LESSON PLANS

GAME BUILDER - BLOCK JUMPER

LESSON PLANS

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

Welcome to CodeMonkey's Block-based game builder guide.
With this guide, you can lead a 1-3 hours workshop, in which your students will build their block-based jumper platformer game.
Block-based coding is designed for young learners, it is based on coding instruction designed as blocks which are connected like a puzzle to construct programs.

To learn how to set up a class, please read A Beginner's Guide to CodeMonkey. The guide can also be found in the Teacher's Resources Menu on your homepage.
Please email us at info@codemonkey.com for any questions you may have along the way.

Have fun!
The CodeMonkey Team

Table of Content

Before You Start

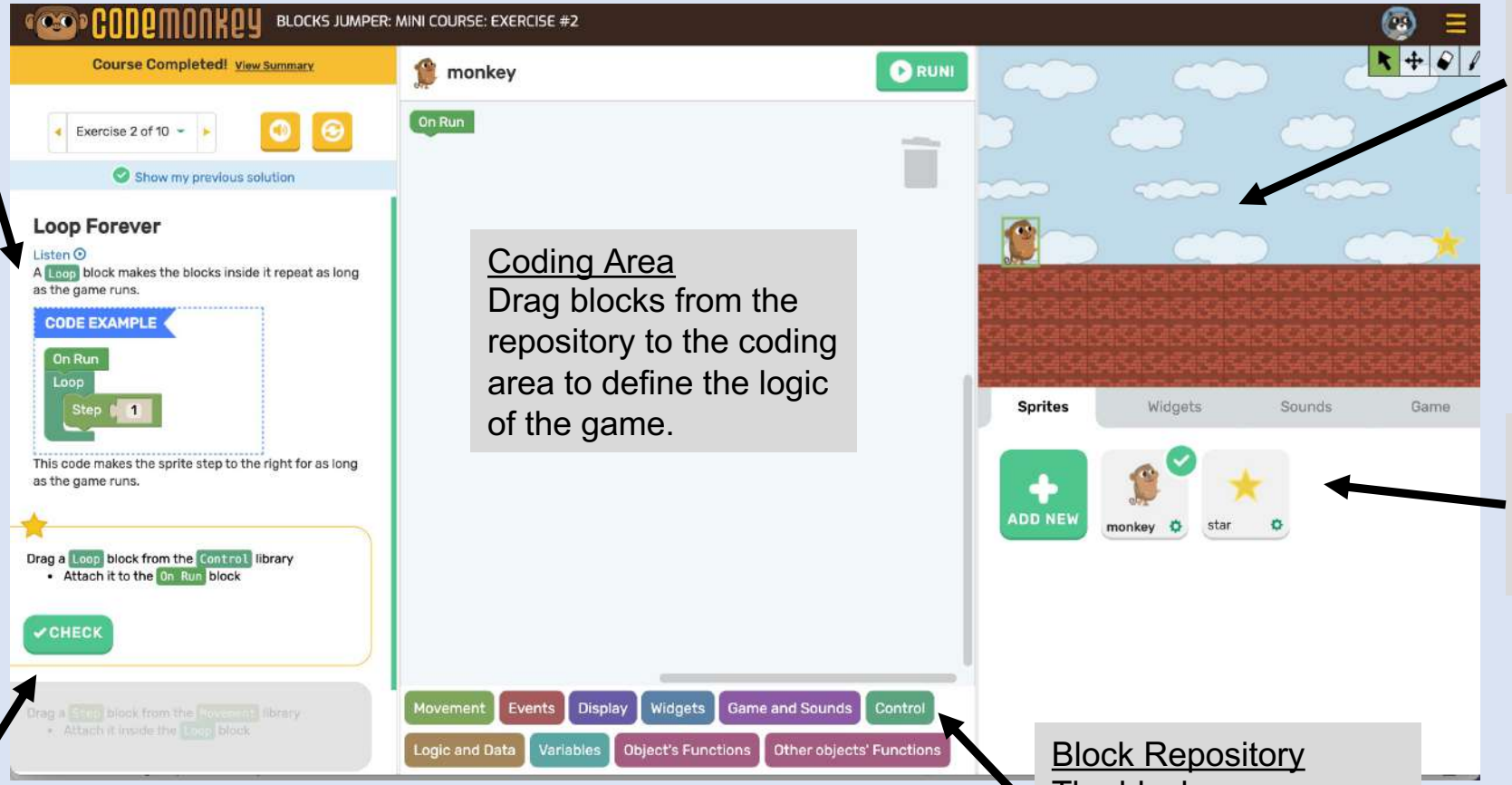
- In this mini course, your students will create a simple platformer game where the monkey runs on the platform until it reaches the star. The player need to tap the screen to help the monkey overcome obstacles like gaps.
- The course includes 10 exercises which lead the student, step-by-step, to creating and publishing their own game.
- We recommend this course for students with good reading skills and some coding background.
- The course is designed for 2 lessons (45 min each.)

