Code Club

codekingdoms

Puzzle Pack 3

About this guide

This guide is for teachers and club volunteers and provides a solution to completing the puzzles. For this reason, **this document should not be shared with children**. This document also outlines the learning objectives of each puzzle.

Introduction

In this project, children will learn to use conditional expressions with logical operators in blocks of JavaScript code to build game puzzles in their kingdom. To give the activities some context, the children are building puzzles to protect their land from invasion by Glitches. Each puzzle should be solvable so that friendly animals can move around the player's kingdom.

Resources

This project requires the use of Code Kingdoms, which is best used in Google Chrome. You can find the website at codekingdoms.com/codeclub

Learning Objectives

JavaScript

Basic functions: I understand how to use JavaScript functions to achieve my aims.

Conditional expressions: Using my knowledge of comparison operators, values, and booleans, I can write a conditional expression that results in true or false.

Constant parameters in functions:I understand how parameters can affect the output of a function.

Changing values (numbers, strings, Booleans): I understand the difference among different types of values and can explain in what situations I might use each.

Computational Thinking

A1: Writing instructions that if followed in a given order (sequences) achieve a desired effect.

A5: Writing instructions that repeat groups of constituent instructions (loops/iteration) to achieve a desired effect;

Computing Progression

PA2: I know that computers need precise instructions.
PP2: I can create a simple program.
PP3: I can run check and change programs.

YA1: I know that algorithms are implemented on digital devices as programs.
YA2: I can design simple algorithms using loops, and selection i.e. if statements.

YP1: I can use arithmetic operators, if statements, and loops, within programs.
YP2: I can use logical reasoning to predict the behaviour of programs.

Challenges

Spikey Button

Introductory code a button and spike puzzle

Old's Buttons

Coordinate your button puzzle with the use of IF statements and NOT logical operator.

Crate Throw
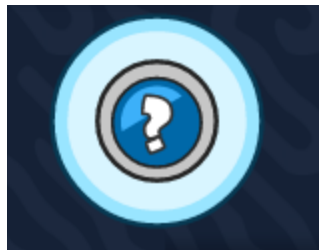
Fling some crates onto coded Red Buttons

Conditional Fling

Coordinate your button puzzle with the use of IF statements. Add flying crates for more fun!

Frequently Asked Questions

Q. What other help is there for the player?

    a.   To help visualise what a puzzle should look like, players can click on the Question Mark icon to see a completed version.

    b.   If a player is unsure which steps of the puzzle they have completed or what still needs doing, they can click on the checklist icon to see a list of the required steps.

Q. What is the purpose of building puzzles?

    a.   Building puzzles is a defence against invading Glitches. The puzzles, however, must be solvable by friendly animals so they can move freely around your kingdom. Glitches aren't very clever so if a puzzle is coded well they are unlikely to be able to solve it. The general rule is if an animal tester can bypass a puzzle piece (e.g. catapult) to reach the checkpoint then a Glitch will easily get past the puzzle.

Spikey Button

Basic Functions · **A1** · **PA2** · **PP2**

Description
This basic puzzle introduces the mechanics of building puzzles and as such is quite straightforward. It explains how to place puzzle pieces (e.g. a button and spikes) and using the coding interface.

Design Tip
Encourage players to place their button in a location that will be tricky for Glitches to find. If their puzzle doesn't keep the Glitches out it should at least hamper their progress.

Steps to complete

1 Place the Blue Button in the puzzle area.

2 Add some code to the Blue Button for when it is pressed down. This is achieved using the onPress event.

3 The obstacle should be solved by pressing the Blue Button so the chunk of code needed is obstacle.solve()
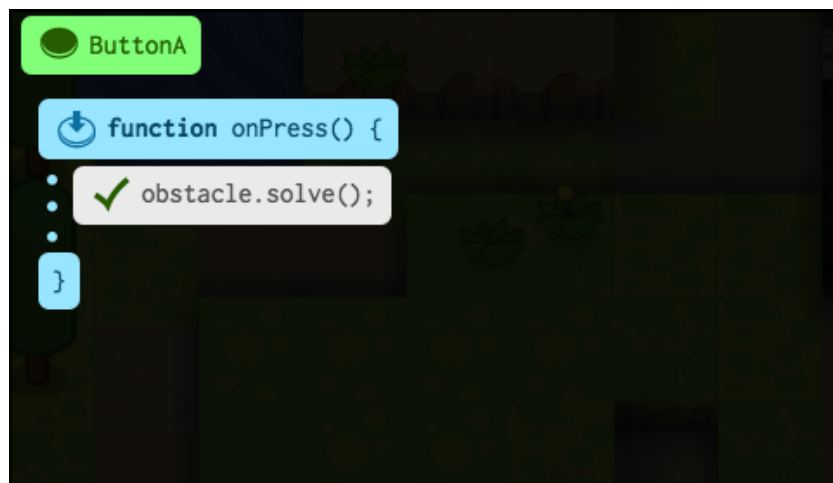
4 Place some spikes to block access to the Blue Button.

