



Hour of Code™ at Box Island!

Curriculum

Welcome to the *Box Island – Hour of Code* curriculum! First of all, we want to thank you for showing interest in using this tutorial with your children or students. Coding is becoming a key part in the 21st century literacy. There is no question about it. The world is transitioning to a point where coding will be a necessary skill in most modern jobs. Study by Oxford researchers suggests that 47 percent of America's occupations will be automated within the next 20 years¹.

Therefore, sparking children's interest and enabling them to take easy and frictionless first steps into the field of computer science is vital. Not only that, we believe that it should be their right to get an early introduction to computer science, just like they get introduced to math, music, sports and other subjects.

The mission of Radiant Games is to empower children by providing the best playful coding experiences possible. We spark interest by crafting innovative coding games that engage through active participation. We ensure a trusted bond with parents and teachers through quality and integrity, where children learn as they play.

Our first game takes place in the tropical wilderness of Box Island. In this curriculum, we will expand on the *Box Island – Hour of Code* mobile tutorial, which was specifically designed for the Hour of Code campaign.

Learn more about the Hour of Code tutorial at <https://boxisland.io/hourofcode>
Learn more about Box Island at <https://boxisland.io>

If you have any questions, contact us! hello@radiantgames.is

- The Radiant Games team

¹http://www.oxfordmartin.ox.ac.uk/downloads/academic/The_Future_of_Employment.pdf

Curriculum Summary

INTRODUCTION - 3

- What is Computer Science?
- About Box Island: One Hour Coding
- Objectives
- Vocabulary

HOW THE GAME WORKS - 5

- Switching Languages
- Instructor's Menu
- World Map
- Gameplay and Evaluation
- Hour of Code Certificate

CODING CONCEPTS AND DESIGN - 11

- Algorithms
- Sequences
- Loops
- Conditionals
- Debugging and Testing

INTRODUCTION

WHAT IS COMPUTER SCIENCE?

Computer science is the scientific and practical approach to computation and its applications. The practical side of computing can be seen everywhere, as technology impacts our lives in so many ways. In today's world, a majority of the population is a computer user or consumer. However, not everyone understands how to get computers to do new and original things. This is where computer science comes in.

Unlike electrical and computer engineers, computer scientists work mostly with software and software systems. The field can be seen on a higher level as a science of problem solving, which requires precision, creativity, and careful reasoning.

Computer science can be divided into three domains; Mathematics, science and engineering. Mathematics provides logic and reasoning. Science provides the methodology for learning and refinement. Engineering provides the techniques for building hardware and software².

There are many interesting areas within the field of computer science, such as artificial intelligence, computer architecture, software engineering and bioinformatics. Computer science and its methods are also used in most other industries and areas. For example: Data from acres is used to instruct watering and fertilizing robots in farming, programmable LED dresses are emerging in fashion and robots are used widely in manufacturing. All these examples have in common that they rely on software and software systems.

ABOUT BOX ISLAND - HOUR OF CODE

Recommended Age Range: 6+

Technical Requirements: Phones or tablets, Android & iOS.

Supported Languages: English, Spanish, German, Italian, French, Portuguese, Brazilian, Chinese, Japanese, Korean, Vietnamese, Indonesian, Turkish, Greek, Romanian, Russian, Croatian, Czech, Hungarian, Polish, Ukrainian, Dutch, Swedish, Norwegian, Danish, Finnish, Icelandic.

² <https://www.cs.mtu.edu/~john/whatiscs.html>

In this fun mobile tutorial, students take part in a journey on Box Island that has been specifically designed for the Hour of Code campaign. Students are introduced to many of the basic fundamentals of coding through engaging puzzles in the tropical wilderness of Box Island. That includes fundamentals such as algorithmic thinking, sequencing, and loops.

In the tutorial, students take part in a journey on Box Island, where they have to help Hiro (main character) to collect stars around the tropical paradise, where the goal in each puzzle is to collect 3 stars. In total, the tutorial consists of 20 logical puzzles where each is represented as a level. Note that you can select a proper age group before starting the tutorial.

The tutorial is intended for beginners of all ages and has the goal of being gender neutral and cultural friendly. The tutorial offers three different sets of levels suitable for different age groups, i.e. 6-8 year olds, 9-11 year olds, and 12+ year olds.

Get the lesson plan for the tutorial here: [Lesson Plan](#).

Get the solution guide for the tutorial here: [Solution Guide](#).

