go!

Play Time 5 min.

Students should complete up to exercise 9.

Use the time to pass between your students and help them.

Encourage students who complete all exercises to solve also the bonus task in exercises #7 and #9.

PART 3

Debriefing

5 min.

Part 3 – Debriefing

Discussion 5 min.

Discuss with your students what we covered in this lesson:

1. What happens when we write the following code?

   @step 1

   - the sprite in which the code is written will step.
     - if it is the monkey's sprite, then the monkey will move
     - if it is the hippo's sprite, then the hippo will move
2. Ask for a volunteer. Tell him to add a hippo sprite and to destroy it when it collides with the monkey
   - Ask your students if there is another way to code this:
     - Of course - we can write the onCollide function in the monkey, or the hippo

monkey:

```
@onCollide hippo, () =>
    hippo.destroy()
```

hippo:

```
@onCollide monkey, () =>
    @destroy()
```

TEACHER's GUIDE

LESSON Nº 3

Make it fun

Exercises 10-15

PART 1

Introduction

15 min.

In this lesson, your students will be introduced to sprite properties.

This lesson introduces new game features such as game world and sounds.

Objectives:

Within this lesson, students will:

- Change a sprite's property
- Set the sprite's position in the "world" of the game
- Change the world's size
- Add sounds
- Use a loop

Components:

- Allow gravity
- "setX", "setY"
- Camera target
- "play"
- loop

Part 1 - Introduction

Activity #1 5 min.

The following activity facilitates the translation of a sprite's position into x and y axes.

Ask for a volunteer. Ask him to stand 4 steps from the door and 2 steps from the board.

After he moves into position, ask your students to describe the volunteer's position.

Ask for another volunteer, ask him to stand 3 steps from the board and 1 step from the door.

Again, ask your students to describe the new volunteer's position.

Tell your students to think of one direction as the x axis and the other as the y axis. For example:

- the line from the door to the wall/window is the **x** axis
- the line from the board to the wall is the **y** axis

Part 1 - Introduction

Activity #1 (con't) 5 min.

Define a point where the board and door meet, tell your students that this is 0,0. Meaning:

- x = 0
- y = 0

Now, ask your students to describe the position of the volunteers again:

1. Volunteer #1:
   a. x = 4; y = 2
2. Volunteer #2:
   a. x = 3; y = 1

Ask for a third volunteer, ask him to position himself at x = 5 and y = 2

- he should move 5 steps from the door and 2 steps from the board

Part 1 - Introduction

Explain 5 min.

The world in our game has two coordinates: x and y. When we place a sprite in the game we set its x and y position.

Open Exercise #9:

Click the monkey's settings icon ⚙ Show your students the value of x and y.

You can also show your students the x and y coordinates of the mouse. Move the mouse on the world and show your students that the coordinates appear in the top left corner.

Part 1 - Introduction

Explain (con't) 5 min.

Show your class that when you move the mouse to the top left corner, x and y are both 0. Move the mouse to bottom right corner and show that the x is 600 and the y is 400 (actually, you can move the mouse very close to these numbers).

This is the default world size.

If we want to change the position of the sprite during the game, we should use the following two functions:

```
@setX
```

```
@setY
```

These functions changes the x and y positions of the sprite it refers to.

Explain that some properties can be changed both from the code and from the settings, like the x and y properties.

Part 1 - Introduction

Explain #2 5 min.

The game has a few settings that the user can change.

These settings can be updated from the Game tab (on the lower right side).

Open Exercise #8

Click on the Game tab and review the features with your students:

- World width and World height
  - the world of the game has a size of 600 (x axis) by 400 (y axis)
  - move your mouse along the boundaries of the world and show your students that the value of x and y changes
  - again, point out that the top left hand corner is 0,0 and as you move to the right, x increases and as you move down, y increases

Part 1 - Introduction

Explain (con't) 5 min.

Change the value of the World's width to 1800

- In the Platformer game, the monkey steps to the right and left, so there is no need to change the world's height

Run the game and press the arrow key to move the monkey to the right. Show your students that the monkey "disappears".

This happens because we only see the world in its default size. When the size of the world is bigger than the default, the game should move along with the sprite.

This is another setting:

- Camera target
  - There is an option to define which sprite the camera follows. When that sprite moves, the game moves with it.

Part 1 - Introduction

Explain (con't) 5 min.

So how do you choose which sprite to follow? The sprite to follow should be the main sprite that moves in the game. In our case, it is the monkey.

You can show your students that if they choose to follow the banana, nothing will happen since this sprite does not move.

Other settings that can be changed in the game, however, in this course, we will not be changing them:

- Physics
  - in this course, use the arcade option
- Background
  - some other backgrounds option exist
  - there is also an option to upload your own background

Part 1 - Introduction

Explain (con't) 5 min.

