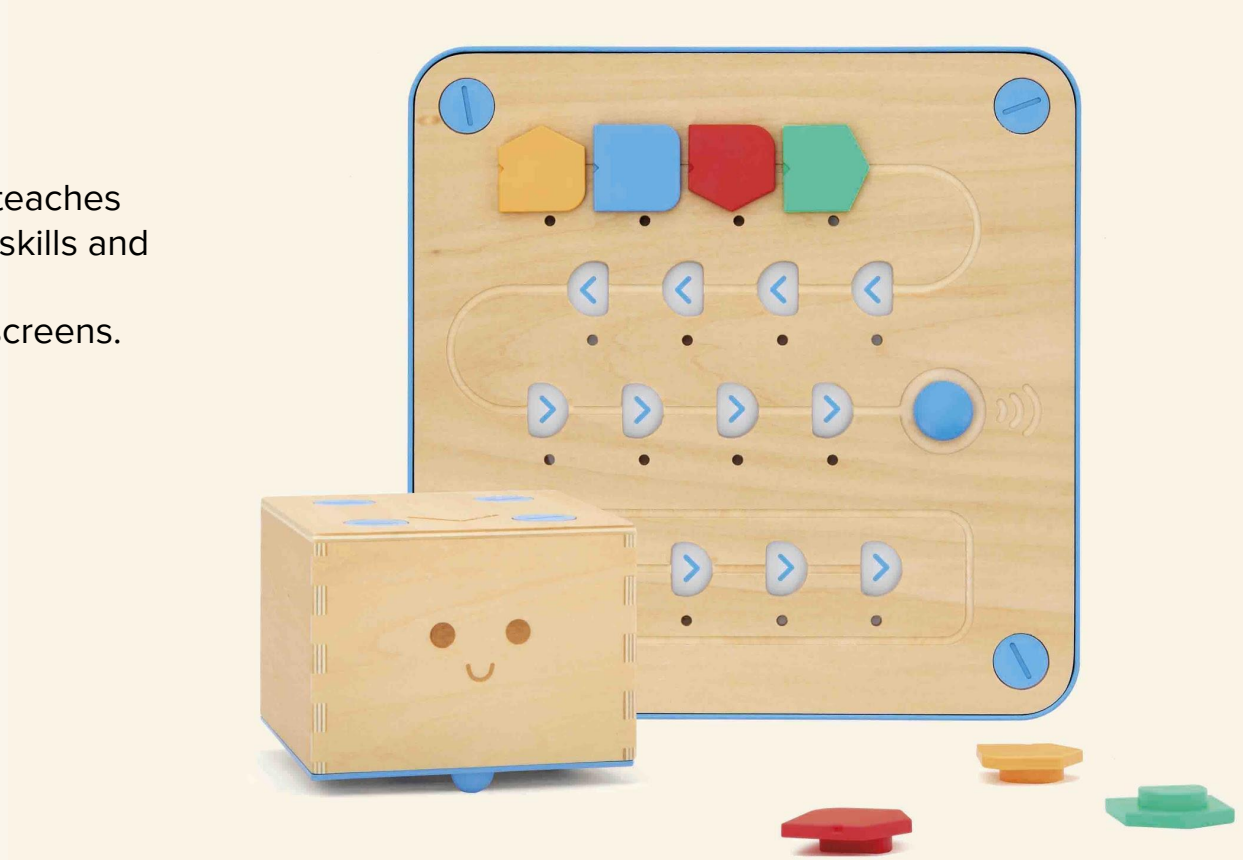

PRIMO

Training Overview

- Quick Company Overview
- The Importance of Early Learning Coding
- **Why Cubetto**
- **Introducing Cubetto**
  - In the box
  - Add ons
  - Competitors
- **Teacher Resources**
- **Programs**
  - Deal Registration
  - Demo Program
  - Cubetto Bundles

Say Hello to Cubetto!

Cubetto is a robot that teaches computational thinking skills and the basics of computer programming, without screens.



The Importance of Early Learning Coding


In the next 20 years as many as
47% of jobs in the USA will
become completely automated.


Coding is a new kind of literacy
for the 21st century.

Why Should Kids Learn Coding and Computational Thinking?

1. Coding nurtures creative expression
1. Programming demystifies tech
1. It teaches problem solving and persistence
1. Children learn by thinking about doing
1. Children learn to think about thinking

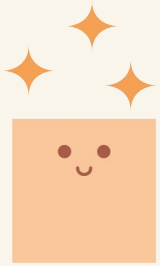
Why Cubetto?

