COMPUTE-IT 1

COMPUTING FOR KS3

MARK DORLING
AND GEORGE ROUSE
Series Editors

DYNAMIC LEARNING
HODDER EDUCATION
LEARN MORE

COMPUTE-IT 1

COMPUTING FOR KS3

Provisional contents

HODDER EDUCATION
AN HACHETTE UK COMPANY

Introduction

Compute-IT – A series of print and digital resources that deliver the new Key Stage 3 Computing Programme of Study

- Are you looking for authoritative and imaginative resources to help you plan for and teach the new Key Stage 3 Computing Programme of Study?
- Are you looking for lessons that deliver the key strands of the curriculum and core skills of computational thinking, planning, programming and review?
- Do you want to feel confident in assessing learning outcomes to provide you with a meaningful record of pupil attainment and progress?

If so, we have the complete solution for you.

Deliver innovative Key Stage 3 Computer Science and ICT lessons for the new curriculum with confidence – using resources and meaningful assessment produced by expert educators.

Developed by an expert author team including members of the Computing at Schools Network and the Department for Education's Teachers of Excellence in Computer Science programme, this complete offering for Computer Science and ICT at Key Stage 3 consists of:

- three student's books,
- three teacher packs, and
- a bank of digital resources and assessments delivered via Dynamic Learning an online service from Hodder Education

Together, these form a cohesive and supportive learning package structured around the key strands of Computing. Creative and flexible in its approach, **Compute-IT** makes Computer Science and ICT for Key Stage 3 easy to teach, and fun and meaningful to learn.

- Deliver the new curriculum with confidence by following well-structured and finely paced lessons along a variety of suggested routes through Key Stage 3
- Make your course engaging and interesting using a range of files provided in different programming languages
- Ensure progression throughout the course with meaningful tasks underpinned by unparalleled teacher and student support – including exemplification of expected outcomes, answers to all activities provided in the student books and as digital files, and assessment guidance
- Assess learning outcomes confidently with ready-prepared formative and summative tasks covering the statements outlined in the new programme of study
- Ensure your students are well prepared for their chosen Key Stage 4 course by monitoring their work against defined Learning Outcomes and Progression Pathway statements*

***A note on assessment and the National Curriculum Programme of Study**

Hodder Education has teamed up with leading Computer Science educators to define learning outcomes that are mapped to our own Progression Pathway statements and the Computing Programme of Study. This provides teachers with much-needed support when planning their delivery of the curriculum and ensuring that assessment is meaningful for both educators and learners.

Our Series Editors

Mark Dorling has extensive secondary teaching and industry experience. His work including the Digital Schoolhouse project (which featured as a case study in the Royal Society Report) and the 2013 TES award for ICT has gained him international recognition. Mark was a member of the DfE working party of experts responsible for recommending a programme of study as part of the DfE ICT National Curriculum review. He currently sits on a DfE Initial Teacher Training Expert Computing Group and is National CPD coordinator for Computing at School.

George Rouse has degrees in mathematics and in technology. After spending time in industry, mostly with British Steel, George moved into teaching mathematics and computer science spending a large part of his career at King Edward VI Camp Hill Boys' School in Birmingham. George has written several books for ICT and Computer Science. He is also an experienced senior examiner of Computing and ICT at GCSE and A level.

The Publisher would like to thank the following for permission to reproduce copyright material:

Photo credits: **p.3** *tl* © Dennis Hallinan / Alamy, *tr* © Classic Image / Alamy, *bl* © REUTERS/Olympic Delivery Authority, *br* © REUTERS/Olympic Delivery Authority; **p.4** *tl* © Sharpshot – Fotolia.com, *tr* © Abigail Woodman, *bl* © Smileus – Fotolia.com, *br* © igor – Fotolia.com; **p.5** © Peter Horree / Alamy (Painting by Theo van Doesburg Countercomposition XVI); **p.8** © Arto – Fotolia.com; **p.9** *tl and b* © Scratch (http://scratch.mit.edu/), *tm* © Hopscotch Technologies, (gethopscotch.com), *tr* © Alice Project, (www.alice.org); **p.10** *t* © dbimages / Alamy, *b* © Tom Grundy / Alamy; **p.11** *tl* © hollygraphic – Fotolia.com, *tr* © Robert Preston Photography / Alamy, *bl* © Python Software Foundation, *br* © Ruby / Yukihiro "Matz" Matsumoto http://www.rubyist.net/~matz/.