- Gravity
  - defines the gravity of the game, i.e. how fast or slow sprites will fall
  - change the gravity (to 500, for example), run the game and show your students that now the monkey jumps much higher
  - another difference is that when you press run, the monkey "falls" slower than with the default gravity

Each sprite has a property that defines whether the sprite is affected by the gravity or not

  - affected means that the sprite will fall down until it reaches a tile or the world boundary
  - not affected means that when the game runs, it stays in its place

Part 1 - Introduction

Explain (con't) 5 min.

Refresh exercise #8:

Open the monkey's sprite and show your students the property Allow gravity. For the monkey's sprite, this setting is checked.

Run the game and show your students that the monkey "falls" until it reaches the world's boundary.

Ask your students if they think this property is checked or unchecked for the banana?

- it is also checked because the banana also falls when the game starts
  - open the banana's settings and show them this property is checked

You can add another banana sprite and un-check this property, run the game and show the difference between the two bananas' sprites.

PART 2

Let’s Go!

25 min.

Course Structure:
The Course is divided into exercises. Each exercise includes tasks that need to be completed. Upon successful completion of an exercise, the next exercise becomes available.

Each task ends with clicking the “check” button, after making the changes in the code, as required in the task. If the code is correct, the next task becomes available, if not, the task is marked as “failed” and retry is needed.

When the Check button is not available, the code should be tested by using the Run button. Usually when running the code, the player is asked to get the monkey to the banana.

Bonus Tasks

Encourage your students to complete these as well. Bonus tasks are more complicated and most of them are not step by step instructions like . in the tasks within the exercises.

Some bonus tasks include topics that were covered in previous exercises.

Part 2 – Let's Go!

Play Time 5 min.

Ask your students to complete exercises 10 - 15.

Encourage your students to also read the instructions in each exercise, especially if they do not understand what needs to be done.

Use your classroom dashboard to keep track of students' achievements.

Part 2 – Let's Go!

Walkthrough 5 min.

The **Sounds** tab includes sounds that we can add to the game.

When we add a sound, we need to name it.

The sound has one function - to play the sound.

Open exercise #10:

Go to the Sounds tab.

This tab shows the sounds we have in the game. When we press the green + button, we will be able to see the different sounds that we can add to the game. Show your students that we can play the sounds before we add them in order to chose the ones we want.

Ask your students when they think we should add sounds? For example:

- When the monkey gets the banana
- When the monkey collides with the tiger

Part 2 – Let's Go!

Walkthrough #2 5 min.

Open exercise #7:

Click to show the solution for this exercise.

In this exercise, the hippo stands still and the monkey jumps on it. Tell your students, what if we want to make the game harder and move the hippo back and forth?

Emphasize, that the hippo should repeat stepping to the right and left.

Ask them which functionality should be used to code this?

- loops

Ask them which loops they are familiar with:

- their answer might include: times, for, until, while

Part 2 – Let's Go!

Walkthrough #2(con't) 5 min.

You can review with your students the differences between each of the loops:

- times - runs x times, x is specified in the syntax
- for - runs on all items in the collection
- until - runs until the condition is met (stops when the condition is changed to true)
- while - runs as long as the condition is true (stops when the condition is changed to false)

Here, the loop is called loop and it does not have any conditions. This means that it runs for as long as the game runs.

The syntax of this loop is:

```
loop

    @step 150

    @step -150
```

This will make the sprite move to the right and left, over and over again for as long as the game runs.

Add this code to the hippo's sprite and run the code. Make sure to move the monkey when the hippo fills the gap or else the monkey will not be able to get to the banana.

Part 2 – Let's Go!

Play Time 10 min.

Students should complete up to exercise 15.

Use this time to pass between your students and help them.

Encourage students who complete all exercises to solve also the bonus task in exercise #12.

PART 3

Debriefing

5 min.

Part 3 – Debriefing

Discussion 5 min.

Tell your students to create a mini game:

- the monkey steps back and forth
- the banana falls from the sky
  - each time the monkey gets the banana, a “new” banana falls

Your students can change the gravity of the game to make the banana fall down slower. They can also add a sound each time the monkey and banana collide.

Click this link for an example of such a game.

The code should be:

monkey:

```
loop
    @step 300
    @step -300
```

banana:

```
@onCollide monkey, () =>
    @setY 0
```

TEACHER's GUIDE

LESSON Nº 4

Count on me

Exercises 16-21

PART 1

Introduction

15 min.

The first exercise in this lesson is a recap on everything your students learned until now.

This lesson introduces the Counter widgets.

Objectives:

Within this lesson, students will:

- Use the counter widget to keep score
- Learn to use the operators += and -=

Components:

- Counter widget
- Operators += and -=

Part 1 - Introduction

Activity #1 5 min.

The following activity facilitates the understanding of the operators += and -=.

Ask for 2 volunteers.

- One will be the “giver”
  - give him a bag of marbles (or anything he can give to the other student)
  - tell him to say how many marbles he gives each time to the “container”
    - “add 4 marbles”
    - “add 3 marbles”
    - “add 5 marbles”
- The other will be the “container”
  - give him a basket
  - tell him each time the “giver” gives him marbles to say out loud the total
    - “I have 4 marbles”
    - “I have 7 marbles”
    - “I have 12 marbles”

Part 1 - Introduction

Explain #1 5 min.

