Sprite Animation 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 Draw

Exercises 1-5

PART 1

Introduction

20 min.

The objective of the Sprite Animations course is to teach students how to create a new sprite sheet and add an animation to it. The course also introduces several basic programming concepts.

The goal of the player is to “eat” as many apples as possible.

Objectives:

Within this lesson, students will:

- Draw a new sprite sheet with three frames
- Add a sprite
- Add animation and start running it

Components:

- Create a sprite sheet
- Add a sprite
- “addAnimation”, “startAnimation”

Part 1 - Introduction

Watch 5 min.

- Open the following link and show your students the game that they will create at the end of this course.
- Ask a volunteer to play the game.
- Ask your students to think of the different sprites they see in the game.
- Tell your students that they will draw the sprites in this game and also define the animations for each of the sprites.

Part 1 - Introduction

Explain #1 10 min.

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

The list should include the following:

- the phrase “Ready, Set, Go”
- the apples
- the message “Game over”

It might be hard for them to point out that the phrase and the message are also sprites.

Part 1 - Introduction

Explain (con't) 10 min.

Discuss with your students what a sprite is.

A sprite is an image in the game.

Show your students the following game.

In this game there are a few sprites:

- monkey
- banana
- tiger
- chocolate
- power up

Some of these sprites change their images and some remain the same.

Part 1 - Introduction

Explain (con't) 10 min.

Run the game and ask your students which sprites change their image:

- monkey
- tiger

It looks as if they are moving.

This is called animation.

Click on Create Games to create a new game.

Add to the game the sprites: monkey, banana and tiger.

- Go to the Sprites tab
- Click on the green + button
- Choose a different sprite each time to add to the game

Part 1 - Introduction

Explain (con't) 10 min.

Each sprite has one or more frames. A sprite sheet consists of the all the frames of the sprite.

For each of the sprites, click on the settings icon ( ) of each sprite, then click on Show preview.

The Show preview option shows all the frames of the sprite sheet.

Show your students that the monkey and tiger have five frames each. The banana only has one frame.

The function step moves a sprite.

Add the following code to each of the sprites (like this for example):

```
@step 100
```

Show how the monkey and tiger look like they are stepping, whereas the banana does not. Run it a few times to emphasize the difference.

Part 1 - Introduction

Explain #2 5 min.

This course starts with drawing one sprite sheet with three frames and adding the sprite to the game.

Then, an animation is defined and starts running when playing the game.

When we want the sprite to appear as if it is moving, we need to define an animation.

Each animation must have the following:

1. name
    - a sprite can have more than one animation
    - name is unique per each sprite
2. order of frames
    - not all frames must be used in the animation
    - each animation can have a different order of frames

Part 1 - Introduction

Explain (con't) 5 min.

The syntax to define an animation is:

```
@addAnimation "key", [0, 1, 2], frameRate, loopAnimation
```

The parameters frameRate and loopAnimation are explained and used later in the course, so for now, we will just delete them.

We will use the syntax:

```
@addAnimation "key", [0, 1, 2]
```

- key - is the name of the animation
  - this name will be used to start an animation or to do something when it ends
- [0, 1, 2] - this is an array of the frames
  - the first frame is always 0
  - this is an example of an animation with three frames
    - the sprite sheet could have more frames or not

Part 1 - Introduction

Explain (con't) 5 min.

Defining an animation is not enough. We need to start running the animation.

The syntax to start an animation is:

```
@startAnimation "key"
```

- the key is replaced with the name of the animation

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 is 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 a retry is needed.

When the Check button is not available, the code should be tested by using the Run button.

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 as 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 needs to be done.

Use your classroom dashboard to keep track of student achievements.

Part 2 – Let's Go!

Walkthrough 5 min.

Open Exercise #1:

Go over the steps to create a sprite sheet, add new frames and add the sprite to the game.

Create a new sprite sheet:

1. Go to the Sprites tab
    - lower right side
2. Click on the green + button
    - the Add a Sprite screen is open
3. Click on You Spritesheet tab
    - it displays all the sprite sheets that you created or uploaded
4. Click on Create a new sheet
    - the Create a sprite sheet screen is open
5. Click OK
    - there is no need to change the width or height

This is the first frame of the sprite sheet. Draw or write the first frame. You can use the icon Aa to insert a text box, or you can use the pencil for free drawing.

Part 2 – Let's Go!

Walkthrough (con't) 5 min.