5 Test the puzzle by pressing the play icon in the bottom right corner of the screen. If the tester animal fails to solve the puzzle, there is an opportunity to edit and improve the puzzle.

Code required to complete the puzzle

When complete, the Blue Button should have the following code:

Skill Development

JavaScript

Basic functions: I understand how to use JavaScript functions to achieve my aims.

Computational Thinking

A1: Writing instructions that if followed in a given order (sequences) achieve a desired effect.

Computing Progression

PA2: I know that computers need precise instructions.
PP2: I can create a simple program.

Old's Buttons

Conditional Expressions A5 YP1 YA2

Description
Old's Buttons introduces the player to IF statements with the logical operator 'NOT'. Both Purple Buttons must be released for the obstacle to be solved – a coded IF-NOT statement will check for this condition.

Design Tip
It is easier to place crates in the map and drag them on top of the buttons rather than trying to place them directly onto the buttons.

Steps to complete

1 Build two trenches using the brick and dirt blocks. Place the buttons in the trenches.

2 Starting with Button A, code an IF-NOT statement that requires both buttons to be released to solve the obstacle.

Repeat for Button B.

3 Code the obstacle to be 'unsolved' when the buttons are pressed.

**4** Place the crates on top of the buttons.

5 Test the puzzle using the play icon in the bottom right corner of the screen.

Code required to complete the puzzle

To create an IF-NOT statement for each button you will need to write the code shown below. The NOT is denoted by the ! symbol.

The section of code means "IF Button B is NOT pressed, solve the obstacle."

IF statements are found under the languages tab when in the Sequencer, as is the NOT expression. Once placed in the Sequencer the condition, in this case ButtonB.pressed, can be dropped inside the IF-NOT statement.
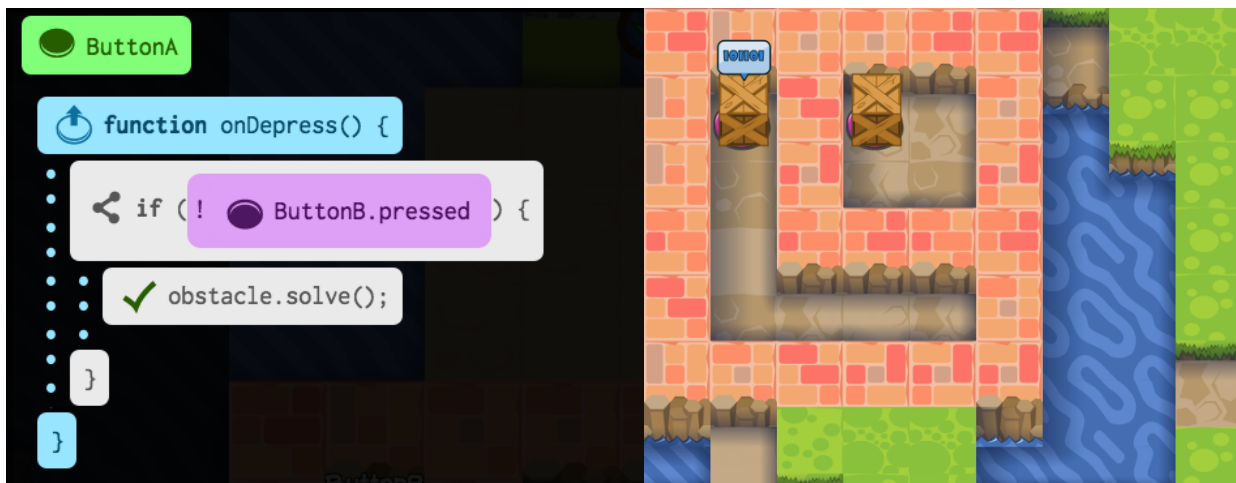
Skill Development

JavaScript

Conditional expressions: Using my knowledge of comparison operators, values, and booleans, I can write a conditional expression that results in true or false.

Computational Thinking

A5: Writing instructions that repeat groups of constituent instructions (loops/iteration) to achieve a desired effect.

Computing Progression

YP1: I can use arithmetic operators, if statements, and loops, within programs.
YA2: I can design simple algorithms using loops, and selection i.e. if statements.

Crate Throw

Changing values

Constant parameters in functions A1 PA2

PP2 YA1 YP2 PP3

Description
Crate Throw is similar to Crate Weight but with the added interest of coding a catapult to fling the crate. Remember Red Buttons need code for when they are pressed and released.

Design Tip
Ensure that the catapult fires the correct distance and direction to land on the Red Button.

Steps to complete

1 Create a narrow waterway to fling the crate across. Use the Question Mark icon if you are unsure how this should look.

2 Place the Red Button and add some code to solve the obstacle for when it is pressed down.

3 Red Buttons also need code for the event when they are released. Using the event onDepress, make sure the puzzle becomes 'unsolved'.

4 Place the catapult and crate on the side of the water opposite the button.

5 Code the catapult's direction so it flings towards the button. Do this using the event OnCreate.

6 Test the puzzle using the play icon in the bottom right corner of the screen.

Code required to complete the puzzle

The Red Button should have identical code to the buttons in Crate Weight and Cliffhanger. The catapult should have its direction correctly coded.