Write on the board:

x = x + 5

Discuss with your students how to translate this equation into words:

Add 5 to x, save the new value in x.

The value of x is unknown but it does not matter. Each time the line is executed, the value of x is increased by 5.

Now write on the board:

y = y - 2

Discuss with your students how to translate it into words:

Reduce 2 from y, save the new value in y.

Part 1 - Introduction

Explain (con't) 5 min.

In computer science there is a shorter syntax for these equations:

- +=
  - add a value to a variable
  - x += 2
    - add 2 to x
- -=
  - reduce a value from a variable
  - y -= 3
    - reduce 3 from y

Part 1 - Introduction

Explain #2 5 min.

The **Widgets** tab includes different types of widgets.

Widgets are objects that we can be added to the game. Each widget has a specific functionality. It does not “step” like sprites do.

Examples of Widgets: Counter, Text, Timer, Clock, Dialog, Button.

Widgets do not have too many properties. All of them have the Show property - we can decide if we want to show the widget in the game or not.

Each widget also has unique properties.

We will learn about most of these widgets later in the course. For now, we will focus on the Counter widget.

Part 1 - Introduction

Explain (con't) 5 min.

The Counter Widget is used mainly to count something in the game.

Ask your students if they can think of examples of something we want to count?

- score
- lives
- collected items

The counter widget has a property named value which stores the value of the counter.

The default is set to 0, but can be changed.

If we have a counter named count, then we can refer to it as:

```
count.value
```

Part 1 - Introduction

Explain (con't) 5 min.

Usually we will want to change the value of the counter, increase or decrease it.

For example:

- if we want to count the items we collected, we will increase it by 1
- if we want to add points to keep score, we will increase it by 100 each time the monkey gets a banana.

Use the operator **+=** to increase a value. This operator adds the number on the right to the variable on the left.

This example adds 5 to a variable named var:

```
var += 5
```

- if its original value was 0, then it is now 5
- if its original value was 10, it is now 15

PART 2

Let’s Go!

25 min.

Course Structure:
The Course is divided into exercises. Each exercise includes tasks that need to be completed. Upon successful completion of an exercise, the next exercise becomes available.

Each task ends with clicking the “check” button, after making the changes in the code, as required in the task. If the code is correct, the next task becomes available, if not, the task is marked as “failed” and retry is needed.

When the Check button is not available, the code should be tested by using the Run button. Usually when running the code, the player is asked to get the monkey to the banana.

Bonus Tasks

Encourage your students to complete these as well. Bonus tasks are more complicated and most of them are not step by step instructions like . in the tasks within the exercises.

Some bonus tasks include topics that were covered in previous exercises.

Part 2 – Let's Go!

Play Time 10 min.

Ask your students to complete exercises 16 - 21.

Encourage students to also read the instructions in each exercise, especially if they do not understand what needs to be done.

Use your classroom's dashboard to keep track of students' achievements.

Part 2 – Let's Go!

Walkthrough 5 min.

Open exercise #17:

Write the following code:

```
@onCollide hippo, () =>

    @jump()

    jumping.play()

    count.value = 1
```

Discuss with your students what will happen each time the hippo and monkey collide and why.

- the first time they collide, the counter will be updated to 1
- after the first time, it will stay at 1
    - this because we are not adding a value but assigning a value

Run the game and show your students that the counter remains the same after the first collision.

Add another sprite and assign a different value (5).

Part 2 – Let's Go!

Walkthrough (con't) 5 min.

Add a tiger sprite and place it somewhere the monkey can get to.

In the monkey's sprite add the following code:

```
@onCollide tiger, () =>
    @jump()
    jumping.play()
    count.value = 5
```

Run the game. Then, move the monkey to the hippo and notice how even though it keeps on colliding with (i.e. jumping on) the hippo, the counter remains 1. Now, move the monkey to the tiger, and notice how it does not matter how many times the monkey collides with (jumps on) the tiger, the counter remains at 5. When the monkey collides with the tiger, the property value is updated to 5. The value does not change because 5 is assigned to the property and not added.

Fix the code to add 5 each time the monkey and tiger collide.

```
count.value += 5
```

Run the game and show that each time the monkey collides with the tiger, 5 is added to the counter. When the monkey collides with the hippo, the counter is updated to 1 and remains at 1 no matter how many times the monkey and hippo collide.

Repeat it a few times to show the difference between = and +=.

Part 2 – Let's Go!

Play Time 10 min.

Students should complete up to exercise 21.

Use this time to pass between your students and help them.

Encourage students who complete all exercises to also solve the bonus tasks in exercise #17 and #20.

PART 3

Debriefing

5 min.

Part 3 – Debriefing

Discussion 5 min.