Create more frames:

There are two options to create new frames:

- click on the green + button
  - this will create a new empty frame
- click on the duplicate icon  (on the bottom left corner of the frame)
  - this will duplicate the frame
  - usually, the changes between the frames is very minor so using the duplicate option offers better accuracy
    - duplicate the frame, then add or remove parts to change the frame

Once all the frames are created, click on the save button. Your sprite sheet is now added to the Add a sprite screen.

Add the sprite to the game by choosing the sprite sheet and clicking on the Add button.

The sprite is added the game.

Part 2 – Let's Go!

Walkthrough (con't) 5 min.

When a sprite is added, a default name that is the same as the sprite is given (i.e. banana when adding a banana sprite). When adding a sprite sheet you created, the default name is mySprite.

If there is already a sprite with that name, then a number will be added as well (mySprite1, mySprite2, etc.).

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

To edit the sprite sheet after it is saved do the following:

- From the sprite's settings icon, click on Edit Sprite Sheet
- Add, delete or change the relevant frame and click on save

Part 2 – Let's Go!

Play Time 5 min.

Students should complete up to exercise 5.

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

Encourage students who completed all of exercises to solve the bonus task in exercise #5.

PART 3

Debriefing

5 min.

Part 3 – Debriefing

Discussion 5 min.

Discuss with your students what was covered in this lesson:

1. What is a sprite?
   - an image in the game
2. What does a sprite sheet contain?
   - the frames of the sprite
3. What are these frames used for?
   - adding an animation to create the illusion that the sprite is moving
4. Name two functions that we used for the animation:
   - addAnimation - to define an animation
   - startAnimation - to start running an animation
5. Can we only define one animation for each sprite?
   - no, we can have as many as we want - each with a different name
6. Can two different sprites have the same animation definition?
   - yes, each animation is defined per sprite

TEACHER's GUIDE

LESSON Nº 2

Let's "eat"

Exercises 6-10

PART 1

Introduction

15 min.

In this lesson, your students will draw another sprite sheet and add it to the game.

This lesson introduces event handling - what happens when clicking on a sprite.

Your students will learn about another parameter of the addAnimation function - the frame rate. They will also deepen their understanding of sprite sheet frames.

Your students will add a sound to the game and play it.

Objectives:

Within this lesson, students will:

- Define an event handler for clicking on a sprite
- Add a sound and play it
- Learn what frame rate is and change it

Components:

- “onClick”
- “play”
- frame rate

Part 1 - Introduction

Activity #1 5 min.

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

Ask for four volunteers:

- Nominate one volunteer to be the “events causer”
- Nominate the other three volunteers to respond to an event (act as the “event handlers”)

Decide with the “event handlers” on their actions (each should have one action). For example:

1. Stepping to the window
2. Writing “Hello” on the board
3. Counting until ten in Spanish (of course, they can count in English too)
4. Jumping four times forward and three times backwards

Tell the “event handlers” that they act only when the “event causer” taps on their shoulders.

Part 1 - Introduction

Activity #1 (con't) 5 min.

Ask the "event causer" to start going around and tap the "event handlers" shoulders.

Each "event handler" should perform his action only when the "event causer" taps their shoulder.

Ask the "event causer" to also:

- tap on other students in the class
  - they should not do anything
  - ask them why they are not doing anything?
    - because they were not programmed to do anything
- tap on other objects in the class (i.e. a chair or desk)
  - nothing will happen
  - ask your students why is nothing happening?
    - because these objects do not have the capability of being handled in an event

Part 1 - Introduction

Explain #1 4 min.

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

Computer events are:

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

When we click on the mouse, we trigger the mouse click event and the onClick function is called. Such a function is also called an event handler.

The onClick function has no parameters. Anytime the mouse is clicked on, the function onClick is called. If the function is not defined in the sprite, then nothing will happen when the sprite is clicked on.

When the mouse is clicked on anything other than a sprite, nothing will happen, because there is no object that listens (“wait”) for the trigger.

Part 1 - Introduction

Explain #2 6 min.

In the previous lesson we drew a sprite sheet with three frames. Each frame had an index according to the order that they appeared in the sprite sheet. The index of the first array was 0, then 1 and so on.

An array is defined using the square brackets [].

When we define an animation we use an array to set which frames will be included in the animation and their order of the appearance.

- there might be a case when we do not want to include all of the frames
- there might be a case when we want to run the animation with a different order of frames than what was drawn in the sprite sheet

The array sets the frames (based on their indexes) and their order.