Course Structure

- The course is divided into exercises. Each exercise includes mini programing tasks that need to be completed.
- Upon successful completion of an exercise, the next exercise becomes available.
- When the students complete a task, they need to press the “Check” button.
  - If the code is correct, the next task becomes available.
  - If not, then the task is marked as failed and they need to change their code and try again.
- Only when completing a task successfully, the next task is unlocked.
- When the “Check” button is not available, the code should be tested by clicking the RUN! Button.
  - The text of the task will indicate that they need to click on the RUN or STOP buttons.
- Some tasks are checked automatically. For example, when you add a sprite or a sound
  - If you name it correctly then the task completed successful

Course Platform

Instructions
A short explanation on the blocks used in the exercise. Including code example.



Coding Area
Drag blocks from the repository to the coding area to define the logic of the game.

Stage Area
Simulation and view of the game. Clicking the 'RUN' button to run the game.

Formatting Area
Sprites, backgrounds, sounds, widgets and more.

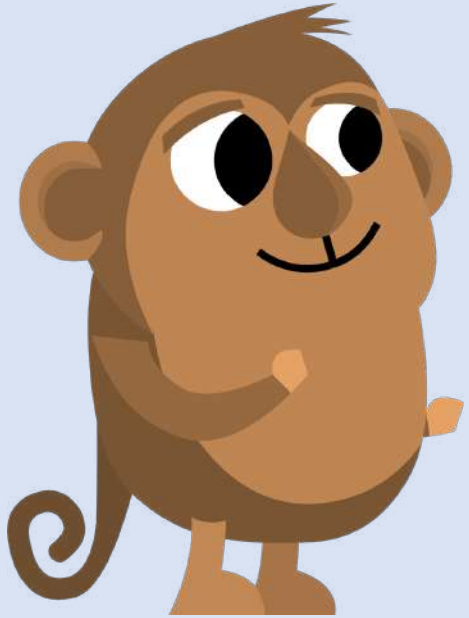
Tasks
The tasks the student need to complete. The first one is unlocked; the rest are locked.

Block Repository
The blocks are grouped by their functionality

Lesson 1 – Introduction

This lesson is designed to introduce basic elements of the game – the sprite concept, stepping and event mechanism.

The students will learn how to manage the sprite's specific code and will add a new sprite.



Objectives

In this lesson, students will:

- Get introduced to the game
- Understand the concept of steps, loops, events and sprites
- Complete challenges 1-4
- Blocks used: Step; Loop; On Collide

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-10 • 1B-AP-16	• 2-AP-12 • 2-AP-15

Part 1: 20 Minutes

Introduction

Unplugged activity	**10 mins.**

The **objective** of this activity is to highlight the meaning of events and how event-based programming works.
This will allow the students to understand what are the triggers (events) for the sprite activity (e.g., step, jump, disappear) and what defines the activity itself (the code attached to the event).

Ask for five (5) volunteers –
2 volunteers will act as the programmers that define the events and instructions, and 2 volunteers will be the sprites performing the instructions and the fifth student will be the orchestrator.

Ask the 2 programmers to define events and instructions and write them on different cards. Make sure the events are defined for a specific sprite or for all.
For example:

- When door opens –> All jump twice
- When teacher raises a hand → sprite 1 walks around the class
- When teacher says hi → sprite 2 walks toward sprite 1
- When sprite 2 collide with sprite 1 → sprite 1 sits on the floor

*If they don't have ideas, the programmers can get help from their friends in the class, make sure you have 4-6 different events.

The cards will be handed to the orchestrator student who will present them to the sprites and will trigger the events (open the door, ask the teacher to raise hands, etc.).
When the event is triggered, the relevant sprite should perform the associated instructions.

After a few events ask the class if they have additional ideas for events and actions, to run another round or two.

Part 1: 20 Minutes

Introduction cont.

Discussion	**5 mins.**

Ask your students to describe the game they played:

1. Did you use spoken instruction? Why yes or how come you didn't need to?
2. Were there any issues?
3. Did a student skipped or miss an event?
4. What was it like to keep track of the different events going on?
5. Can an event trigger another event?
6. Is it similar to real world events? (for example, traffic lights, school bell, phone ringing)

Part 1: 20 Minutes

Introduction cont.