Cubetto Is Unique

Screenless

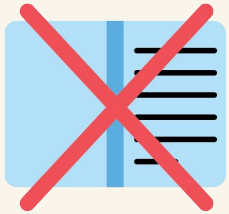
Cubetto fosters hands-on, kinaesthetic learning: removing the distractions of screens and apps and providing richer sensory information to the child.

Plug & Play



Cubetto works out of the box, with little to no prep or prior experience required to teach with it.

No Reading Required

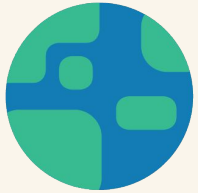


While some young kids know the alphabet and its sounds back to front, others are still learning. Cubetto requires no reading or writing skills to be effective.

Pre-K and Up

From its tactile, wood construction to its colorful, tangible coding blocks, Cubetto is designed specifically for young children and early learning.

Inclusion Friendly



Cubetto is gender neutral, hands-on, and can be used by children irrespective of reading / writing ability or language barriers.

Why do SCHOOLS need Cubetto?

- Schools are under pressure to deliver high digital literacy standards in Early Childhood
- Early Childhood teaching staff often don’t have the confidence to teach STEM subjects
- Cubetto is a proven tool, used by teachers worldwide with quantifiable results

Why do TEACHERS need Cubetto?

- Teachers need tools that are simple to use and engaging while also delivering real teaching
- Teachers need resources that are flexible to differentiate learning across abilities
- “Who’s got the time for this?” Cross-curricular teaching is a must

Why do CHILDREN need Cubetto?

- Evidence shows that early childhood is a formative time to develop computational thinking skills.

- Children learn best through play, and especially through hands-on rather than abstract activities.

About Cubetto

How does Cubetto work?

Children learn best through doing and exploring.

The Cubetto Playset is designed to represent key concepts of computer programming in a tangible way.

Its intuitive design fosters enquiry-based learning and teamplay

Introducing Cubetto

In The Box

Cubetto

Made of tactile and hard-wearing wood he’s your child’s guide into the world of coding. Screenless, friendly and ready to play straight out of the box.

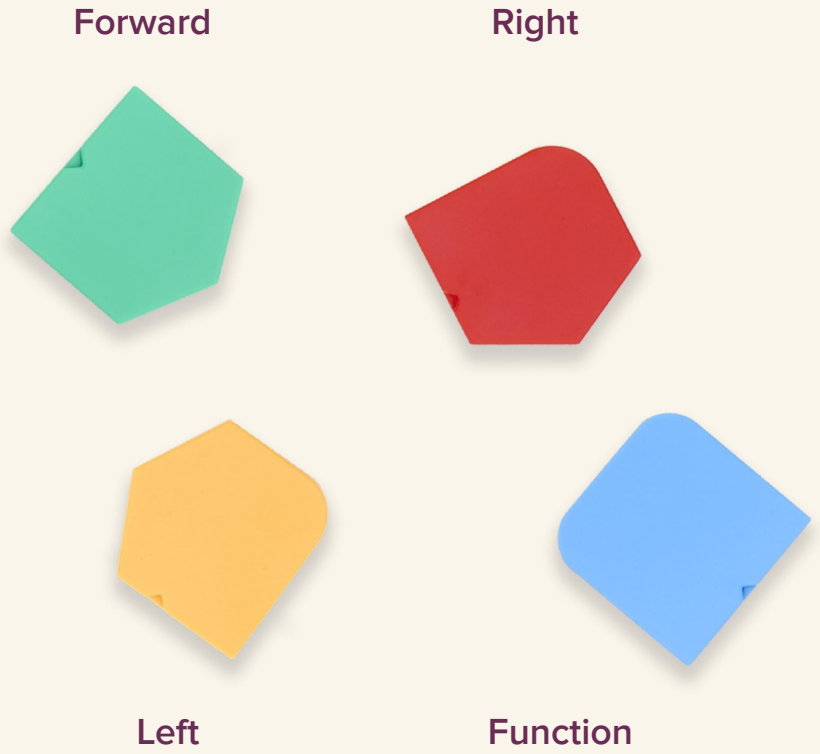
In The Box

Coding Blocks

A tactile coding language you can touch and manipulate. Each block represents an individual action.