Part 1 - Introduction

Explain (con't) 6 min.

When defining an animation, we can also define how fast or slow the animation will run. This is called the frame rate.

The frame rate defines how many frames will run per second. A higher frame rate means that the animation will run faster.

In the following example:

```
@addAnimation "test1", [0, 1, 2, 3], 4
```

The frame rate is four, which means that every second, four frames will run. Since there are four frames, the animation will run for one second.

```
@addAnimation "test2", [0, 1, 2, 3, 4, 5, 6, 7], 4
```

What about the second example? How long will it run for?

It will run two seconds, because each second, four frames will run and there are eight frames all together.

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 is 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 a retry is needed.

When the Check button is not available, the code should be tested by using the Run button.

From now on, usually when running the code, the player is asked to click on an apple.

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 as 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 - 10.

Encourage them to 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 student 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, play, which plays the sound.

Open Exercise #9

Go to the Sounds tab.

This tab shows the sounds we have in the game. When we press on the + green button we see the different sounds that we can add to the game. Show your students that we can play the sounds before adding them by clicking on the blue triangle.

Ask your students when they think we can add sounds. For example:

- When an apple is “eaten”
- When the game ends

Part 2 – Let's Go!

Play Time 5 min.

Students should complete up to exercise 10.

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 #8.

PART 3

Debriefing

10 min.

Part 3 – Debriefing

Discussion 10

Discuss with your students what we covered in this lesson:

1. When defining the following animations, which will run longer?
   - `@addAnimation "test1", [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 5`
   - `@addAnimation "test2", [0, 1, 2, 3, 4], 5`
   - The animation “test1” will run longer because it has ten frames, its frame rate is five, so it will run for two seconds. The animation “test2” will run for just one second.
2. How can we make both animations run the same time?
   - One option - change the frame rate of “test1” to 10
     - then both animations will run for one second
   - Other option - change the frame rate of “test2” to 2.5
     - then both animations will run for two seconds

Draw a new sprite sheet with ten frames, each can be a number between 0 - 9. Add two sprites from this sprite sheet and then define and run an animation in each of the sprites. Change the frame rate accordingly to show both of the above.

You can use this link as an example.

Part 3 – Debriefing

Discussion 10

The code in one sprite:

```
@addAnimation "test1", [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 5

@startAnimation "test1"
```

The code in the second sprite:

```
@addAnimation "test2", [0, 1, 2, 3, 4], 5

@startAnimation "test2"
```

TEACHER's GUIDE

LESSON Nº 3

Let's End This

Exercises 11-15

PART 1

Introduction

15 min.

This lesson introduces another event handler - when an animation ends. Your students will learn how to stop an animation.

Your students will deepen their understanding of the order of frames in the sprite sheet.

Your students will learn to duplicate a sprite.

Objectives:

Within this lesson, students will:

- Write an event handler for when an animation ends
- Stop an animation
- Further learn about the order of the frames
- Duplicate a sprite

Components:

- “onAnimationEnd”
- “stopAnimation”
- duplicate a sprite

Part 1 - Introduction

Review 2 min.

Recall the topic of event handling that was covered in the previous lesson.

- what is an event handler?
  - actions triggered by the user that take place while the code is running
- what happens if we do not define an event handler function?
  - nothing will happen
- what happens if we define an event handler function but the player does not trigger the event?
  - for example, if we define an onClick function for a sprite, but the player does not click on the sprite.
    - nothing will happen because no activity will trigger the function onClick

Part 1 - Introduction

Review (con't) 2 min.

Discuss which event handling your students are familiar with. Their answers may include:

- Mouse click event
- Keyboard event
- Mouse event
- Swipe event

We defined and used the onClick function for a sprite which is called each time the mouse is clicked on the sprite.

Part 1 - Introduction

Activity #1 5 min.

The following activity facilitates the understanding of an event and an event handler, specifically an event that occurs when another action ends.

Ask for two volunteers:

- Nominate one volunteer to be the "event causer"
- Nominate the other volunteer to respond to an event (be the "event handler")

The "event causer" will read sentences and bow after each one.

Tell the "event handler" to clap his hands each time the "event causer" bows.

Ask the "event "causer" to start reading the sentences.

The "event handler" should clap his hands after each bow.

Ask the "event causer" to complete reading sentences without bowing. In this case, the "event handler" should not clap his hands.

Part 1 - Introduction

Activity (con't) 5 min.