Discuss with your students what was covered in this lesson:

1. Counter widget
    - What is the counter's property name?
        - value
    - How do we update this property?
        - counter.value
2. Adding and subtracting operators:
    - **+=**
    - **-=**

Discuss with your students how to show how many bananas are left for the monkey to collect.

- need to add a counter
    - set the value to be the total number of bananas
- in each banana sprite, reduce 1 from the counter (when the monkey collides with it)
- you can also add a sound when there are no bananas left
    - in each banana, check if value is 0, then play a sound

Part 3 – Debriefing

Discussion 5 min.

You can use one of the exercises solved in this lesson - for example Exercise #21.

You can also click on Create Games and build a new game for this purpose. Use this link as an example.

Remind your students that they can make the world size bigger and move some of the bananas further to the right.

Add a counter and name it **count**.

Change the initial value to the number of bananas.

In the example there are 5 banana, so it is set to 5.

```
@value = 5 # instead of 0
```

Part 3 – Debriefing

Discussion 5 min.

The code in each banana:

```
@onCollide monkey, () =>
    @destroy()
    collectBanana.play()
    # Update your score here
    score.value += 100
    count.value -= 1 # reduce 1 from count
    if count.value == 0 # add this if you want to play a sound
        end.play()      # when all the bananas were collected
```

The code in bold is the addition.

Make sure to add a sound and name it **end**.

TEACHER's GUIDE

LESSON Nº 5

Parameters for all

Exercises 22 - 28

PART 1

Introduction

15 min.

This lesson will introduce new functions. Your students will call the functions each time with different arguments.

This lesson will also introduce the Timer widget. Your students will learn its functions.

Objectives:

Within this lesson, students will:

- Use functions with different arguments
- Use a Timer widget and its functions:
  - start
  - onEnd

Components:

- “jump”, “setSpeed”, “setScale”
- “hide”, “show”
- Timer widget

Part 1 - Introduction

Explain 15 min.

[Open exercise 21]:

Click on the monkey sprite. Discuss with your students the functions that are called here and their parameters in this sprite:

- step - parameter is a number
- jump - no parameter

If they also say onKey - this is the function's definition. It is called each time a key is pressed on the keyboard.

When a function has no argument, we use parentheses ().

Functions without an argument:

- play
  - `play()`
- destroy
  - `destroy()`

Functions with an argument:

- setX number
  - `setX 0`
- setY number
  - `setY 0`

Part 1 - Introduction

Explain (con't) 15 min.

There are two more functions your students will use today without arguments:

- hide()
  - hides a sprite or widget
- show()
  - shows a sprite or widget

Tell your students: there is also a checkbox in the sprite's or widget's settings. Discuss the difference between that checkbox and these functions:

- the hide/show is a property that can be defined both before and during the game
- checkbox - to set this property before the game starts, to make it the initial state (shown or hidden) of the sprite/widget
- functions - to change it during the game

Part 1 - Introduction

Explain (con't) 15 min.

Ask your students if they can think of another property that can be defined both from the sprite's settings and during the game.

- x, y
  - in the sprite's settings, the user can change the initial position of the sprite
  - setX and setY changes the sprite's position during the game

Now, ask if they can think of a property that can only be changed before the game starts?

- gravity
  - this is because the developers of the game builder thought the gravity is not something that should change during the game.

Today we will review properties that can only be changed during the game.

Part 1 - Introduction

Explain (con't) 15 min.

Before we discuss the new functions, we will discuss the parameters of the functions.

Note:

- parameter - is a variable in the function definition
- argument - is the actual value that is passed when calling the function

The argument of the "step" function is a number. The monkey steps only 1 step each time the arrow key is pressed.

Your students also saw a call to the step function with a different parameter:

- in the tiger's sprite
  - @step 200

Part 1 - Introduction

Explain (con't) 15 min.

Change the parameter of step in the monkey's sprite to 20:

@step 20

Run the game and show the monkey steps more each time.

Some functions can be defined both with a parameter or without.

- If the function is called without an argument, then a default value is used.
- If the function is called with an argument, then the value of that argument is used

Part 1 - Introduction

Explain (con't) 15 min.

An example of such a function is the jump. Up until now, this function was called without an argument.

```
@jump()
```

The monkey jumped to the same height each time.

This function can also be called with an argument.

For example:

```
@jump 3
```

Change the code and run the game to show that the monkey jumps much higher now.

Part 1 - Introduction

Explain (con't) 15 min.

Discuss with your students what other properties the sprite has:

- speed
- scale

This defines how fast/slow the sprite moves and how big/small it appears in the game.

These properties can not be changed before the game. The game always starts with the default speed and scale of the sprites.

To change these properties there are two functions:

- setSpeed
- setScale

The argument of both functions is a number

- it sets the sprite's speed/scale to the value passed

Part 1 - Introduction

Explain (con't) 15 min.

The speed and scale change based on the default values.

For example, if we want to change the sprite's scale to be twice as big, the code will be:

```
@setScale 2
```

Open exercise 20:

Add another key to change the monkey's scale. For example:

```
@onKey = (key) =>
    ...
    # Add this:
    if key == 'c'
        @setScale 2
```

Run the game, press on 'c' and show that the monkey is bigger.