Explain	**5 mins.**

Event is something which happens or takes place. An event is usually short.

Event-based programming is a programming method in which the flow of the program is determined by events. Events cab be triggered by the user or by the program itself.

For example:

- Pressing a key on the keyboard
- Clicking on the mouse
- Scrolling the mouse
- Sprite touches the ground

As a result, events affect what is happening that while the code is running.

In programing, after defining an event we will define an event handler:

- A piece of code (a function) that is called when the event happens and then does something
- An example is the On Run function that is called when we press the RUN button

Part 2: 20 Minutes

Playtime

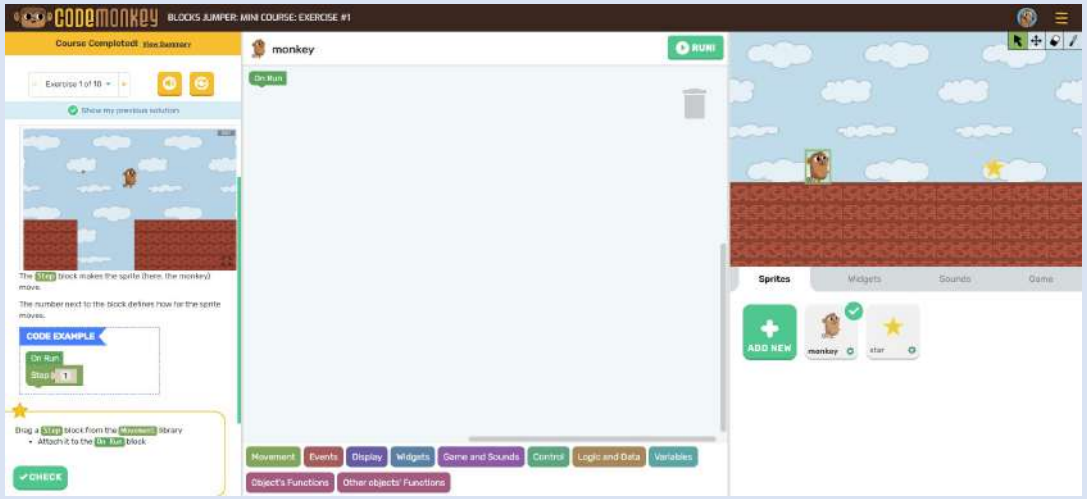
Playtime
Ask your students to complete exercises 1 - 4. Encourage the class to also read the instructions carefully, especially if they do not understand what needs to be done. Use the time to pass between your students and help them. Encourage students who completed all exercises to help their friends or add new blocks and features to the game. Use your classroom dashboard to keep track of your students' achievements.

Exercise 1

What should you pay attention to

Learn how to move the monkey –

1. Sprite – on the right lower area you will see the sprites in the game. The monkey sprite is checked, meaning the code in the code area belongs to the monkey. The current sprite is also indicated on the top left of the code area
2. Event – the main event of the game is pressing the run button on the right-top of the code area.
3. Code area
    - The first block appears in the code area is the ‘On Run’ – this is the main event block. The code attached to it will be executed once the user clicks the RUN button.
    - The Step block has a number inside (argument) – the number indicates how many steps the sprite will move. The student can change the number based on the requirements