OBJECTIVES

By participating in this tutorial, students will

- Gain basic skills in the fundamentals of coding in an engaging way
- Put those skills into perspective with real-world applications and problems
- Identify key computer science vocabulary
- Gain valuable skills in analyzing mistakes and coming up with suggested solutions (i.e. debugging)
- Be positively influenced to learn more about computer science and coding

VOCABULARY

- **Code** – To write code, or to write instructions for a computer
- **Algorithm** – An algorithm is a step-by-step method of solving a problem
- **Program** – An algorithm that has been coded and therefore can be run by a machine.
- **Debugging** – Finding and fixing problems in your algorithm or program.
Simulate – The process when a program is being run (on a machine).

HOW THE GAME WORKS

In this chapter, we will go through the features of the *Box Island – Hour of Code* tutorial and its gameplay. The first screen displayed in the tutorial is the main menu screen. The screen has two sections, i.e. Hour of Code and Box Island. It also contains an info button in the bottom right corner where you can mute sound or audio, change languages and access the instructor's menu that will be covered in further detail below. Press the Hour of Code play button to proceed to the tutorial.



SWITCHING LANGUAGES

The tutorial currently has in-game support for 27 languages. After clicking the info button on the main menu screen, simply tap the language button from the info list and select a language of choice.



PARENTS / INSTRUCTORS MENU

Parents and instructors (i.e. teachers, educators) can access several features that are only intended for them. The parents' menu can be accessed through the info button.

Privacy: A link to the tutorial's privacy policy

EULA: A link to the tutorial's end-user license agreement

Credits: Displays the makers of this tutorial

Like us: A link to like the tutorial's Facebook page

Follow us: A link to follow the tutorial's Twitter page

Rate us: A link to the App Store/Play Store (depends on device) to rate the tutorial

Subscribe: Allows you to subscribe to Box Island updates



CREATING AN AVATAR

Once you have selected to enter the Hour of Code section of Box Island from the main menu, you have to create an avatar profile. The profile will store the student's progress on the device. A profile can be easily deleted if needed.

After a nickname has been chosen, you can select between three different age groups for the tutorial, i.e. 6-8 year olds, 9-11 year olds and 12+ year olds. The level difficulty in the tutorial is custom for each group.



WORLD MAP

The world map displays an overview of a student's progress in the tutorial. The journey represented on the map is designed in a clock-wise manner, which allows you to visualize and evaluate the progress of a student very quickly. To simplify the evaluation even further, there is a counter displayed at the top of the screen that displays how many stars the student has collected in total in the levels played.



The tutorial has a total of 20 logical puzzles represented as levels. They can be accessed in any order, but we strongly recommend that they'll be solved in a sequential order (from first to last). Dependent on which age group was selected for the student, sections of the coding fundamentals are represented on the map. For 6-8 year olds, they will have sections of sequences and loops. For 9-11 year olds and 12+ year olds, they will have sections of sequences, loops and conditionals. Each section has the role of introducing that particular concept to a student in an effective manner. We will discuss these fundamentals further in the "Coding concepts and design" chapter below. The yellow button after the last level is the

certificate of completion button. The button will navigate the student to a screen where he is congratulated for completing an Hour of Code. The button can be clicked at any time.

GAMEPLAY AND EVALUATION

Gameplay is the central focus of the tutorial. Students have to help Hiro, the main character, to collect stars around the tropical paradise of Box Island. In this section, we will explain the coding instructions used, the two phases of the gameplay and how we evaluate puzzle solutions.

Instructions

There are 6 instructions in total that can be used in solving the puzzles in the game.

Forward: Tells Hiro to move forward one tile

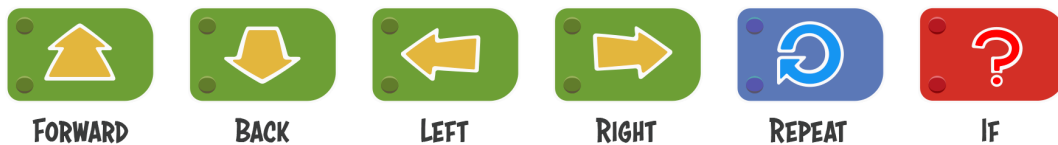
Backwards: Tells Hiro to move backwards one tile

Left: Tells Hiro to move one tile to the left

Right: Tells Hiro to move one tile to the right

Repeat until tile: Allows Hiro to repeat a sequence of instructions until he reaches a tile of a certain color (Loops)

If stands on tile: Allows Hiro to identify the color of the tile he is currently standing on (Conditionals)



Puzzle phase of the gameplay

In the puzzle phase, students have to break down puzzles displayed in the 3D game world of the gameplay and apply algorithmic thinking to come up with a solution to them. The puzzles can include movable crates, colored tiles, two types of red enemies (both static and laser-sighted enemies), and the stars that are to be collected. Students use the toolbox on the left hand side to drag and drop instructions, one instruction at a time, and attach them to the "On Start" token. This way, sequences of instructions are formed to represent a *program* that aims to solve the problem of collecting the stars in a particular level. Note that stars can be spread around the board of the level.