Part 1 - Introduction

Explain (con't) 15 min.

Click on the key 'c' a few times and show your students that the monkey stays the same scale (the monkey is not getting bigger).

- This is because each call to the setScale function with the argument 2 changes its scale to be twice as big as the default scale.

To set the sprite back to the default speed/scale, use the value 1 when calling these functions.

```
@setScale 1
```

Part 1 - Introduction

Explain (con't) 15 min.

Add the following code to the monkey's sprite:

```
@onKey = (key) =>

    ...

    # Add this:

    if key == 'r'

        @setScale 1
```

Run the game, click on the keys 'c' and 'r' to switch between the sizes of the monkey.

In the example above, the monkey changes its scale when a key is pressed. What if we want the monkey to be bigger right from the beginning of the game?

Add the following code before the definition of the onKey function:

```
@setScale 2
```

PART 2

Let’s Go!

20 min.

Course Structure:
The Course is divided into exercises. Each exercise includes tasks that need to be completed. Upon successful completion of an exercise, the next exercise becomes available.

Each task ends with clicking the “check” button, after making the changes in the code, as required in the task. If the code is correct, the next task becomes available, if not, the task is marked as “failed” and retry is needed.

When the Check button is not available, the code should be tested by using the Run button. Usually when running the code, the player is asked to get the monkey to the banana.

Bonus Tasks

Encourage your students to complete these as well. Bonus tasks are more complicated and most of them are not step by step instructions like . in the tasks within the exercises.

Some bonus tasks include topics that were covered in previous exercises.

Part 2 – Let's Go!

Play Time 10 min.

Ask your students to complete exercises 22 - 28.

Encourage your students to also read the instructions in each exercise, especially if they do not understand what needs to be done.

Use your classroom's dashboard to keep track of students' achievements.

Part 2 – Let's Go!

Walkthrough 5 min.

This is the first time we used a Timer widget.

When we add a timer widget, we need to start it and specify how many seconds we want it to run for.

For example, if we want to start counting down from 5 seconds, we will use:

```
timer.start 5
```

We can start the timer in the timer's code or from any other sprite or widget.

When we start it from the timer's code, the timer will start the running once we run the game.

When we start the timer from any other sprite/widget code, the timer will start running when we get to that code.

Note that if we start the timer in another sprite/widget but not inside a function, the timer will also start running when we run the game.

Part 2 – Let's Go!

Walkthrough (con't) 5 min.

The Timer widget has another function - an event function that is called each time the timer ends. The function is called onEnd and it is already defined when we add a Timer widget.

Go to the Widgets tab and click on the timer widget. Show your students that the function onEnd is defined there.

```
@onEnd = () =>

    # Your code here
```

If we don't write any code inside this function, then nothing will happen when the timer ends.

Open Exercise #25:

Go to the Widgets tab and click on the timer. Discuss what happens in this exercise when the time ends:

- a sound is played

The code of the timer does not include the start function. In this exercise, starting the timer is called from the monkey's code. Go to the Sprites tab, click on the monkey and show where this function is called:

- when the monkey collides with the power up

For how many seconds will the timer run?

- 5 seconds

Part 2 – Let's Go!

Play Time 5 min.

Students should complete up to exercise 28.

Use the time to pass between your students and help them.

Encourage students who complete all exercises to also solve the bonus task in exercises #23 and #27.

PART 3

Debriefing

10 min.

Part 3 – Debriefing

Discussion 10 min.

Change the monkey's scale some more.

Use two Timer widgets. Each time a timer ends it will:

1. start the other timer
2. change the monkey's scale
    - for example to 0.5 and to 1.5

You can add this to one of the exercises, for example to #28.

You can use the following link as an example.

Make sure to start one of the timers. If you want to make the monkey bigger or smaller in the beginning, make sure to also change the monkey's scale.

In the monkey's code, add the following:

```
@setScale 1.5
```

Part 3 – Debriefing

Discussion 10 min.

Assume that the timer widgets' names are timer1 and timer2.

The code in the timer1:

```
@start 2  # this starts the timer when the game start running
@onEnd = () =>
     monkey.setScale 0.5
     timer2.start 2
```

The code in the timer2:

```
@onEnd = () =>
     monkey.setScale 1.5
     timer1.start 2
```

You can also add code to hide and show the timers (as in the example).

TEACHER's GUIDE

LESSON Nº 6

Yes or no?

Exercises 29-32

PART 1

Introduction

20 min.

In this lesson, your students will learn about and use if/else statements. They will also define and use a sprite's property.

Your students will learn to duplicate a sprite.

Objectives:

Within this lesson, students will:

- Learn if/else
- Learn to define and use a sprite's property
- Duplicate sprite

Components:

- if/else
- Sprite's property
- Duplicate sprite

Part 1 - Introduction

Activity #1 8 min.