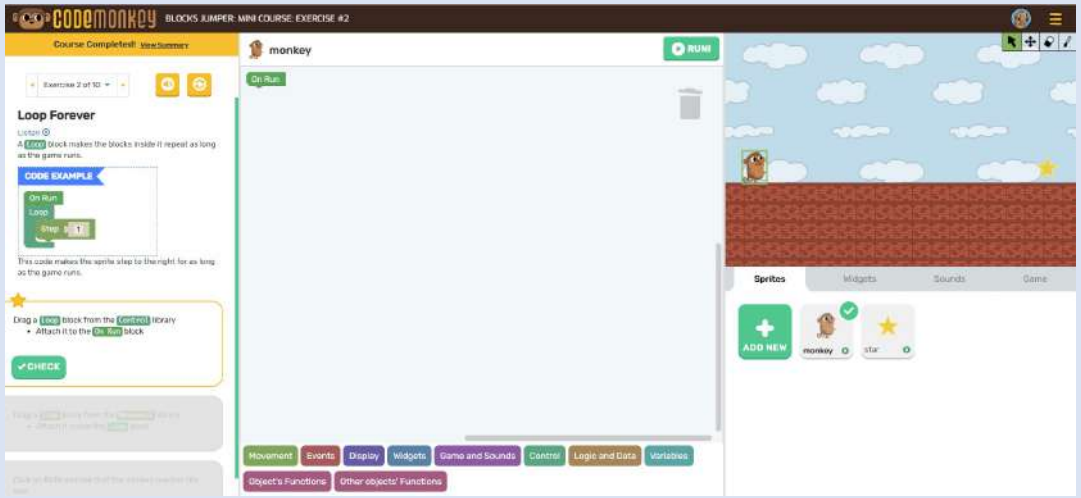


Exercise 2

What should you pay attention to

This exercise is focused on using the Loop block.
We assume that students used loops in the past, to make sure all the students understand the concept, you could emphasize that a Loop block is a repetitive mechanism that makes the blocks inside the loop repeat as long as the loop runs.
The loop will be executed for as long as the game runs.
In programming there are other loop mechanisms that repeat the instructions based on different rules. This is not covered in this course.

Pay attention – in the first exercise, in order to move the monkey to the star we had to change the argument of Step block to 300.
When using a Loop block, a single step will be repeated until the game ends.



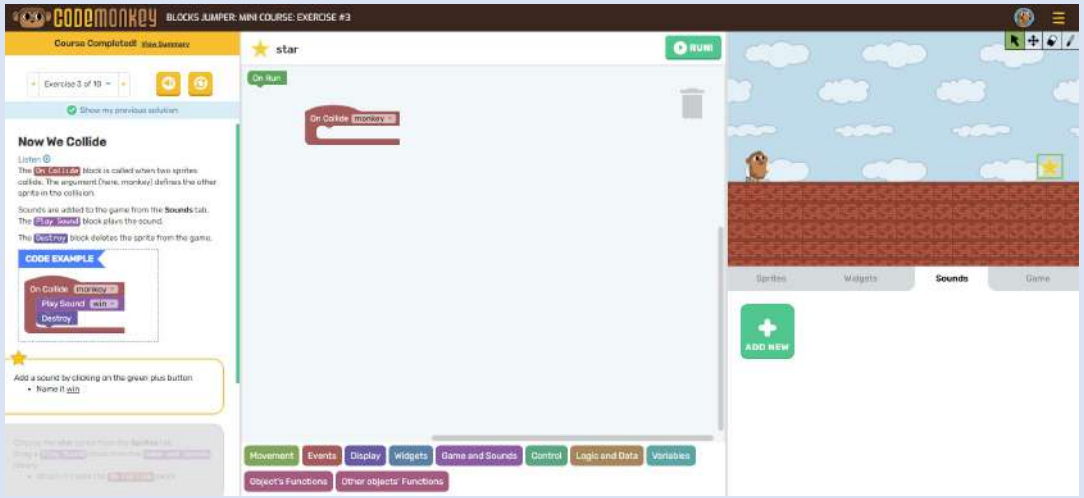
Exercise 3

What should you pay attention to

This exercise is focused on defining the end state of the game – when the monkey reaches the star.
Pay attention – we add code to the star sprite. Make sure all the students understand that the code they write in this exercise refers to the star, they can see the star on the top left side of the code area. Switch between the sprites to show that the code changes accordingly.

Lesson's main blocks:

1. 'On Collide' – this an event type block - the event handles the case where a sprite (in this case, the star) collides with another sprite chosen from the dropdown menu within the block. We will add code inside this event block to be executed when the event occurs.
2. 'Play Sound' – a game and sound type block - playing the sound selected from the dropdown menu.
3. 'Destroy' – a display type block – deletes the sprite from the game.