In the level presented in the image here above, students can use 4 instructions to solve it. The correct program is: "Forward, Right, Right, Backwards". In general, there can be more than one correct sequence to puzzles in the tutorial.

A solution to a level is evaluated using a 3-star system. The goal is to collect all 3 stars, but users are allowed to progress through the experience even if they only get 1 or 2 stars.

Note that students can move the camera of the 3D game world, which allows them to take a better look at certain parts of puzzles. Students can also tap the water in the environment for fun, which will cause a water splash.

Execution phase of the gameplay

Once the play button in the interface is pressed, the assembled program for a puzzle is executed. The camera of the 3D world pans down and follows the movement of Hiro. The toolbox on the left hand side is switched out for a live "debugger", which allows students to follow which instruction is being executed in the program at any given moment. Students can stop the execution at any time by pressing the stop button in the top right corner. We will discuss the value of debugging and testing in the "Coding concepts and design" chapter below.



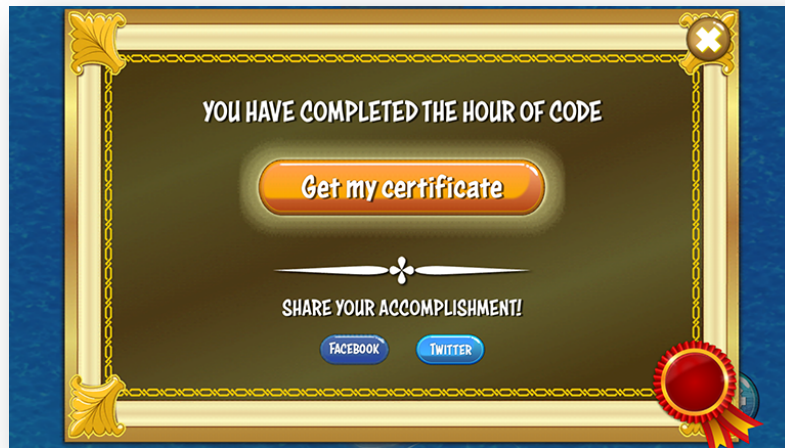
Results after executing a solution

If the execution of a program results in Hiro collecting the stars in the level, we can determine that the level has been solved. If the constructed program includes the fewest number of instructions possible, the student will receive 3 stars and will be motivated to continue his journey around Box Island. However, if the student receives either 1 star or 2 stars based on how close to the best solution he was, he will be motivated to retry the level.



HOURLY OF CODE CERTIFICATE

When the last level of the tutorial is completed, a certificate screen will congratulate the student for completing an Hour of Code. The screen provides a “Get my Certificate” button that will navigate to Code.org’s [certificate of completion](https://code.org/certificate) site. The screen also provides Facebook and Twitter buttons to share the experience. Note that the certificate screen can always be accessed through the world map of the tutorial.



CODING CONCEPTS AND DESIGN

Now that we have covered how the game works, let's explore the underlying coding concepts and designs.

ALGORITHMS

An *algorithm* is a step-by-step method of solving a problem. Algorithmic thinking (also referred to as planning) is in the core of solving a computational problem by coding. It is an exercise in observing a problem and breaking it down into hierarchies of smaller problems. Once you've broken a problem down into a set of smaller problems, you can lay out the step-by-step method of solving the problem.

In general, algorithms can be performed with or without a computer. For example, here is an algorithm for solving the problem of brushing your teeth:

1. Enter bathroom
2. Find toothpaste and toothbrush
3. Place toothpaste onto brush
4. Return toothpaste to storage place
5. Turn water on
6. Wet toothbrush head
7. For a total of 2 minutes, brush all teeth in your mouth

8. Rinse mouth
9. Rinse toothbrush
10. Return toothbrush to storage place
11. Exit bathroom

The above example is a task that humans perform every day. However, in order for a computer to understand the algorithm, it needs a more detailed list of unambiguous instructions for carrying out the execution. For example, “8. rinse mouth” is a very ambiguous step in the algorithm above. How much water should be used, for how long should you rinse your mouth, in which angle should you position your head, etc. Therefore, a big portion of the algorithmic thought process is to come up with unambiguous solutions that computers can execute.