Each playset will contain the perfect starter set:

6x Forward
4x Left
4x Right
2x Function



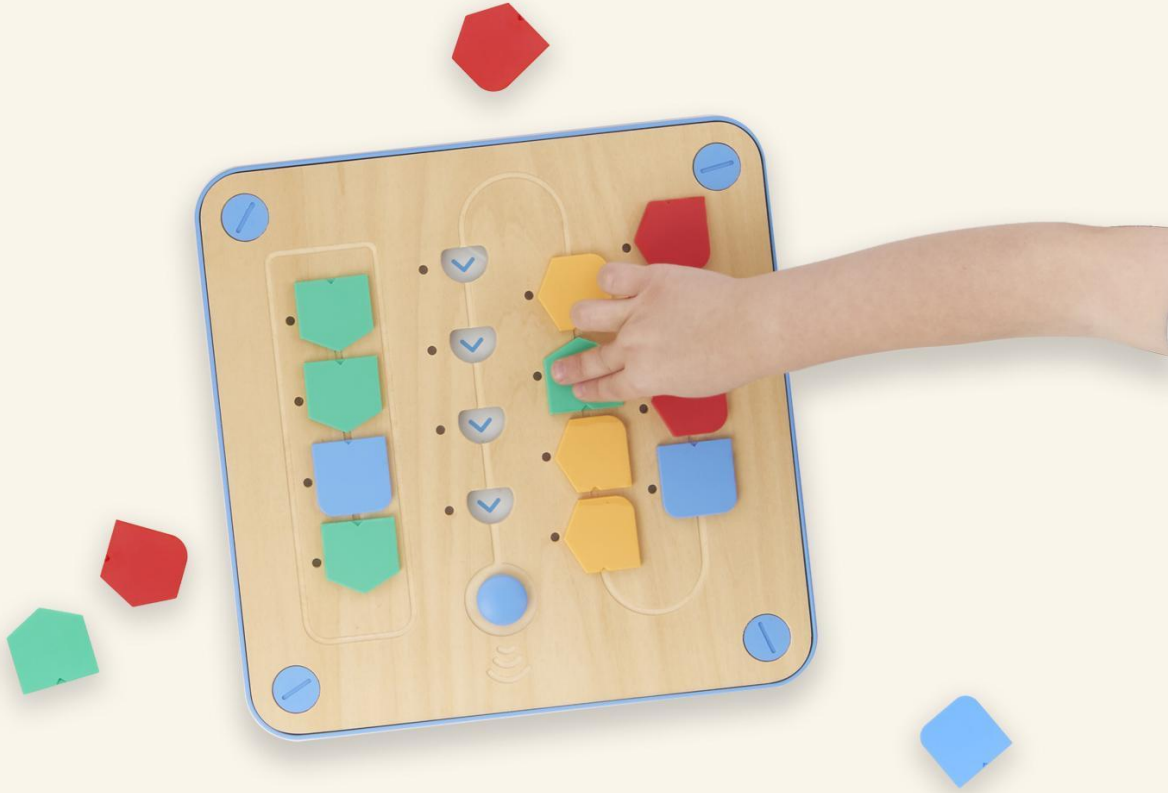
In The Box

Control Board

Place the blocks in the queue line on the board to tell Cubetto where to go. Hit the blue button and Cubetto executes your program.

Each Cubetto pairs with any board, simply turn on each pair, one at a time.

Each playset uses standard, easy-to-find AA batteries



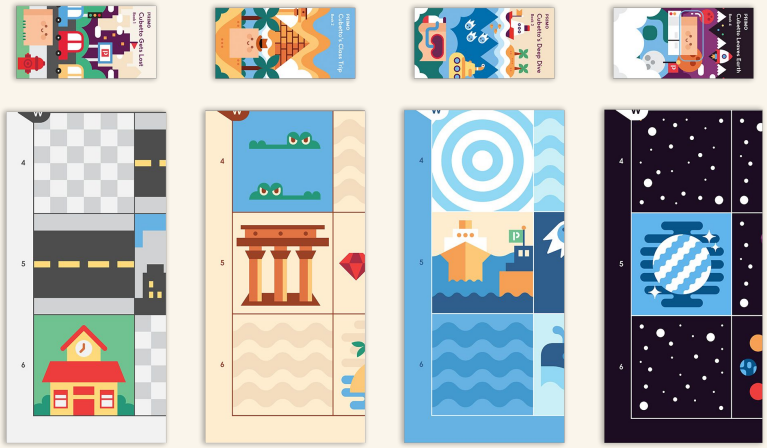
In The Box

Use the **World Map** to create your own adventures, and follow the included **Story Book** for progressive challenges and activities.