Exercise 4

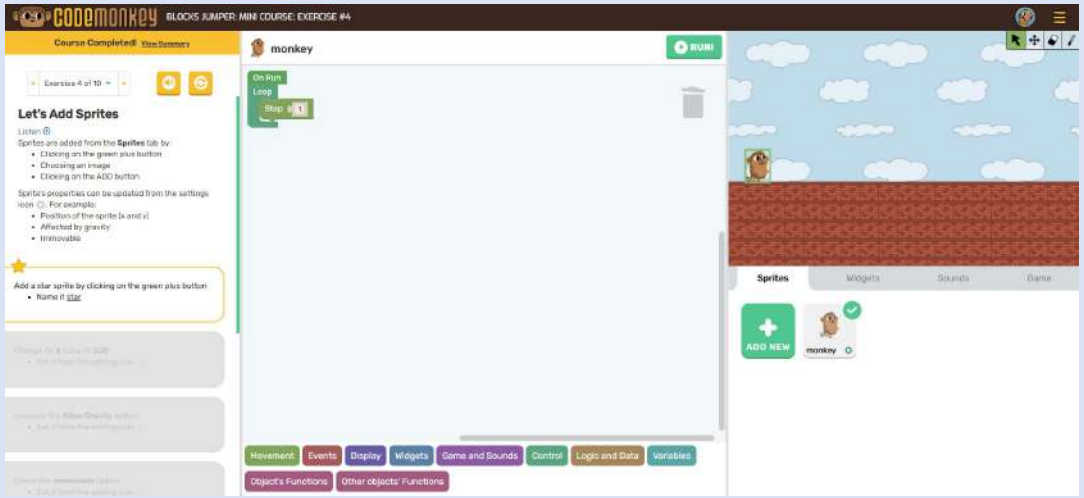
What should you pay attention to

This exercise is focused on adding a sprite.
In the previous exercise the students defined instructions for the star – now they will learn how to add a star.

The students will learn how to add the star sprite, position it and define its settings.
After adding the sprite, they will add the 'On Collide' block like in exercise 3.

When you click RUN you can see a combination of the 2 parts –

1. After clicking the RUN button - The monkey will start moving until it reaches the star
2. Upon colliding with the star, you will hear a sound and the star will disappear



Part 3: 5 Minutes

Debriefing

Summary	5 mins.

Discuss with your students what we covered in this lesson:

What is an event?

- We used 2 different events – the On Run (when the player clicks the RUN button, and the game begins) and the On Collide event (when two sprites collide)

What is an event handling function?

- A function that is called (executed) when the event happens and then does something

Name the sprites that we will be using in this game

- monkey, star

Which instruction blocks did we learn –

- On Run
- Step
- Loop
- On Collide
- Play Sound
- Destroy

What are arguments used for?

- Arguments allow us to use the same block (instruction) differently.
- For example, the Step block has an argument to set the number of steps; the On Collide block has an argument to set the other sprite colliding with; and the Play Sound block has an argument for the sound to be played.
- Once a block has arguments, it gives the programmer greater flexibility for using it in different ways and for different situation – for example play one sound when winning and a different sound when losing.

Lesson 2 – There is work to do

In this lesson, your students will get a better understanding on how to add more functionality to the game using new events, sounds, messages and functions.

By the end of the lesson the students will be able to publish their game.

Objectives

In this lesson, students will:

- Learn how to add a new sprite and set game properties
- Complete exercises 5-10
- Blocks used – Set Speed; On Game Tap; Jump; Reset Game;
- Use the Dialog widget

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-09 • 1B-AP-10 • 1B-AP-16	• 2-AP-12 • 2-AP-15 • 2-AP-18

Part 1: 10 Minutes

Introduction

Discussion	**10 mins.**

If the lessons are held in two consecutive hours you can skip this intro, if you have couple of days between the lessons you can use this opening to brings the students back on track.

Open the game on exercise 4 which was completed in the previous lesson.
Ask your students –

1. What can be added to the game to make it more fun?
2. Will they add obstacle?
3. Events?
4. Additional sounds?
5. Or maybe another sprite?

Platformer game design is endless. Here are some ideas, encourage your students to come up with their own:

- Background - There are several backgrounds pre-loaded in the game. Select the one you want from the Game tab.
- Communicate – Add new widgets and sounds, make it your own.
- Customize the sprites – you can use your own sprites or new sprites that will be available once you complete the course.

Part 2: 30 Minutes

Playtime

Playtime

Ask your students to complete exercises 5 - 10.
Encourage the class to also read the instructions carefully, especially if they do not understand what needs to be done.

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

Encourage students who completed all exercises to help their friends or add new blocks and features to the game.
Use your classroom dashboard to keep track of your students' achievements.