In this *Box Island – Hour of Code* tutorial, problems are represented as logical puzzles in a 3D game world, which are very visual and easy to break down. In the following sections, we will expand on the coding fundamentals that are covered in the tutorial. We will also expand on testing and debugging programs.

SEQUENCES

In general, an algorithm is executed in a sequential fashion, from its first step to its last. This means that the order of the instructions executed matters a lot. If we change the sequence of a step-by-step method provided by an algorithm, the outcome of the problem we are solving can change drastically.

Lets take the algorithm above for brushing your teeth as an example. If we change the order of the algorithm in such a way that we return the toothbrush to its storage place (step 10) before we brush our teeth for 2 minutes (step 7), we will obviously not be successful in solving the problem we set out to solve (i.e. brushing our teeth).

Similarly, if we change the order of the program below in such a way that Hiro will



move to the right before initially moving forward, he will jump into the water and not be able to collect the stars.

In *Box Island – Hour of Code*, the first levels for all age groups of the tutorial focus on introducing students to algorithmic thinking and the sequential order of instructions in algorithms.

LOOPS

In coding, a loop is a sequence of instructions that is continually repeated until a certain condition is met. Loops are a very powerful mechanism that computers can execute extremely often in a short amount of time (over million times per second for simple loops!). Therefore, they are very handy for repetitive tasks. Understanding which sequence of instructions to repeat (i.e. pattern recognition) is the key to using loops successfully.



The sequence of instructions inside a loop is often referred to as the body of the loop. Before executing the instructions in the body of a loop, we first need to check whether the condition of the loop is met. In this tutorial, we use colors of the tiles in the puzzles as the possible conditions (red, yellow, green, and blue tiles). In the example program above, the condition of the loop is a red tile. If Hiro is not standing on red tile, we will enter the body of the loop and execute “Forward” and then “Right”. Once Hiro has completed those instructions, we go back up to the condition and check again whether Hiro is still not on a red tile. We will repeat this process



until Hiro lands on a red tile. When that happens, we will jump below the loop and execute the next instruction that follows. If there is no instruction left to execute, the program will stop. Above, a level is demonstrated where the program above is used successfully to collect the stars.

In *Box Island – Hour of Code*, after getting an introduction on sequences, the tutorial for all age groups focuses on introducing students to loops and the art of recognizing patterns that can be repeated.

CONDITIONALS

In coding, a conditional (a.k.a. if-statement) is a mechanism that executes a sequence of instructions only if a certain condition is met. Conditionals are one of the most important elements in coding, because they enable the same program to act differently each time it is executed, depending on the input to the program.



Conditionals and loops essentially behave in the same manner when it comes to checking its condition and then executing its body. The difference is that a conditional will execute only once, whereas a loop will execute until its condition is met. In this tutorial, just as with loops, we use colors of the tiles in the puzzles as the possible conditions for conditionals (red, yellow, green, and blue tiles). In the example program above, we use a conditional within a loop. Hiro will move forward until he reaches a yellow tile, but will check after each move forward whether he is on a red tile. If Hiro is on a red tile (the conditional), he will move to the left.



Otherwise, he will skip the body of the conditional (if he's not on a red tile). Above, a level is demonstrated where the program above is used successfully to collect the stars.

In *Box Island – Hour of Code*, for 9-11 year olds and 12+ year olds, the last section of levels in the tutorial focuses on introducing students to conditionals and the art of recognizing patterns that can be repeated.

DEBUGGING AND TESTING

Debugging is the process of locating and fixing errors that prevent a program from executing correctly. These errors are commonly referred to as *bugs*. In order to debug a problem, we must pay special attention to the sequence of instructions in the “buggy” program. Debugging involves identifying what the bug is, then isolating the source of the bug, and then fixing it. In order to be assured that the bug has been fixed, we should test the program by executing it again. If the program is still “buggy”, then we should repeat the above process until we have fully resolved the bugs in the program.

When a program is executed in the tutorial, the live “debugger” on the left hand side of the screen helps students follow which instruction is being executed in the program at any given moment. For example, in the level below, the program has a bug in the body of its conditional that causes Hiro to jump in the water.



The live “debugger” helps us identify the bug. That is, there should only be one forward instruction in the conditional, not two. We remove the unnecessary left

instruction from our program and test the updated program by executing it again. This time, Hiro is able to avoid all troubles and collect the stars successfully!