Ask your students:

- what triggered the "event handler"?
  - the bow activity
- why did the "event handler" not clap his hands when the "event causer" did not bow?
  - because the activity that triggers him did not occur

Part 1 - Introduction

Explain #1 4 min.

In today's lesson we will learn about another event handler.

- An event that occurs when an animation ends.

Each time an animation ends, the function onAnimationEnd is called.

When we want something to happen each time an animation ends, we will define this function.

The syntax to define this function is:

```
@onAnimationEnd "key", () =>
```

The parameter key is the name of the animation.

This function is defined per animation. If we have two animations for the same sprite, then we will need to define two onAnimationEnd functions (assuming we want something to happen when both ends.)

Part 1 - Introduction

Explain (con't) 4 min.

Open exercise #10:

An example for something to happen when an animation ends is to play a sound.

Add a sound to the game.

In the apple's sprite, add the following code:

```
@onAnimationEnd "eat", () =>

    mySound.play()
```

Discuss with your students other examples for something to happen when the animation ends:

- hide the sprite
- change the sprite's position
- start another animation

Part 1 - Introduction

Explain #2 4 min.

Run the game (you can continue with exercise # 10).

Ask your students: what do you notice happens to the the apple when the animation ends?

- the last frame of the eaten apple is displayed

Run the game again to emphasize this.

Click on the apple a few times (after the animation ends), each time the animation starts with the first frame and stays on the eaten apple.

There is an option to reset the frames of the animation.

The function stopAnimation stops the animation. The function has one parameter which defines if to reset the animation or not:

- yes - reset the animation to the first frame
- no - do not reset

Part 1 - Introduction

Explain (con't) 4 min.

Change the code to reset the apple's sprite after the animation ends:

```
@onAnimationEnd "eat", () =>
    @stopAnimation yes
```

Run the code and show that after the animation ends, the apple is whole again.

Change the parameter to 'no' and click run again.

Actually, there is no point in calling the stopAnimation when the animation ends with parameter set to 'no' because it does not do anything.

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 is 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 Check button is not available, the code should be tested by using the Run button. From now on, usually when running the code, the player is asked to click on an apple.

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 as 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 11 - 15.

Encourage them 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 student achievements.

Part 2 – Let's Go!

Walkthrough 5 min.

Open exercise #15:

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 give you examples of sprites we could want to duplicate:

- apples to click on

In our game, we want to have more than one apple to click on to make the game more fun.

Each time we duplicate the sprite, the code is duplicated as well. After we finish writing all of the sprite's code, we will duplicate it. The code is included in the duplicated sprite.

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 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

10 min.

Part 3 – Debriefing

Discussion 10 min.

Discuss what was covered in this lesson:

1. When is the function onAnimationEnd called?
    - Each time the animation ends
2. Is it called when the any animation ends?
    - No, this function is defined per animation and sprite
3. Define an animation, start it by clicking on the sprite and then duplicate the sprite. What will happen when we click on the duplicated sprite?
    - The animation will run for that sprite
4. Will both animations run together?
    - No, each animation will start running only when we click on the sprite

Use the sprite sheet from the previous lesson (10 frames, each with a number between 0 - 9). Add a click event to stop the animation. First, do not reset the frames, then change it to run with resetting the frames.

You can use this link as an example.

Part 3 – Debriefing

Discussion 10 min.

Run the game, click on the sprite on the right (the one with numbers 0 until 9) and see that the animation stops before the animation ended.

Your code should be:

```
@addAnimation "test1", [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 5

@startAnimation "test1"

@onClick = () =>

    @stopAnimation no
```

You can also add a click event to the other sprite and click to stop it too.

TEACHER's GUIDE

LESSON Nº 4

Let's Count

Exercises 16-21

PART 1

Introduction

20 min.

This lesson will introduce two new widgets. The Counter widget, which will be used to count the number of apples clicked on, and the Timer widget, which will be used to time the game.

Your students will change the sprite's position randomly.

Objectives:

Within this lesson, students will:

- Use a Counter widget
- Change a sprite's position randomly when the animation ends
- Use a Timer widget and destroy the sprite when the time ends

Components:

- Counter widget
- "setX", "setY", "random"
- Timer widget
- "destroy"

Part 1 - Introduction

Explain #1 5 min.

The "world" in the game has two coordinates: x and y. When placing a sprite in the game, its x and y positions are set.

Open Exercise #17:

Click the apple's settings icon (⚙). Show the value of x and y.