“A treasure hunt encourages kids to think logically and engage in real-time debugging”

Add-ons

Thematic Adventure Packs (Map + Storybook)



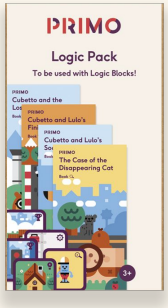
A perfect STEAM resource, each Adventure Pack comes with a Map and a Story Book.

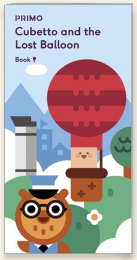
The story books develop an educational narrative, paired with specific challenges that take your students on epic coding adventures.

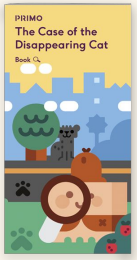
- **Big City** - A fast-paced ride through a bustling metropolis.
- **Ancient Egypt** - A historical journey to the Great Pyramids.
- **Blue Ocean** - A submarine voyage below the waves.
- **Deep Space** - A trip to the fringes of our solar system.

Add-ons

Logic Pack







Perfect for 1st grade and up, Logic Pack includes 4x storybooks and flashcards, each focusing on one of the cornerstones of Computational Thinking:

- Cubetto and Lulo's Soapbox Adventures (Planning)
- Cubetto and Lulo's Finish Line Frenzy (using Variables & Pattern Recognition)
- Cubetto and the Lost Balloon (Abstraction)
- The Case of the Disappearing Cat (Prediction)

Requires Logic Blocks

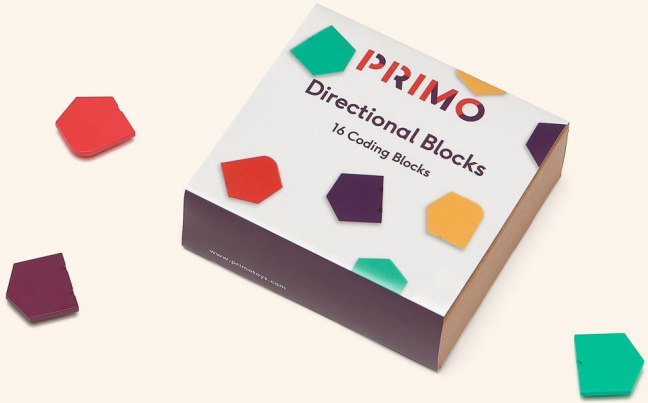
Add-ons

Directional Blocks

perfect for younger students they are your introduction to directional coding

4x Forward - 4x Backwards

4x Left - 4x Right



Right - Turns Cubetto right 90°

Forward - Moves Cubetto forward 1 square

Left - Turns Cubetto left 90°

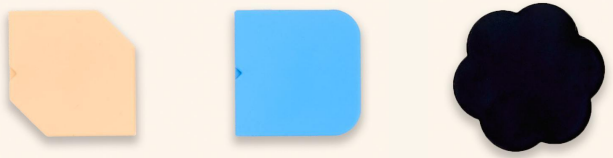
Backwards - Moves Cubetto backwards 1 square

Add-ons

Logic Blocks

required for the Logic Pack, the logic blocks take student's computational thinking skills up a notch by introducing more complex logic

4x Random - 4x Negation - 4x Function



Negation - Executes the opposite of the following block

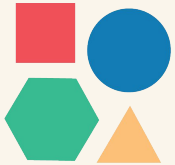
Function - recalls a function string

Random - Executes a random move

Cubetto in the Classroom

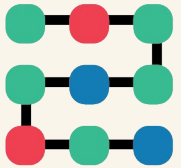
Decomposition

Abstraction

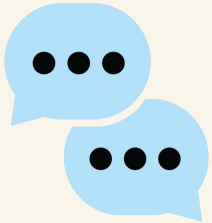


Action Causality

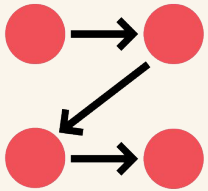
Pattern Recognition



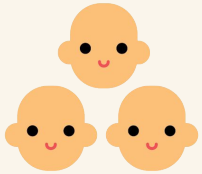
Communication



Physical Exercise

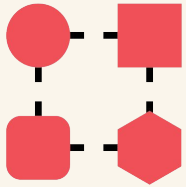


Social / Emotional



Mathematics

Logical Reasoning



Learning Outcomes

Cross Curricular Versatility

Cubetto fosters student learning in key areas like social-emotional, creative thinking, STEM, and Common Core.