The following activity facilitates the understanding of if/else conditionals.

Ask for 2 volunteers and refer to them as player1 and player2. Taking turns, each will roll two dice and add the numbers.

The rules of the game:

- If the sum is an even number
  - the player that rolled the dice gets 1 point
- if the sum is an odd number
  - the other player gets 1 point

Draw a table on the board to keep score.

Let the volunteers start playing and each time ask:

- “What is the sum of the dice?”
- “Is it an even number?

Part 1 - Introduction

Activity #1 (con't) 8 min.

Ask your students to write the pseudo-code for this game.

- a common way to express a set of actions (or an algorithm) without writing actual code is called pseudo-code

They might suggest the following code:

```
if sum is even
    me gets 1 point
if sum is odd
    opponent gets 1 point
```

Although this code will run correctly, the proper syntax will be to use if/else:

```
if sum is even
    me gets 1 point
else
    opponent gets 1 point
```

Part 1 - Introduction

Activity #1 (con't) 8 min.

Now, add another rule to the game:

- If the sum is an even number
  - if both dice show the same number
    - the player that rolled the dice gets 2 points
  - otherwise
    - the player that rolled the dice gets 1 point
- if the sum is an odd number
  - the other player gets 1 point

Let them play and ask them each time:

- "What is the sum of the dice?"
- "Is it an even number?"
- if yes, "do both dice show the same number?"

Part 1 - Introduction

Activity #1 (con't) 8 min.

How should the pseudo-code be updated with the new rule of the game?

```
if sum is even
    if same number
       me gets 2 points
    else
       me gets 1 point
else
    opponent gets 1 point
```

Part 1 - Introduction

Explain 6 min.

Oftentimes when writing code, there is a need for only **one** of the two following scenarios to happen:

- do one action if a condition is met
- do another action in case the condition is not met

Discuss the following with your students:

1. What was the condition in the activity?
    - If the sum of the two dice is even
2. What happened when the condition was met (true)?
    - the player gets a point
3. What needs to be the sum of the dice for the condition not to be met (false)?
    - the sum is an odd number
4. What happens when the condition is not met?
    - the opponent gets a point

Part 1 - Introduction

Explain (con't) 6 min.

Open exercise #2:

Show the solution for the exercise.

In the onKey function there are 2 if statements.

Discuss with your students what will be the difference if we change this code to an if/else statement.

Write the two codes on the board:

2 ifs:

```
@onKey = (key) =>
    if key == keyboard.left
        @step -1
    if key == keyboard.right
        @step 1
```

if/else:

```
@onKey = (key) =>
    if key == keyboard.left
        @step -1
    else
        @step 1
```

Part 1 - Introduction

Explain (con't) 6 min.

The difference:

- when we run the game with the code of the two ifs:
  - only when the user presses on the right arrow key, the monkey will step to the right
- when we run the game with the code of the if/else:
  - the monkey will step to the right when the user presses any key except for the left arrow key.

Run the game and show both options. Make sure to to press various keys in each run to emphasize the difference.

Part 1 - Introduction

Explain #2 6 min.

Open exercise #28:

Switch between the sprites and discuss with your students the different types of code:

1. Functions' definition and implementation
   - onKey, onCollide
2. Calling functions
   - with argument
     - step, jump, setScale
   - without arguments
     - play, destroy
3. Conditional statements
   - if
4. Updating a property (of the Counter widget)

Part 1 - Introduction

Explain (con't) 6 min.

In the exercises your students will complete today they will use and define a property.

The value of the property will be used to check the mode of the monkey - super mode or regular.

In each case, the monkey and the tiger will behave differently.

- super mode:
  - monkey is bigger and faster
  - colliding with the tiger makes the tiger jump
- regular mode:
  - monkey's scale and speed is regular
  - colliding with the tiger sets the monkey back to the beginning

Part 1 - Introduction

Explain (con't) 6 min.

The monkey starts with regular mode and only after the monkey "takes" (collide with) the power up, its mode is changed to super mode.

The super mode lasts for a few seconds and then the monkey returns to its regular mode.

Discuss with your students how this can be implemented:

- define a property in the monkey's sprite
- initialize the property
- when the monkey collides with the power up:
  - change the property value
  - start the timer
  - change the monkey to be bigger and faster
- When the timer ends:
  - change the property value
  - change the monkey to be its regular scale and speed

Part 1 - Introduction

Explain (con't) 6 min.

- when the monkey collides with the tiger:
  - if the monkey is in super mode
    - make the tiger jump
  - else
    - return the monkey to its beginning position

PART 2

Let’s Go!

20 min.

Course Structure:
The Course is divided into exercises. Each exercise includes tasks that need to be completed. Upon successful completion of an exercise, the next exercise becomes available.

Each task ends with clicking the “check” button, after making the changes in the code, as required in the task. If the code is correct, the next task becomes available, if not, the task is marked as “failed” and retry is needed.

When the Check button is not available, the code should be tested by using the Run button. Usually when running the code, the player is asked to get the monkey to the banana.