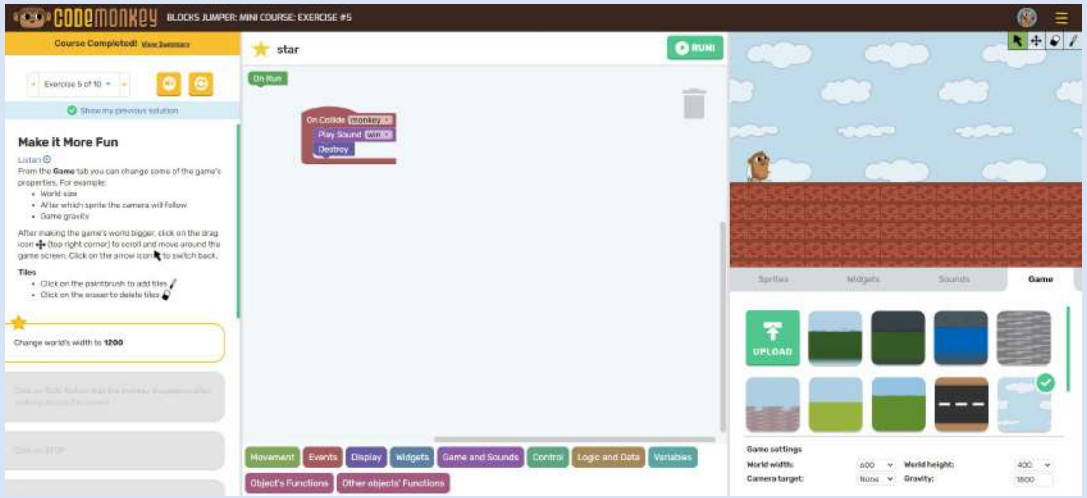
Exercise 5

What should you pay attention to

Now that we have a code for the monkey and the star, we can go deeper into the game's properties.

In this exercise the student will –

- Change the world's size
- Learn how to control the camera
- Add new tiles

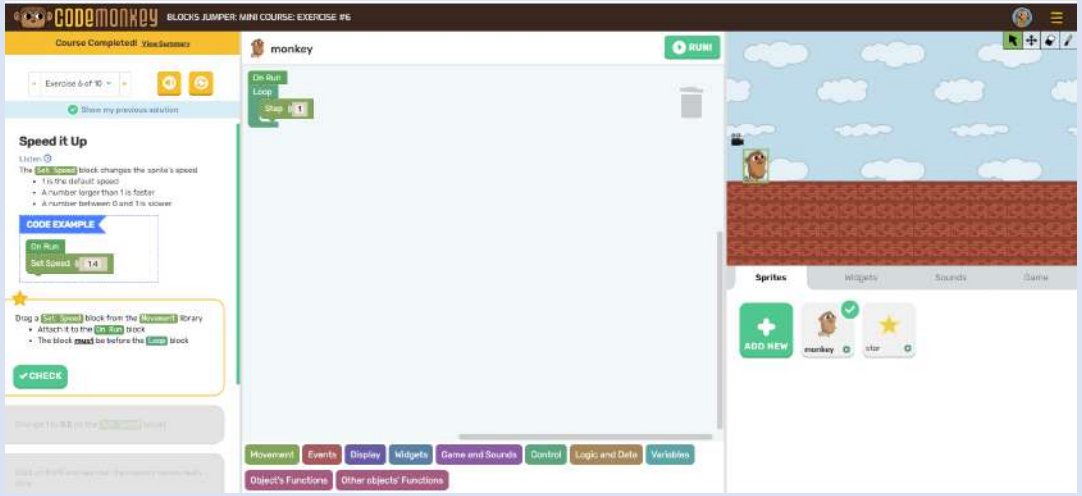


Exercise 6

What should you pay attention to

This exercise is about setting the sprite's speed using the Set Speed block.
Pay attention to the correct place to connect the Set Speed block, you can discuss with the students what is the difference between the following options:

1. Before the loop – this is the correct place, set it once before the monkey start moving
2. Inside the loop – this will invoke the action on each and every iteration of the loop
3. After the loop - the loop ends when the game ends; we can not connect a block after the loop since we will never reach that point

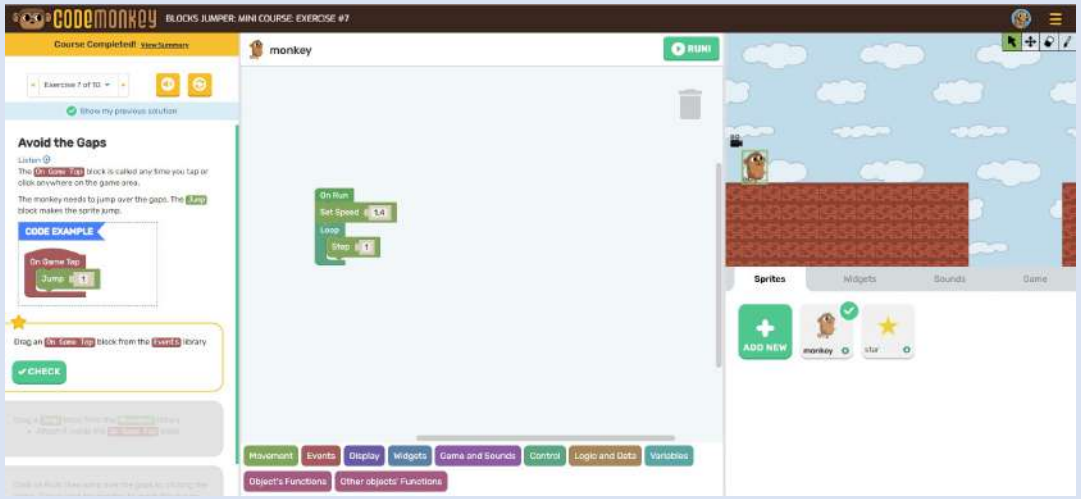


Exercise 7

What should you pay attention to

After enlarging the game in exercise 5 we can add more challenges – this exercise is focused on jumping over gaps.
To do so we will use the ‘Jump’ block from the Movement library.