Orders: please contact Bookpoint Ltd, 130 Milton Park, Abingdon, Oxon OX14 4SB. Telephone: (44) 01235 827720. Fax: (44) 01235 400454. Lines are open 9.00–17.00, Monday to Saturday, with a 24-hour message answering service. Visit our website at www.hoddereducation.co.uk

Cover photo © adimas – Fotolia

Unit 3 Drawing and manipulating shapes

Challenge

Your challenge is to write a program that creates an artwork based on drawing and positioning shapes found in Celtic or Islamic art.

3.1 Shapes, patterns and algorithms

Computer science, art and maths

There are many ways in which computer science is linked to the world around us and there are very strong links between maths, art and computer science.

The term **abstraction** can be used in all three subjects. In computer science abstraction means finding **generalisations** by identifying common patterns in real situations.

In maths, abstraction involves considering a problem theoretically, separating it from the everyday contexts that it has been associated with in the past, so that generalisations can be used in other contexts. Abstraction, in art, describes the process of interpreting the subject rather than depicting it exactly.

Being creative and imaginative are skills that artists and computer scientists need. Just think of the creativity that goes into designing the video games you play. The characters, environments, graphics and game-play are all developed by designers and programmers working together.

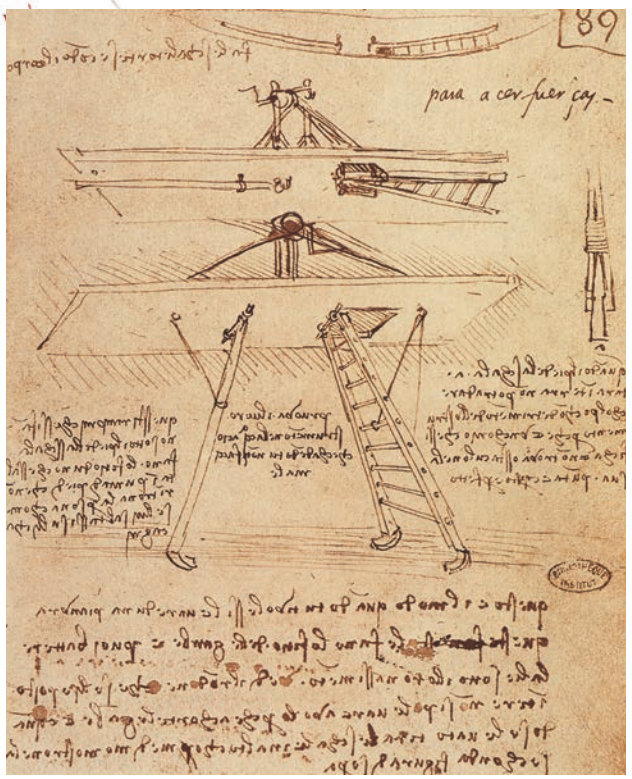
Mathematicians, artists and computer scientists all need to solve problems and find patterns in their work. For example, when trying to find a cure for a disease, scientists will look for patterns in the medical data of people who are already ill with the disease.

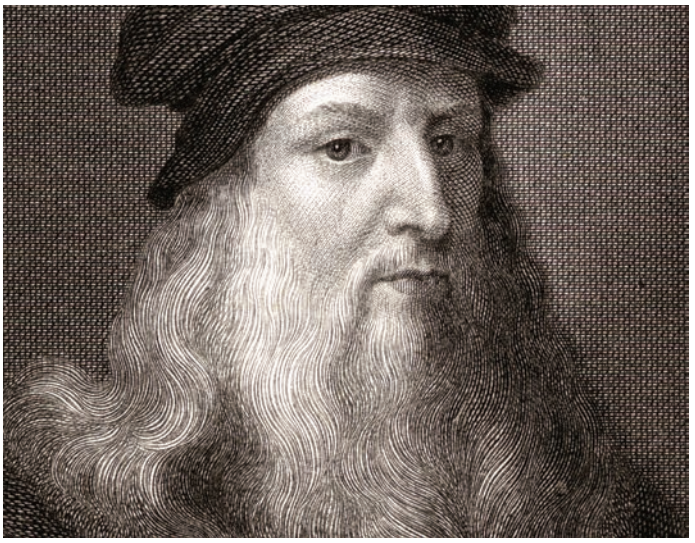
Key Term

Abstraction: Working with ideas or solving a problem by identifying common patterns in real situations, concentrating on general ideas and not on the detail of the problem itself.