Bonus Tasks

Encourage your students to complete these as well. Bonus tasks are more complicated and most of them are not step by step instructions like . in the tasks within the exercises.

Some bonus tasks include topics that were covered in previous exercises.

Part 2 – Let's Go!

Play Time 10 min.

Ask your students to complete exercises 29 - 32.

Encourage your students to also read the instructions in each exercise, especially if they do not understand what needs to be done.

Use your classroom's dashboard to keep track of student achievements.

Part 2 – Let's Go!

Walkthrough 5 min.

Open exercise #31:

In this exercise, we will duplicate sprites. This functionality is very useful when we want to have many occurrences of the same sprite with the same functionality.

Ask your students to provide examples of sprites we may want to duplicate:

- bananas
- chocolates

In our game, we want the banana to get destroyed each time the monkey collides with it and for points to be added to the score. Also, when the monkey collides with a chocolate sprite, we want the chocolate sprite to get destroyed and for points to be subtracted from the score.

If we write the onCollide function in the monkey's sprite, each time we duplicate the sprite, we will still need to add another onCollide function with a different sprite name (i.e. banana1, banana2).

When we write the onCollide function in the banana's sprite, each time we duplicate the sprite, the code for the onCollide with the monkey will already exist and we will not need to add additional code.

To duplicate a sprite, open the sprite's settings and click on the duplicate icon 

Part 2 – Let's Go!

Play Time 5 min.

Students should complete up to exercise 32.

Use this time to pass between your students and help them.

Encourage students who complete all exercises to solve also the bonus task in exercises #30 and #32.

PART 3

Debriefing

5 min.

Part 3 – Debriefing

Discussion 5 min.

In the previous lesson, we added two timers - one to make the monkey bigger and one to make the monkey smaller. When one timer ended, the other timer started, over and over again. You can use this link to remind your students.

Now, create the same functionality (every 2 seconds the monkey changes its size) but using only one timer.

The Timer widget can run in a loop. Meaning, each time the time ends, it starts to run again. The function onEnd is still called each time the timer ends.

The syntax to start a timer in a loop is:

```
@start 5, yes
```

You can use the following link as an example.

Part 3 – Debriefing

Discussion 5 min.

One way to solve this with just one timer is to define a property of the timer.

This property will indicate if the monkey's scale is big or small and change the monkey's scale accordingly. Also, the property needs to be updated each time.

The timer's code:

```
@isBig = yes  # initialize the property to yes, as the monkey is bigger at the beginning

@start 2, yes  # starts the timer to run in a loop

@onEnd = () =>

    if @isBig  # when isBig is yes, it means that the monkey is big and needs to be changed to small

        monkey.setScale 0.5

        @isBig = no  # monkey will now be smaller, so isBig needs to be updated to no

    else  # when isBig is no, it means that the monkey is small and needs to be changed to big

        monkey.setScale 1.5

        @isBig = yes  # monkey will now be bigger, so isBig needs to be updated to yes
```

TEACHER's GUIDE

LESSON Nº 7

When it ends...

Exercises 33 - 35

PART 1

Introduction

5 min.

This lesson introduces the Dialog widget.

Your students will use it to display the score and a message when the game is over.

Students will also learn to reset the game.

Objectives:

Within this lesson, students will:

- Use a Dialog widget to display a message
- Reset the game

Components:

- Dialog widget
- “game.reset”

Part 1 - Introduction

Explain 5 min.

The Dialog widget has a property called text, which holds the text that is displayed to the user.

The Dialog widget is used to display messages during the game. For example, before the game starts or when it ends.

The text can be updated from two places:

1. the widget's settings
2. the code of any sprite/widget

The Dialog widget has an event function onConfirm, which is called once the button is clicked.

Part 1 - Introduction

Explain (con't) 5 min.

Sometimes, we want to show text and include a parameter inside of it.

For example:

- show the score
- show how many bananas were collected

The syntax to add a parameter to the text is:

```
dialog.text = "You scored: #{score.value} points!"
```

When the code runs, the computer replaces the variable with its value. For example, if we set the text of the dialog to be:

```
dialog.text = "You scored: #{score.value} points!"
```

When the dialog is shown, given score.value is 300, then the message will be:

You scored: 300 points!

PART 2

Let’s Go!

15 min.

Course Structure:
The Course is divided into exercises. Each exercise includes tasks that need to be completed. Upon successful completion of an exercise, the next exercise becomes available.

Each task ends with clicking the “check” button, after making the changes in the code, as required in the task. If the code is correct, the next task becomes available, if not, the task is marked as “failed” and retry is needed.

When the Check button is not available, the code should be tested by using the Run button. Usually when running the code, the player is asked to get the monkey to the banana.

Bonus Tasks

Encourage your students to complete these as well. Bonus tasks are more complicated and most of them are not step by step instructions like . in the tasks within the exercises.

Some bonus tasks include topics that were covered in previous exercises.

Part 2 – Let's Go!