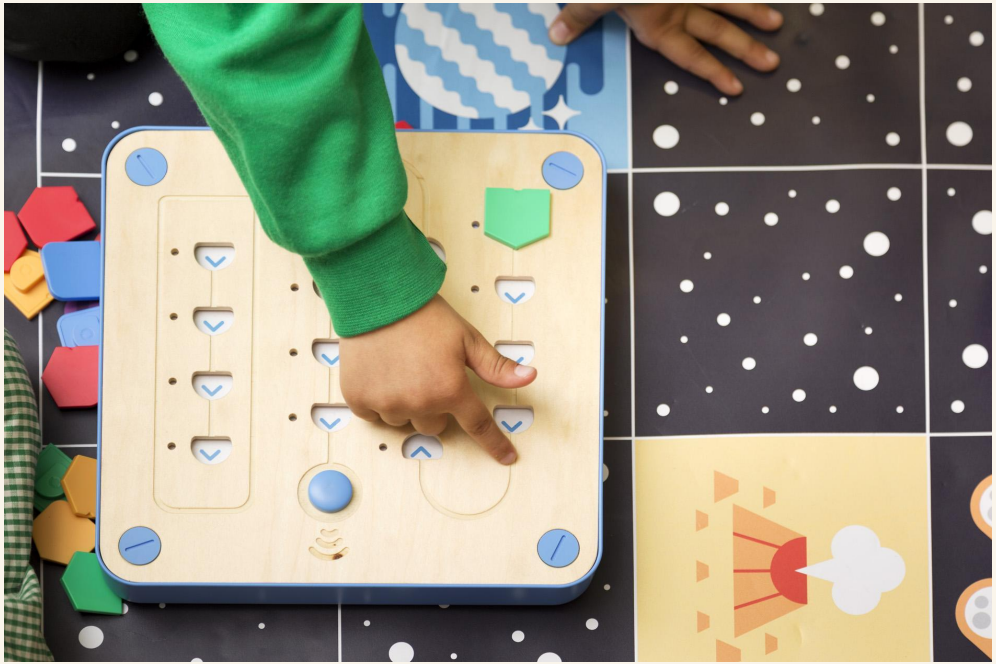
Programming

The tangible blocks have the potential and scalability of any real procedural programming language.

Learning Outcomes

Algorithms

- Algorithms are sets of precise instructions that form a program.
- Cubetto’s blocks are a physical representation of an instruction that combine to create a program.



Learning Outcomes

The Queue

- Instructions in programs are executed following a precise order.

- Cubetto represents this physically using the Control Board and Coding Blocks.

Learning Outcomes

Recursions

Create a subroutine by “packaging” a sequence in the function line, and call it in the queue with a blue block when you need it.

Function Block - Executes a function

Learning Outcomes

Debugging

Fixing mistakes is as easy as swapping a block if Cubetto doesn’t arrive where he needs to.

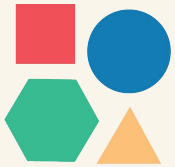
Deeper Learning

Learning with Cubetto is not all about programming skills. Students are exposed to a variety of subtle computational thinking challenges developing high thinking skills such as:

Decomposition

The process of breaking down a problem into smaller manageable parts.

Abstraction

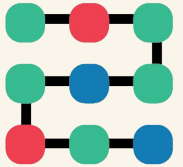


The ability to think about a solution to a problem without trying it out first.

Action Causality

If I do this here, then that will happen there. Integral to how programming works.

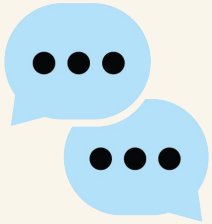
Pattern Recognition



Finding similarities that can help us solve more complex problems more efficiently.

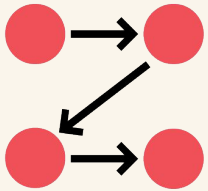
Other Key Learnings

Communication



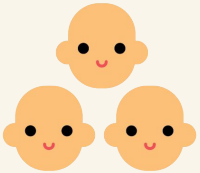
Children practice listening through a range of stories and narratives and develop their own explanations and narratives

Physical Exercise



Children master control and coordination in large and small movements around the playset.