Skill Development

JavaScript

Constant parameters in functions:I understand how parameters can affect the output of a function.

Changing values (numbers, strings, Booleans): I understand the difference among different types of values and can explain in what situations I might use each.

Computational Thinking

A1: Writing instructions that if followed in a given order (sequences) achieve a desired effect.

Computing Progression

PA2: I know that computers need precise instructions.
PP2: I can create a simple program.
PP3: I can run, check and change programs.

YA1: I know that algorithms are implemented on digital devices.
YP2: I can use logical reasoning to predict the behaviour of programs.

Conditional Fling

Changing values

Constant parameters in functions

Conditional Expressions A1 YA2 PP2

YA1 YP2 PP3

Description
Conditional Fling combines the IF statements of Young's Buttons with the flinging crates of Crate Throw. We suggest you complete those puzzles first.

Design Tip
Ensure that the catapults fire the correct distance and direction to land on the buttons.

Steps to complete

1 Create a narrow waterway to fling the crate across. Use the Question Mark icon if you are unsure how this should look.

2 Place the Purple Buttons on one side of the water.

3   Starting with Button A, code an IF statement that requires both buttons to be pressed down to solve the obstacle. Then repeat for Button B.

4   Code the obstacle to be 'unsolved' when the buttons are depressed (same as for Red Buttons).

5   Place the two catapults and two crates on the side of the water opposite the buttons.

6   Code the direction of Catapult A so it flings towards the button. Do this using the event OnCreate. Repeat for Catapult B

7   Test the puzzle using the play icon in the bottom right corner of the screen.

Code required to complete the puzzle

To create an IF statement for each button you will need to write the code shown below.

IF statements are found under the languages tab when in the Sequencer. Once placed in the Sequencer the condition, in this case ButtonB.pressed, can be dropped inside the IF statement.

The catapults should have their direction correctly coded.
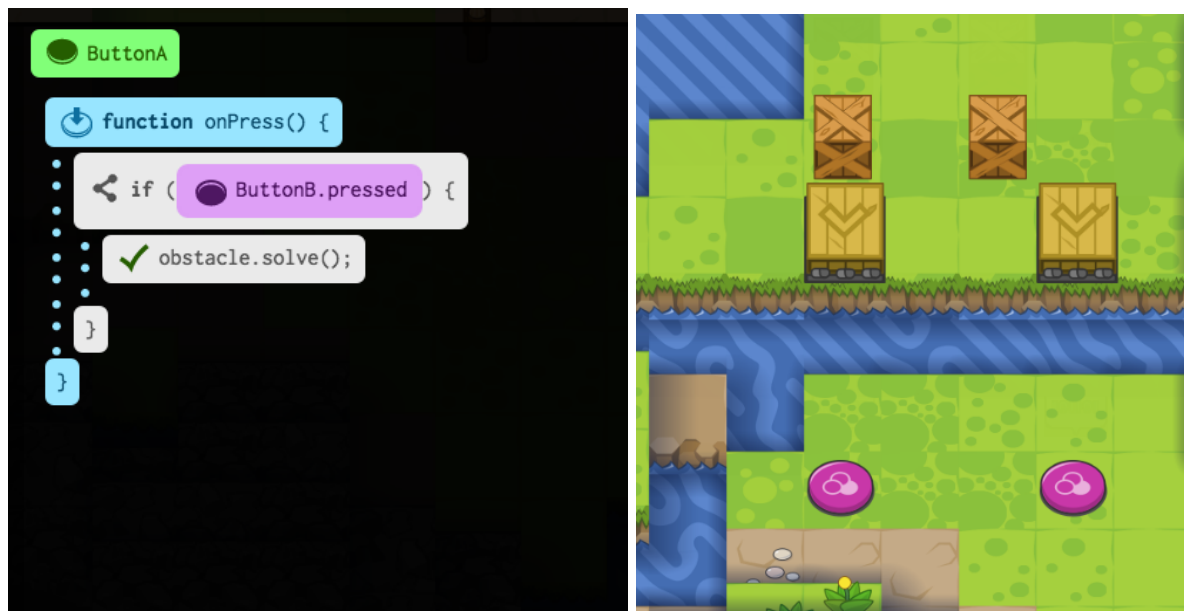
Skill Development

JavaScript

Changing values (numbers, strings, Booleans): I understand the difference among different types of values and can explain in what situations I might use each.

Conditional expressions: Using my knowledge of comparison operators, values, and booleans, I can write a conditional expression that results in true or false.

Constant parameters in functions:I understand how parameters can affect the output of a function.

Computational Thinking

A1: Writing instructions that if followed in a given order (sequences) achieve a desired effect.

Computing Progression

PP2: I can create a simple program.
PP3: I can run check and change programs.

YA1: I know that algorithms are implemented on digital devices as programs.
YA2: I can design simple algorithms using loops, and selection i.e. if statements.

YP2: I can use logical reasoning to predict the behaviour of programs.