But how can we trigger the monkey to jump at the right point in time? We can use the keyboard or the mouse to make it jump when we reach a gap.
We can add another mouse event - the ’On Game Tap’ block.
This block is called any time you tap or click anywhere on the game area.



Exercise 8

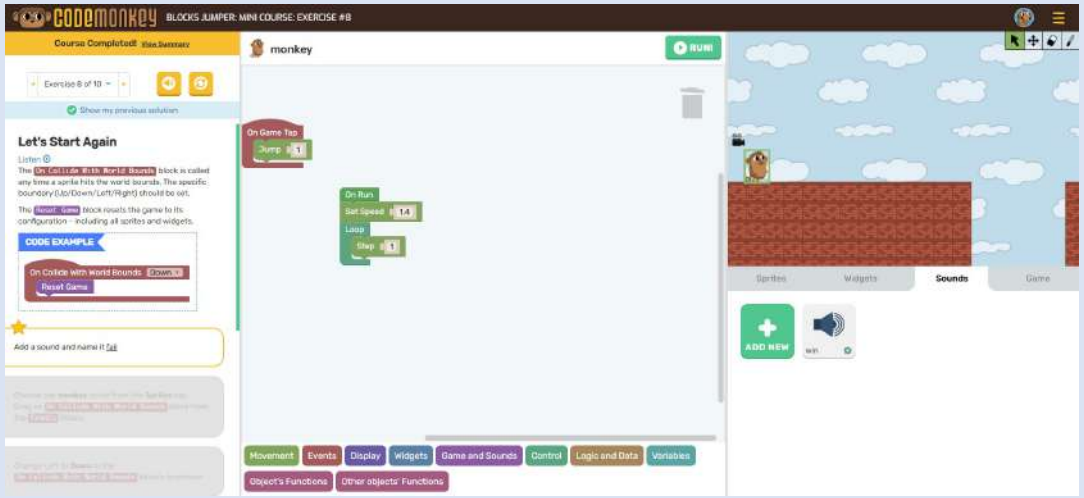
What should you pay attention to

Let’s make the game a bit more complex. The monkey is not allowed to touch the ground. If it touches the ground, restart the game.
What is needed here?

1. We need a new event – that detects the situation where the monkey touches the ground

The ‘On Collide With World Bounds’ block is called any time a sprite hits the world bounds.
The block has a dropdown menu to select the specific boundary (Up/Down/Left/Right) as an argument (similar to setting the number of steps within the move block or selecting another sprite in the ‘On Collide’ block).

We will use the ‘Reset Game’ block to reset the game to its original configuration - including all sprites and widgets.



Exercise 9

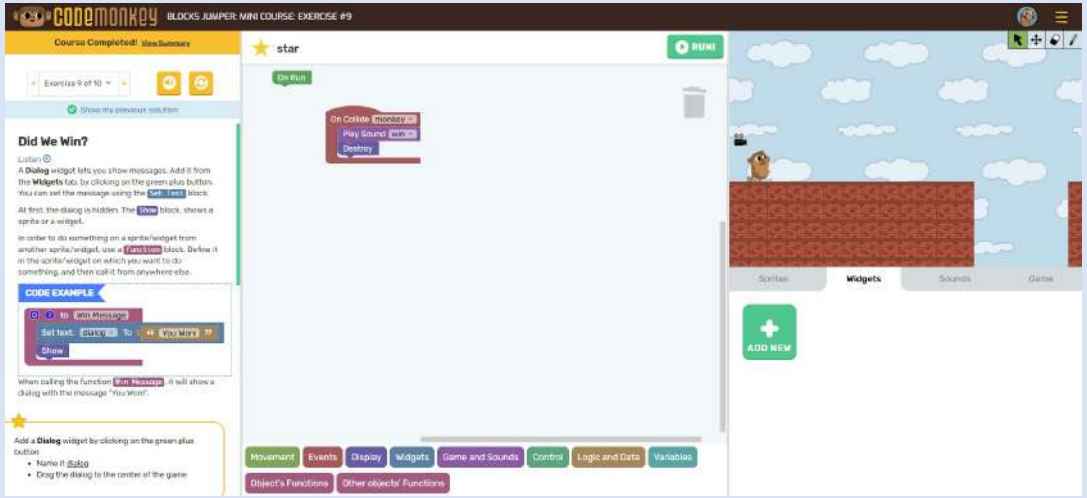
What should you pay attention to

In this exercise you will learn how to use widgets.
Widgets are game objects that display information or interact with the player.
For examples – counter, dialog, timer and more.
To create a wow effect when winning, let's add a Dialog widget from the **Widgets** tab, by clicking on the green plus button.