Generalisation: Taking concepts used in the solution of a particular problem and using them to solve other problems that have similar features.

(You used both abstraction and generalisation when you tackled the problem of the malaria outbreak in Kitanga District in Unit 2.)

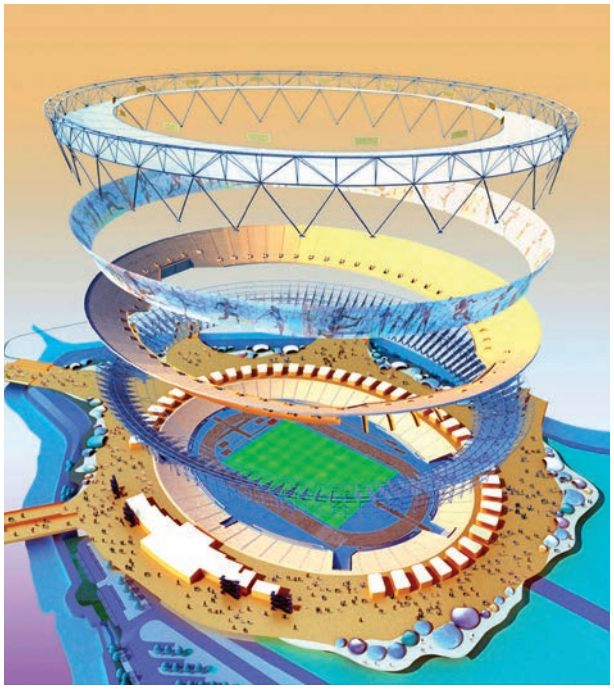




Leonardo Da Vinci was an artist, scientist, mathematician and inventor. He would probably have been a great computer scientist too, had modern technology existed during his lifetime. He was an abstract thinker and would often take ideas from one area, identify the important concepts and apply these to another area. He was applying computational thinking skills to problem solving.

Think-IT

3.1.1 In what other ways do you think computer science, art and maths are linked? The images below should provide some food for thought.



The original plans for the London 2012 Olympic and Paralympic Stadium and a photograph of the finished stadium.

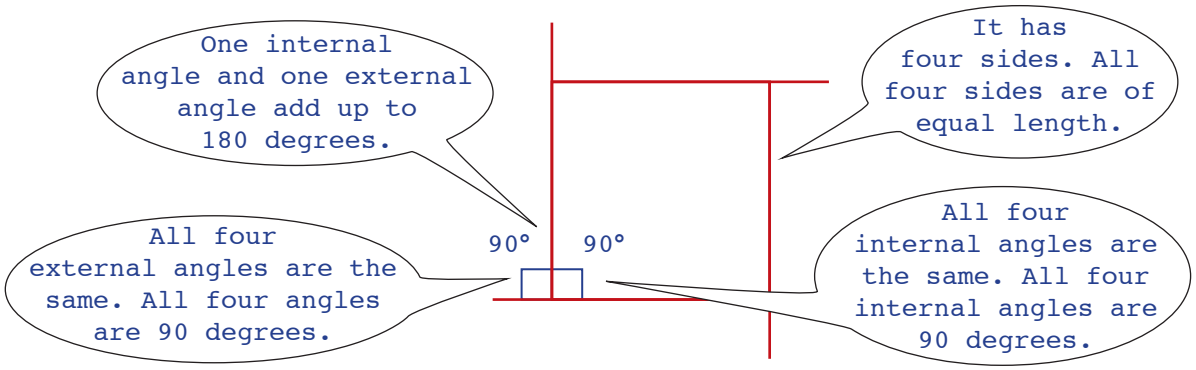
Key Term

Geometrical shapes: Shapes that are defined by a set of mathematical rules.