Social / Emotional

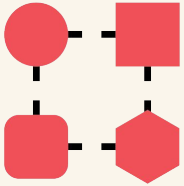


Children become confident by trying new, open ended activities that remove “wrong” outcomes, and easily encourage group work.

Mathematics

Children add and subtract blocks from a sequence. They solve problems, including doubling, halving and sharing to get Cubetto from A to B.

Logical Reasoning



Children use technology purposefully to create, organise, store, manipulate and retrieve meaningful sequences.

Teacher Resources

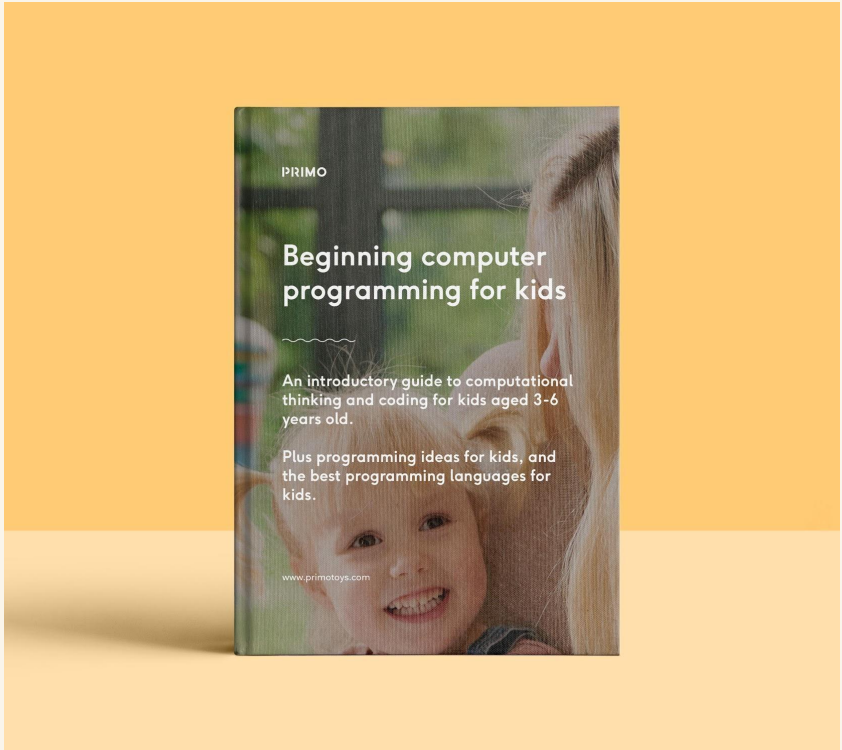
Introductory Resources

Guide to Coding for Kids

Aimed at parents, this ebook helps them breaks down the basic concepts of STEM for kids, offers tips on learning to code with the little ones using fun games, and help demystify readers' tech fears.

Teacher's Guide

New technology can sometimes be overwhelming to understand and adopt. Created by educators, for educators this guide makes it simple for teachers to integrate Cubetto and tangible programming in their classrooms



Lesson Plans

Lesson Plans Archive

2 years worth of content split across preschool and Kindergarten (100+ individual lessons)

NGSS, CC and ISTE standards alignment coming soon!