Play Time 15 min.

Ask your students to complete exercises 33 - 35.

Encourage your students to also read the instructions in each exercise, especially if they do not understand what needs to be done.

Use your classroom's dashboard to keep track of student achievements.

Encourage students who complete all exercises to solve also the bonus task in exercise #33.

PART 3

Debriefing

25 min.

Part 3 – Debriefing

Discussion 25 min.

There is another widget that was not used in the exercises of this course. It is the Clock widget.

Here, we will explain this widget and then use it to show how many seconds it takes to collect all the bananas.

Let us start with the Clock widget.

The Clock widget is similar to the Timer widget. The difference is that the Clock widget starts from 0 and counts how many seconds have passed, whereas the Timer widget counts backwards the seconds until 0.

The Clock widget has a function called start, this function does not need arguments.

```
clock.start()
```

Part 3 – Debriefing

Discussion 25 min.

[Open Exercise #30]

Go to the Widgets tab and add a Clock widget.

Start the clock. This is the default code for the Clock widget, so we actually do not need to add anything here.

The clock's code:

```
@start()
```

Run the code and show your students that the clock starts running.

Ask your students what will happen if we also start the clock in both of the banana sprites in their onCollide function.

The answer is that each time the onCollide function is called (the monkey gets a banana) the clock will start counting from 0.

Part 3 – Debriefing

Discussion 25 min.

The banana's code:

```
@onCollide monkey, () =>

    clock.start()

    @destroy()

      …
```

Run the code and show how each time the monkey gets a banana, the clock starts running from 0 again.

The clock has another function getSeconds. This function, which does not get an argument, returns the number of seconds that have passed since the clock started running.

We can use the operator <, > or == in an if statement to check how many seconds have passed and do something based on them.

Part 3 – Debriefing

Discussion 25 min.

Ask your students to give you examples:

- if the monkey gets all the bananas in less than x seconds, the player gets bonus points
  - add y points to score
- if the monkey gets all the bananas in more than x seconds the player “pays” a penalty
  - reduce z points from score

Now, we want to give the user a message of how long it took to collect all the bananas. Which widgets should we add?

- Counter widget - to count how many bananas were collected
  - in each banana, add an if statement to check whether all the bananas were collected
- Clock widget - to know how many seconds have passed
- Dialog widget - to show the number of seconds to the user

Part 3 – Debriefing

Discussion 25 min.

Discuss with your students what will be in the banana's code:

```
@onCollide monkey, () =>
    @destroy()
    collectBanana.play()
    score.value+=500
    bananaCount.value += 1
    if bananaCount.value == 5
        dialog.text = "You did it in : #{clock.getSeconds()} seconds!"
        dialog.show()
```

This code will show how many seconds have passed after getting 5 bananas.

Tell your students that now we want to display two messages depending on the number of seconds. Discuss with them what needs to be added to the code.

Part 3 – Debriefing

Discussion 25 min.

```
@onCollide monkey, () =>
    @destroy()
    collectBanana.play()
    score.value+=500
    bananaCount.value += 1
    if bananaCount.value == 5
        sec = clock.getSeconds()
        if sec < 7
            dialog.text = "Wow, you did it in : #{sec} seconds!"
        else
            dialog.text = "Not bad, it took you : #{sec} seconds!"
        dialog.show()
```

This code will display a different message depending if it took less or more than 7 seconds to collect 5 bananas.

Part 3 – Debriefing

Discussion 25 min.

Now, ask your students what will happen if:

- instead of 5 bananas, there are 10 bananas
- instead of less than 7 seconds, we changed it to 15 seconds

The code needs to be changed in each of the banana's sprites. What could be a better solution?

- Use a function!
  - define one function in the monkey's sprite
  - move all the new functionalities to the new function
  - in each banana, call the new function

You can use the following link as an example or use exercise #31.

Part 3 – Debriefing

Discussion 25 min.

The new function in the monkey's sprite:

```
@handleBanana = () =>
    collectBanana.play()
    score.value += 500
    bananaCount.value += 1
    if bananaCount.value == 10
        sec = clock.getSeconds()
        if sec < 15
            dialog.text = "Wow, you did it in : #{Math.round(sec)} seconds!"
        else
            dialog.text = "Not bad, it took you : #{Math.round(sec)} seconds!"
        dialog.show()
```

The use of Math.round() returns the rounded number of the variable sec.

Part 3 – Debriefing

Discussion 25 min.

The code in each of the banana's sprites:

```
@onCollide monkey, () =>

    @destroy()

    monkey.handleBanana()
```

Discuss with your students why calling the function destroy is not part of the handleBanana function and remains in the banana sprite

- to destroy a banana in the monkey's sprite we need to add the banana's name before the function destroy
- in the monkey's sprite, we do not know which banana we need to destroy

Start with 5 bananas, then add more bananas and show your students there is only one place in the code that needs to be changed (in the function handleBanana).

You can also change the number of seconds to display which message. This change is also needed in only one place.

GREAT JOB!