Shapes

Shapes, and **geometrical shapes** in particular, play an important part in maths, art and computer science.

A square is a geometrical shape. The key features of a square are:



Think-IT

3.1.2 What shapes can you see around you? Look at the objects below. Redraw them as shapes and label their features.

a

b

c

d

Patterns

Patterns also play an important part in our lives. Patterns are repeated concepts defined by rules. From number patterns to visual and scientific patterns, they are everywhere! Patterns help us to see and understand how things work.

Think-IT

3.1.3 **a)** What patterns can you recognise in these number sequences?

11, 22, 33, 44 **1, 2, 4, 8, 16**

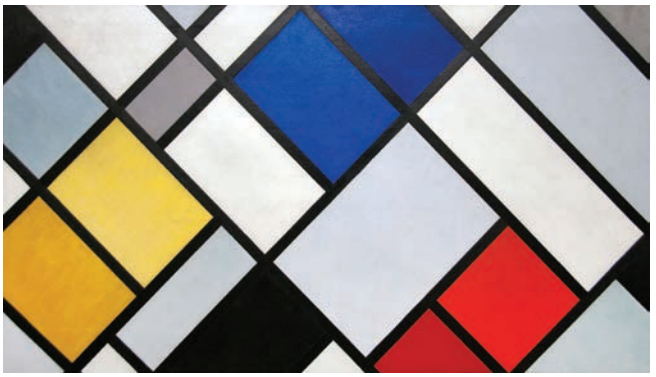
b) What are the next three values in each series?

c) What patterns can you recognise in the photographs on page 6?

d) Look around you. What repeating patterns can you see?

Algorithms

Theo van Doesburg was an abstract artist. He often explored the shape and colour of objects by representing them as geometric shapes.



Think-IT

3.1.4 What shapes can you see in van Doesburg's painting?

Van Doesburg's paintings are made up of squares, rectangles and lines of different sizes. By looking at the painting in this way, you have undertaken **decomposition**.

Decomposition is used to solve problems by breaking them into smaller parts. For example, if you want to make a packed lunch for school, you will naturally break the task down into smaller parts, making sandwiches, making a drink and adding your other favourite lunch foods. To solve the problem of creating a van Doesburg-style artwork, the challenge is how to create the squares, rectangles and lines that make up one of van Doesburg's paintings. Let's start with drawing a square.

Key Term

Decomposition: The process of breaking something down into smaller parts.

Key Term

Algorithm: A set of step-by-step instructions which, when followed, solve a problem.

Designing an algorithm

An **algorithm** is a set of step-by-step instructions which, when followed, solve a problem. We can use an algorithm to help us draw our square.

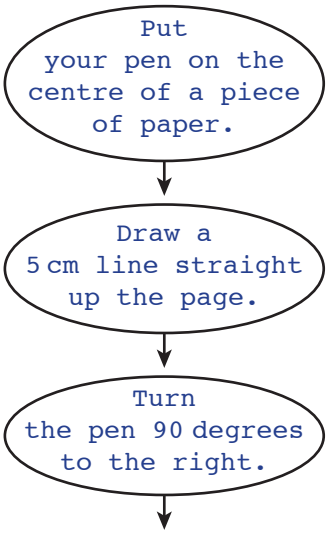
Plan-IT

3.1.5 **a)** Imagine you have to give an alien instructions on how to draw a square. What would you tell the alien to do? Write down your instructions and give them to a friend to try out.

Hint: Use the following as a starting point. Can you improve on these instructions?

- Put your pen on the centre of a piece of paper.
- Draw a 5 cm line straight up the page.
- Turn the pen 90 degrees to the right.

b) Did your friend manage to draw a square? Do you need to adjust your instructions?



Key Term

Coordinates: A set of values used to show an exact position. In two dimensions we use x and y values, where x is the distance across the page and y the distance up the page.