Show the x and y coordinates of the mouse. Move the mouse onto the world and show the coordinates in the top left corner.

Part 1 - Introduction

Explain (con't) 5 min.

Show how at the top left corner, the x and y axes are both 0. Move the mouse to the bottom right-hand corner to demonstrate that the x value at this corner is 600 and the y is 400 (notice that it is very difficult to get the mouse to the exact corners, so the numbers you will show will be close to 0,0 or 600,400).

The world's size in the game is 0-600 horizontally and 0-400 vertically.

The following two functions are used to change the position of the sprite during the game:

```
@setX
```

```
@setY
```

These functions change the x and y position of the sprite it refers to.

Part 1 - Introduction

Explain (con't) 5 min.

The position of the sprite (its x and y position) are properties of the sprite.

The sprite's settings include properties that can be set before the game starts.

There are three options regarding changing the sprite's properties. These can be changed from:

1. the sprite's settings only
2. the code only
3. both the code and the sprite's settings

Part 1 - Introduction

Explain (con't) 5 min.

Discuss with your students examples of each of the options:

1. the sprite's settings only
    - Allow gravity - defines if the sprite is affected by the gravity of the game
    - Collide with world bound - defines if the sprite can step outside the world boundaries, or if it gets stuck
2. the code only
    - Sprite's speed - setSpeed
    - Sprite's scale - setScale
3. both the code and the sprite's settings
    - X position - setX; X
    - Y position - setY; X
    - Show or hide the sprite - show()/hide(); Show
    - Rotation of the sprite - setRotation; Rotation

Part 1 - Introduction

Explain #2 5 min.

Open exercise #17:

Add another sprite to the game, for example, the banana sprite.

Show your students that when clicking on a sprite, the sprite's name (and code) is changed in the editor.

When a function is called, the name of the sprite needs to be added

in order to know onto which sprite to apply the function.

For example, in this exercise, there are a few functions in the apple's code: addAnimation, onClick, onAnimationEnd, stopAnimation.

Before each function, the @ character 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 onClick function from the apple sprite to the banana sprite.

Ask your students what will happen now when you run the game:

- if you click on the apple?
  - nothing will happen because the onClick is not defined for the apple
- if you click on the banana?
  - the bite sound will be played
  - the apple animation will not start because in the banana's code the @ refers to the banana and not to the apple.

Run the game and click both on the apple and the banana to demonstrate the above.

Part 1 - Introduction

Explain (con't) 5 min.

The code in the banana's sprite:

```
@onClick = () =>
    @startAnimation "eat"
    bite.play()
```

The code in the apple's sprite:

```
@addAnimation "eat", [0, 1, 2, 3], 2
@onAnimationEnd "eat", () =>
    @stopAnimation yes
    counter.value += 1
```

Ask your students what needs to be changed in the **banana** sprite only so that the "eat" animation will start?

- The sprite's name needs to written instead of the @.

Change the code and run the game again to show the animation starts when clicking on the banana. The updated code will be:

```
@onClick = () =>
    apple.startAnimation "eat"
    bite.play()
```

Part 1 - Introduction

Explain #3 10 min.

Time to explore the **Widgets** tab.

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

Example of widgets: Counter, Text, Timer, Clock, Dialog, Button.

The Widgets do not have many properties. All of them have the Show property - we can decide if we want to show it or not.

Each widget also has unique properties.

In today’s lesson we will learn about the Counter and Timer widgets.

Part 1 - Introduction

Explain (con't) 10 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 might want to count?

- score
- lives
- collected items

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

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

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

```
count.value
```

Part 1 - Introduction

Explain (con't) 10 min.

Usually, we will want to increase the value of the counter. For example, we will want to increase it by 1 to count the number of times we clicked on the apple.

We use the operator **+=** to increase a value.

This operator adds the number on the right to the variable on the left.

For example:

```
var += 5
```

This adds 5 to the variable var. If its original value was 0, then it is now 5. If its original value was 10, it is now 15.

In this example, we will use:

```
count.value += 1
```

Part 1 - Introduction

Explain (con't) 10 min.

Open exercise #20:

This is the first time we will use a Timer widget.

Whenever 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 run for 5 seconds, we will code:

```
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 running in seconds once we run the game.

Part 1 - Introduction

Explain (con't) 10 min.

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 not inside a function, the timer will also start running when we run the game.

The Timer widget has another function. This is an event function which 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 Widget tab and add a Timer. Show your students that the function onEnd is defined there.