The message appears on the screen – uncheck the show button to hide it.
At this point we are going to add the functionality of the widget -

- In order to do something on a sprite/widget from another sprite/widget, we will use a function block.
- Define it in the sprite/widget on which you want to do something, and then call it from anywhere else.

Once we defined the widget's function, we can go to the Sprites tab and add the function we just created – this is done from the Other Object's Functions library.

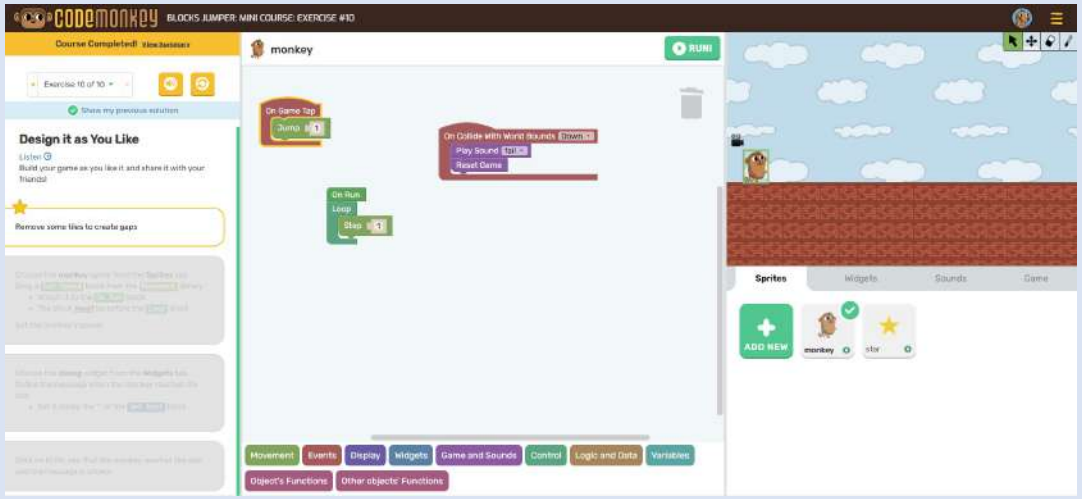


Exercise 10

What should you pay attention to

Congratulations! This is the last exercise of the couse and your students can publish and play their games.
When playing the game - discuss with your students the main concepts they learned, make sure they understand how they developed each and every element of the game

- Block programing
- Sprites
- Events
- Loops
- Widgets



Part 3: 5 Minutes

Debriefing

Summary	**5 mins.**

Congratulations! You have published your first game.
At this point you can start enhancing the game – encourage your students to spice up the game with their own ideas

1. Add another sprite
2. Design and style the game - change the game background, add tiles, steps, position star, change world size, set monkey's speed & scale
3. Add a Counter widget that counts the lives of the monkey - reduce one life when the monkey hits the ground, pause the game when there are no more lives
4. Add another Dialog widget and show a message - game over; or use the same Dialog message and update the text based on if player won or lost
5. Add a Timer widget – the monkey needs to get to the star before the time ends; better not fall in any gaps…

Main Coding Concepts

Concept	Explanations
Sprite	Sprite is referred to a graphical object, usually a two-dimensional bitmap that is integrated into a larger scene. A game can have multiple sprites.
Event	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: • input events - Pressing a key on the keyboard; Clicking on the mouse; Moving the mouse; etc. • game events - sprites collusion with another sprite; sprite collusion with world boundary; etc. 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.
Loop	Loop is a coding mechanism for repeating a sequence of instructions until a particular condition is met, this is fundamental algorithmic mechanisms that is used in most of the programming challenges.
Widget	Widgets are objects that we can be added to the game, like Counter, Text, Timer, Clock, Dialog, Button. Once a widget is added to the game, we can add functionality to control and manage the widget – for example when and how to present it. In this course, we will introduce the Dialog widget.
Function	Function is a set of instructions bundled together to perform a specific task. Some of the functions have parameters. Parameter is a special kind of information/data that allows us to configure the function, like number of steps is a parameter for the function step. Using parameters increases the flexibility the programmers has when using a function.