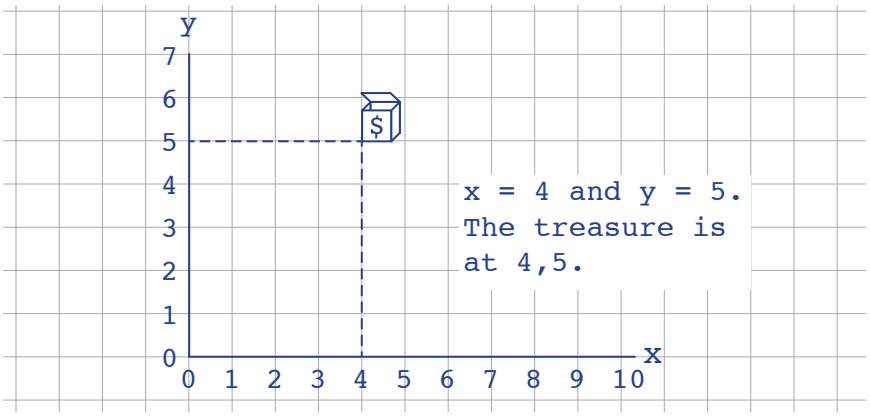
Refining an algorithm

Van Doesburg's paintings have lots of shapes in different places. Each shape has a different location on the canvas, but how do we specify that location?

Mathematicians use **coordinates** to identify a specific location by giving two values:

- x tells you where the location is horizontally
- y tells you where the location is vertically.

A computer understands coordinates, so it is possible to tell a computer where to start drawing a shape using x and y values to specify the starting point.



Plan-IT

3.1.6 Plot an *x* axis and a *y* axis on a piece of squared paper. Number each axis from 0 to 20. Draw three squares in different locations and in different colours on the grid.

Write an algorithm for your alien telling it how to draw the squares. How are you going to tell the him the colour to use for which square and when to pick up and put down the pen?

Using iteration in algorithms

It is important that the writing of algorithms is done as effectively as possible. Making an algorithm short and expressing it clearly helps us to describe the pattern of repeated actions more effectively. One of the ways you can make an algorithm shorter is to repeat instructions; to use **iteration**. Look at the two examples of an algorithm on the right. The algorithms are basically the same, but the second one is shorter because we say repeat the left right pattern five times rather than writing it all out five times.

Without iteration

```
Put right foot forward
Put left foot forward
Put right foot forward
Put left foot forward
Put right foot forward
Put left foot forward
Put right foot forward
Put left foot forward
Put right foot forward
Put left foot forward
```

With iteration

```
Repeat the following
  five times:
Put right foot forward
Put left foot forward
```

Key Term

Iteration: Using repetition of a process to create a more efficient solution.

Plan-IT

3.1.7 Look at the instructions for drawing three squares that you wrote for Plan-IT 3.1.6. Can you use iteration to make your algorithm shorter and clearer?

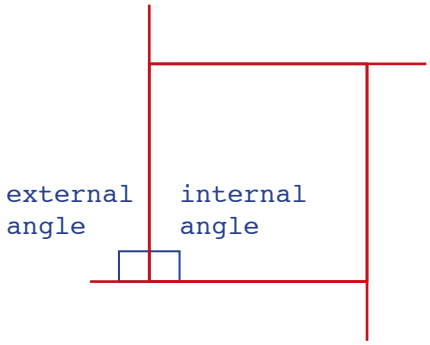
3.1.8 To draw a square you need to turn through 90 degrees each time you need to turn a corner. This is the external angle.

a) What are the external angles – the angles you must turn through when drawing the shape – for:

- a regular triangle
- a regular pentagon
- a regular hexagon
- a rectangle?

b) What would a shape with 360 sides look like?

c) Write an algorithm for drawing a triangle, a rectangle, a pentagon and a hexagon.



3.1.9 On paper draw your own van Doesburg-style artwork, based on geometric shapes. Later, you will be creating algorithms to enable a computer to draw your design, so think about the shapes, patterns and colours you use and where they will appear on the page. For example, will you use squares, rectangles and lines like van Doesburg or will you include other shapes in your design? Will you stick to bold, primary colours, or introduce other colours into your artwork? You can be as creative as you like!

3.2 Drawing and manipulating shapes using graphical programming software

Paired programming in action

Unit 3.1 explored van Doesburg's artwork and how to create an **algorithm** for drawing a square using **iteration**. Now we are going to find out how to use graphical programming software to create geometrical shapes. We can use **decomposition** and **abstraction** to do this.