```
@onEnd = () =>

    # Your code here
```

If we do not write any code inside this function, then nothing will happen when the timer ends.

Part 1 - Introduction

Explain (con't) 10 min.

Discuss with your students what should be in the code so that when the time ends, the animation will stop.

The function that stops the animation - stopAnimation should be called from the onEnd function.

The code will be:

```
@start 1.5

@onEnd = () =>

      apple.stopAnimation no
```

Run the code, click on the apple and see that the animation stops when the timer ends and before the animation ends.

Emphasize that we need to write apple.stopAnimation because we are **not** writing in the apple's sprite!

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 is 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 a retry is needed.

When the Check button is not available, the code should be tested by using the Run button.

From now on, usually when running the code, the player is asked to click on an apple.

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 as 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 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 student achievements.

Part 2 – Let's Go!

Walkthrough 5 min.

Open exercise #18:

In this exercise, we will use the random function.

This function gets two parameters:

- minimum value
- maximum value

The function returns a random number between the minimum and the maximum.

The syntax of the function is:

```
random minimum, maximum
```

For example, if we write the following code

```
random_var = random 0, 5
```

Then each run random_var will be a different number between 0 and 5.

Part 2 – Let's Go!

Play Time 5 min.

Students should complete up to exercise 21.

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

Encourage students who complete all of the exercises to also solve the bonus task in exercises #17 and #19.

PART 3

Debriefing

5 min.

Part 3 – Debriefing

Discussion 5 min.

Discuss with your class what we have covered in this lesson:

1. When an animation starts by clicking on the sprite, how can we make sure it will run only once?
   - destroy the sprite when the animation ends
2. What if we wanted to run the animation only 5 times?
   - use a counter
   - when the animation ends:
     - increase the counter by 1
     - check if the value of the counter is 5, if it is, then destroy the sprite

You can use this link as an example.

```
@onAnimationEnd "test1", () =>

    @destroy()
```

```
@onAnimationEnd "test2", () =>

    count.value += 1

    if count.value == 5

        @destroy()
```

Part 3 – Debriefing

Discussion 5 min.

Now, tell your students that we want:

- two sprites, each with their own animation
- animation starts by clicking on one sprite
- count how many times the sprite is clicked on in x seconds
  - do not let the user click on the sprite after the time is up
- when the time is up, count how many times the second sprite is clicked on in x seconds
  - again, do not let the user click on the sprite after the time is up

Discuss with your students what needs to be coded:

- in each sprite:
  - define an animation
  - define onClick to start the animation
  - define onAnimationEnd to add 1 to a counter
- add 2 counters
  - one for each of the sprites

Part 3 – Debriefing

Discussion 5 min.

- add 2 timers
  - start the first one when the running the game
  - when it ends:
    - destroy the first sprite
    - start the second timer
  - when the second timer ends:
    - destroy the second sprite

You can use this link as an example.

Part 3 – Debriefing

Discussion 5 min.

Sprite & timer #1:

```
# Sprite’s code:
@addAnimation "test1", [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 10
@onClick = () =>
    @startAnimation "test1"
@onAnimationEnd "test1", () =>
    counter1.value += 1

# Timer’s code:
@start 8
@onEnd = () =>
     mySprite1.destroy()
     timer2.start 8
```

Sprite & timer #2:

```
# Sprite’s code:
@addAnimation "test2", [0, 1, 2, 3, 4], 5
@onClick = () =>
    @startAnimation "test2"
@onAnimationEnd "test2", () =>
    counter2.value += 1