	Lesson 1	Lesson 2	Lesson 3	Lesson 4
NC Computing Objectives	To control a digital device	To use logical reasoning to predict behaviour of simple programs	To create a simple program	To debug a simple program
Outcomes	• I can make Cubetto move • I can say one thing that Cubetto can do and can't do	• I can predict what a block will do • I can say how Cubetto feels and what its face will show	• I understand that an algorithm is a set of instructions in order • I can write a simple storyboard	• I can debug a simple algorithm • I can identify 2D shapes
Cross-curricular Subject	English	PSHE	English	Maths
Computational Thinking	Tinkering	Logic, Creating	Decomposition, Collaborating	Algorithms, Debugging
Main Activities	Cubetto's Diary 1. Open Cubetto Unit (with support) and explore their insides. Talk about what you can see. 2. Meet Cubetto and find out what it can and can't do, and what surprises you. Write down in a simple diary.	Cubetto's Feelings 1. Move Cubetto to somewhere it feels happy then sad, changing its face when it arrives. Use: "If I use the _ block, Cubetto goes _". 2. Discuss emotions for Cubetto. Finish the sentence: "If Cubetto is in the _ square, then it feels _".	Cubetto's Holiday 1. Discuss importance of storytelling in order to learn algorithms. Draw a storyboard in three parts to tell a holiday story. 2. Write a simple algorithm to tell Cubetto's holiday story and test out each other's programs.	Cubetto's Dance 1. Debug a series of algorithms in a group to reach a chosen square containing a shape. 2. In pairs one writes an algorithm to make Cubetto 'dance' to a square with a 2D shape. Hiding one step, the other debugs it.
Challenge	Talk about how Cubetto moves	Can you talk about what the blue block does? Why is it helpful?	Can you take a longer journey to get to the castle? Write the algorithm.	Can you write an algorithm to make Cubetto dance in a circle?
Creative Play	Make Cubetto a costume.	Role play a time when you are happy or sad.	What happened next to Cubetto? Role play the story.	Draw a castle or boat using 2D shapes.
Resources	Diary template, 2p coins / play screwdrivers	Face templates, Sticky tack, Coloured pens, Mini whiteboards	Storyboard template, Primo board template, Whiteboard pens	Primo board template, Algorithms to debug, 2D shapes , Music (optional)
Assessment	Diaries, Photos, Verbal statements, Observation	Face templates, Photos, Verbal statements, Observation	Storyboards, Photos, Verbal statements, Observation	Photos, Verbal statements, Observation

Lesson 4: Cubetto's Dance (1 of 2)

Cross-curricula Area: Maths

NC Objectives	Outcomes	Resources Needed	Prep Needed	Resources Provided	Key Vocabulary
To debug a simple program	• I can debug a simple algorithm • I can identify 2D shapes	• Coloured whiteboard pens • 2D shapes • Music (optional)	• Check batteries • Set up algorithm example on Primo Board	• Primo Board template	• 2D shape names • Algorithm • Debugging • Prediction

Computational thinking concept

Algorithms

Computational thinking approach

Debugging

Teacher-led Introduction

1. Ask for volunteers to make different 2D shapes using their bodies (e.g. puff up face and make a big circle with their arms.)
2. Show the map and ask pupils where they can see a circle, triangle, square and rectangle. Ask: Can you see other shapes?
3. Explain that today the pupils will write algorithms to make Cubetto dance to places on the map that have different shapes in them. BUT there are problems with the algorithms!
4. Show the first algorithm example on the Primo Board and ask: Will this work? Why/why not? If not, how can we fix it?
5. Model pressing the action button (won't work). Model working out what is wrong and how you can fix it. Model trying again.
6. Explain that when we try to fix an algorithm that doesn't work, this is called debugging.
7. Ask: What is it called when we use what has happened before to tell us what will happen in the future? Recap prediction.

Lesson 4: Cubetto's Dance (2 of 2)

Creative Play

Draw a castle or boat using 2D shapes.

Guided Activity

1. Choose a place on the map that has circles in it: this is your starting point.
2. Choose another place on the map that has squares or rectangles in it (not too far away): this is your end point.
3. On the Primo Board and with coloured pens, write an algorithm to make Cubetto dance from the start to the end.
4. Check your algorithm works by testing it on Cubetto.
5. When you have made sure your algorithm works, rub out one of your blocks. It now needs debugging!
6. Find a partner who has finished their algorithm too.
7. Swap Boards and debug the algorithm.
8. Discuss with your partner what was missing and how you worked it out.

Independent Activity

1. Look at the first algorithm that needs fixing (e.g. Start on the boat, end on a triangle.) Ask: Where on the map can you see triangles? Where on the map does Cubetto want to get to? (In this example: start on the boat, end on the mountains).
2. Ask: Can you predict if this algorithm will work? If it won't work, what's wrong with it?
3. Ask: How can we debug this algorithm? Discuss until the group agrees on what to do.
4. Test out the algorithm with Cubetto.
5. Repeat for the other algorithm examples.

Challenge

Can you write an algorithm to make Cubetto dance in a circle? Can you make it dance forever?

Plenary and Assessment

1. Ask: What does debugging mean? What kinds of problems did we find today? How did we debug them?
2. Children share the algorithms they fixed (or created).
3. Ask: What was wrong? How did you work out what was wrong? How did you fix it?
4. Ask: What 2D shapes did we find on the map? What shapes can't we see?

Questions?