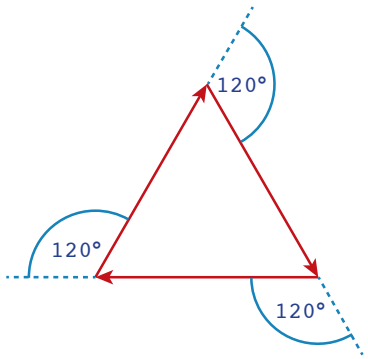
Paired programming

'Paired programming' is a term used in industry to describe a method of programming where two programmers work together on one computer. One programmer – the driver – writes the code, while the other – the observer – reviews each line of code (the technical name for the instructions given to the computer) and debugs it (the name given to the process of correcting code), developing ideas and generally looking for improvements.

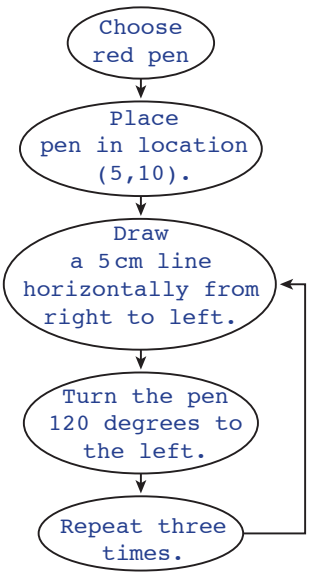
Plan-IT

3.2.1 **a)** In Plan-IT 3.1.9 you drew your own van Doesburg-style artwork. Now, list the geometrical shapes in your drawing.

b) Using paired programming, write an algorithm for each geometrical shape. For example:



1. Choose red pen
2. Place pen in location (5,10).
3. Draw a 5cm line horizontally from right to left.
4. Turn the pen 120 degrees to the left.
5. Repeat three times.



From algorithm to graphical programming

A computer will not be able to understand the plain English algorithms you have written. It needs you to program in a way that 'translates' your algorithms into a language it can understand. You can do this using graphical or text-based programming software.

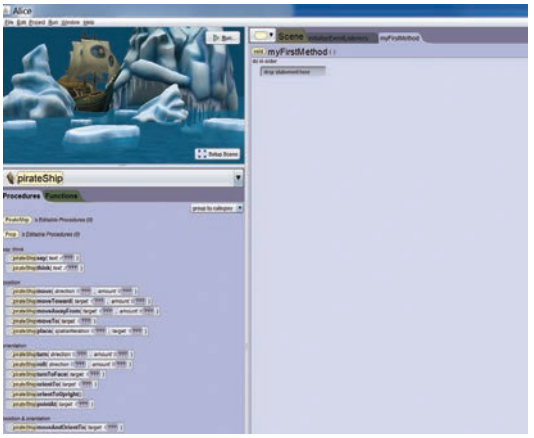
Graphical programming represents elements visually rather than textually, often using graphics in a drag-and-drop interface. Graphical programming software is a good place to start learning how to program, before moving on to writing program with a text-based programming language.

Key Term

Graphical programming: a programming language that allows users to create programs using graphics rather than text.