# Timer’s code:
@onEnd = () =>
     mySprite2.destroy()
```

TEACHER's GUIDE

LESSON Nº 5

Let's Code

Exercises 22-26

PART 1

Introduction

20 min.

The following lesson introduces a few important coding topics:

- array - using an array of apple sprites
- for loop - using a for loop to iterate on an array
- function - using a function to start the game

Your students will also define an animation that will run in a loop over and over again.

Objectives:

Within this lesson, students will:

- Use an array to manipulate all apple sprites together
- Use a for loop to iterate on the array
- Use a function to bind all instructions to start the game
- Define an animation that will run in a loop

Components:

- array, for, function
- loop animation

Part 1 - Introduction

Activity 5 min.

The following activity facilitates the understanding of an array and a for loop.

Ask for a volunteer:

Write on the board a few instructions for the volunteer to perform:

1. step 5 steps
2. turn 180
3. step 3 steps
4. jump
5. step backwards 4 steps

Ask the volunteer to perform the instructions.

Now ask for five more volunteers. Name them v0, v1, v2, v3 and v4.

Part 1 - Introduction

Activity (con't) 5 min.

Write on the board the set of instructions for each of the volunteers:

v0:

1. step 5 steps
2. turn 180
3. step 3 steps
4. jump
5. step backwards 4 steps

v1:

1. step 5 steps
2. turn 180
3. step 3 steps
4. jump
5. step backwards 4 steps

v2:

1. step 5 steps
2. turn 180
3. step 3 steps
4. jump
5. step backwards 4 steps

v3:

1. step 5 steps
2. turn 180
3. step 3 steps
4. jump
5. step backwards 4 steps

v4:

1. step 5 steps
2. turn 180
3. step 3 steps
4. jump
5. step backwards 4 steps

Part 1 - Introduction

Activity (con't) 5 min.

Make sure the volunteers do not stand according to their numbers. For example, v0 next to v3 and then v1 and so on.

Tell them that when you call their names, each need to look at the board and perform the instructions.

Call out their names in order - start with v0, when he finishes all five instructions move to the next one and so on.

- the order is v0, v1, v2, v3, v4

Part 1 - Introduction

Explain #1 5 min.

Discuss with your students how to make the code presented in the example efficient.

1. define an array of volunteers
2. use a for loop on the array

Write the new code on the board:

```
volunteers = [v0, v1, v2, v3, v4]

for v in volunteers

    v.step 5

    v.turn 180

    v.step 3

    v.jump()

    v.step -4
```

Part 1 - Introduction

Explain (con't) 5 min.

Ask your students:

1. what is the name of the array?
   - volunteers
2. how many times will the set of instructions be repeated?
   - five times
3. what will be the order of the volunteers?
   - v0, v1, v2, v3, v4
4. how many lines of code do we need to change if instead of turn 180 we want all the volunteers to turn left?
   - one line
5. what do we need to change if we want to start from v4, then v3 and so on until v0?
   - change the definition of the array to:

```
volunteers = [v4, v3, v2, v1, v0]
```

Part 1 - Introduction

Explain (con't) 5 min.

An array is a collection of items in a specific order.

In our game, we will define an array with sprites' name as its items.

Each item in the array has an index. The index starts from 0 until n-1 (for n items in an array).

The array is defined using the square brackets [].

When referring to an item in an array, we will write the name of the array and its index.

For example, to refer to items in the array volunteers, we will write:

- first item - volunteers[0]
- second item - volunteers[1]
- third item - volunteers[2]
- fifth item - volunteers[4]

Part 1 - Introduction

Explain (con't) 5 min.

Write the following array on the board:

```
numbers = [5, 2, 6, 7, 0, 9]
```

Ask your students:

- what is the name of the array?
  - numbers
- how many items are in the array?
  - six items
- what is the index of the last item, what is its value?
  - 5
  - numbers[5] = 9
- what is the value of numbers[2]?
  - 6
- what is the index of the item whose value is 0? How do we refer to it?
  - 4, numbers[4]

Part 1 - Introduction

Explain #2 5 min.

Discuss with your students:

- What is a loop:
  - a loop repeats a sequence of instructions until a certain condition is met
- when to use a loop:
  - when the same set of code needs to be repeated a number of times
    - sometimes, when the number of times is unknown, a variable can be used instead
- in the activity, the code was written five times on the board. How could the code be shorter?
  - use a loop!

Part 1 - Introduction

Explain (con't) 5 min.

Discuss with your students what kind of loops they are familiar with:

- times
  - repeats for x number of times
  - stops after the x^{th} time
- until
  - repeats until the condition is met
  - stops when the condition becomes a true statement
- while
  - repeats as long as the condition is met
  - stops when the condition becomes a false statement
- loop
  - repeats for as long as the code runs
  - never stops
- for
  - repeats per the numbers of items in the array

Part 1 - Introduction

Explain (con't) 5 min.

In today's lesson, we will focus on for loops.

A for loop is used when we have an array and want to perform the same actions on all the items in the array. The code will be executed in the order of the items in the array.

The syntax of a for loop:

```
for name in array
```

- array - the array the for loop iterates on
- name - is the loop's variable
  - when the code runs, it replaces the variable with the item from the array

A for loop supports any array. If the array changes, the code does not need to change.

Part 1 - Introduction

Explain (con't) 5 min.

Loop on the array called numbers. For each item, step its value and turn right. Ask your students to write the code for this, using a for loop. The code will be:

```
numbers = [5, 2, 6, 7, 0, 9]

for x in numbers

    @step x

    @turn right
```

Review with your students how this code is executed:

1. x = 5 → step 5 and then turn right
2. x = 2 → step 2 and then turn right
3. x = 6 → step 6 and then turn right
4. x = 7 → step 7 and then turn right
5. x = 0 → step 0 and then turn right
6. x = 9 → step 9 and then turn right

Part 1 - Introduction

Activity #2 5 min.

The following activity builds on the first activity to facilitate the understanding of an array and a for loop.

Ask for five volunteers. Name them v0 to v4.

Discuss how to convert the code on the board (from the previous activity) into a code that uses a loop:

```
volunteers = [v0, v1, v2, v3, v4]

for v in volunteers

    step 5

    turn 180

    step 3

    jump

    step backwards 4
```

Part 1 - Introduction

Activity (con't) 5 min.

Ask the volunteers to "execute" the code.

Now ask your students what needs to be done if we want the volunteers to perform their instructions in a different order?

- change the definition of the array.

Ask the volunteers to go back to their first positions. Change the array, for example:

```
volunteers = [v1, v0, v4, v3, v2]
```

The rest of the code remains the same!

Ask your students to "execute" the code based on the new array. Each volunteer performs the same actions as before. The order in which they take turn is different.

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 is 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 Check button is not available, the code should be tested by using the Run button. From now on, usually when running the code, the player is asked to click on an apple.

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 as 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 22 - 26.

Encourage 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 student achievements.

Part 2 – Let's Go!

Walkthrough #1 5 min.

Open exercise #24:

In this exercise, a function is defined and used.

Discuss with your students the reasons why functions are helpful. Their answers should include:

- Once we write the function we can call it later on from any place in the code
  - we do not need to repeat code
- If we want to add more code, we only need to add it once
- When we call a function, we do not need to know how it works
  - just its name, parameters and return value
- It is easier to understand the code once it is broken down to functions

The syntax for defining a function:

```
functionName = (argument) =>
```

Part 2 – Let's Go!

Walkthrough (con't) 5 min.

Go to one of the apple sprites (they can list more functions that are not in these sprites). Ask your students to list:

- functions that are defined in the sprite
  - onClick
  - onAnimationEnd
- functions that are called:
  - addAnimation
  - startAnimation
  - play
  - stopAnimation
  - setX
  - setY

Part 2 – Let's Go!

Walkthrough #2 5 min.

Open exercise #25:

In this exercise, your students will draw a new sprite sheet and then define an animation that will run in a loop.

There is an example in the exercise of an animation that runs in a loop.

The last parameter when defining an animation defines if the animation will run in a loop or just once.

The syntax of defining animation is:

```
@addAnimation "key", [0, 1, 2], frameRate, loopAnimation
```

- loopAnimation:
  - no - will run once
    - this is the default
  - yes - will run in a loop

When the animation runs in a loop, the function onAnimationEnd is not called. Instead the function onAnimationLoop is called each time the animation ends (we don't use this function in this course).

The syntax is:

```
@onAnimationLoop "key", () =>
```

Part 2 – Let's Go!

Play Time 5 min.

Students should complete up to exercise 26.

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 #24.

PART 3

Debriefing

5 min.

Part 3 – Debriefing

Discussion 5 min.

Discuss with your students what we covered in this lesson:

1. How can we make our code more shorter and reusable?
    - use functions
    - use loops
2. What is the index of the last item in an array and why?
    - number of items in the array minus 1
    - because the first index starts from 0
    - for example, if an array has 8 items, the index of the last item is 7
3. How can we stop an animation that runs in a loop after x number of times?
    - define a variable and set it to 0
    - define the function onAnimationLoop
        - increase the variable by 1
        - if the variable equals to x
            - call the function stopAnimation
4. How can we start it again after the animation stops?
    - define onClick and start the animation

Part 3 – Debriefing

Discussion 5 min.

You can use this link as an example.

The following is the code that stops the animation after four times it runs:

Counter widget named count:

```
@value = 4
```

Sprite's code:

```
@onAnimationLoop "test1", () =>
    count.value -= 1
    if count.value == 0
        @stopAnimation yes
```

Code to start animation when clicking on it:

```
@onClick = () =>
    count.value = 4 # to let it run for 4 times after it starts again
    @startAnimation "test1"
```

GREAT JOB!