Scratch blocks

Daisy the Dinosaur



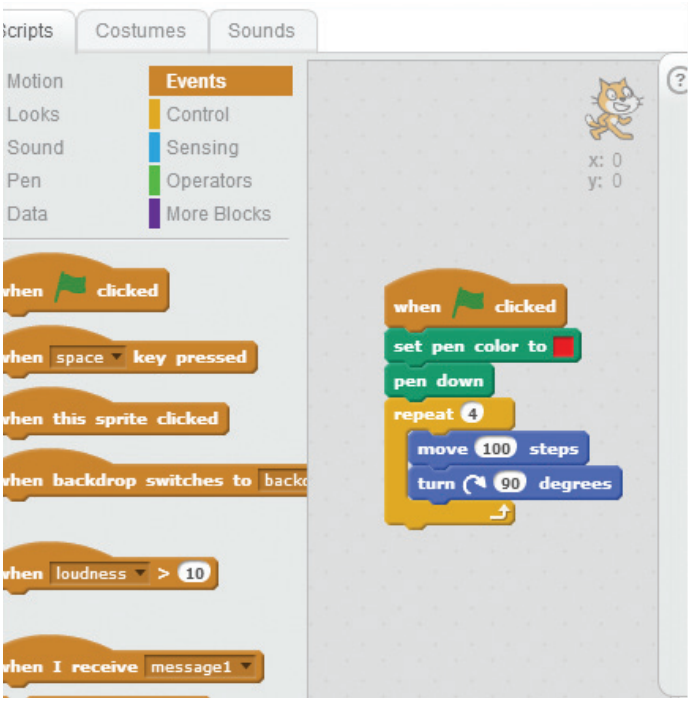
Alice

Compute-IT

3.2.2 a) Using paired programming, convert the algorithms you have created into your chosen graphical programming language.

Hint: Have you remembered to use **iteration** to write your algorithm?

In Scratch, this can be done through using the 'Repeat' block which you can use to specify how many times an instruction should be repeated.

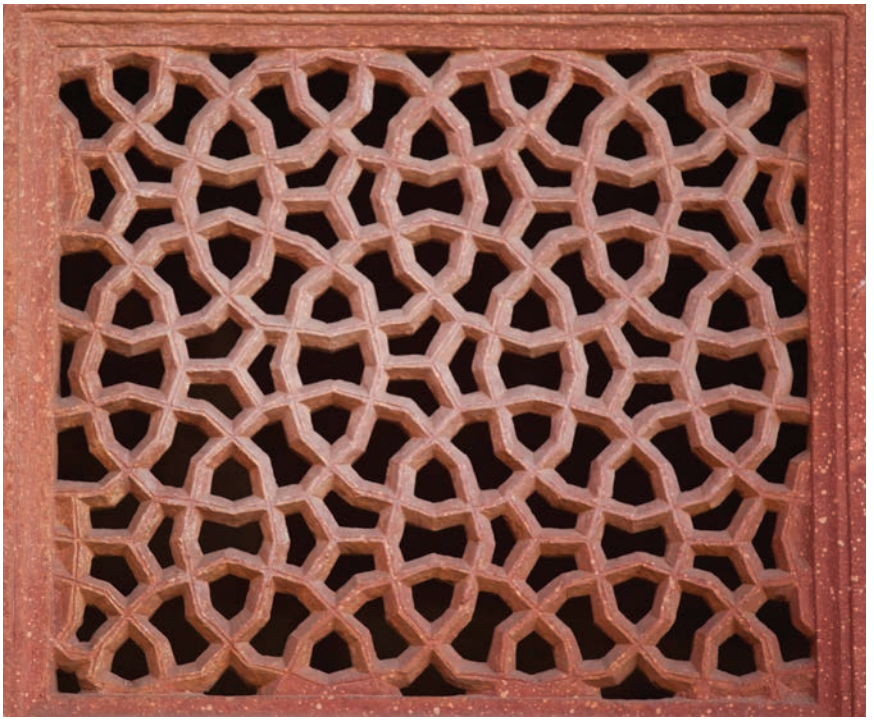


b) Write a program that will position the shapes automatically to create a short animation.

3.3 Drawing and manipulating shapes using text-based programming software

Celtic and Islamic art

Celtic art and Islamic art use patterns and shapes to represent the natural world in an abstract way.



Examples of Islamic and Celtic objects

Think-IT

3.3.1 Both Celtic and Islamic art make use of pattern and shape, but in a very different way to van Doesburg. What similarities and differences can you see between the two different artworks below and the van Doesburg painting on page 7? Consider the shapes used and the way they are positioned.

An example of Celtic art

An example of Islamic art

Text-based programming languages

Text-based programming requires the user to write code in the form of a sequence of text-based instructions into the computer to create a program, rather than dragging and dropping graphical elements when you use graphical programming software. The program controls a 'turtle', or cursor, on the screen.

Text-based programming languages are used to create programs that perform many different tasks. Most of the interactivity you experience on websites and mobile apps is created using text-based programming languages such as:

Key Term

Text-based programming: a programming language that requires the user to write code in the form of a sequence of text-based instructions into the computer to create a program.

Python

C++

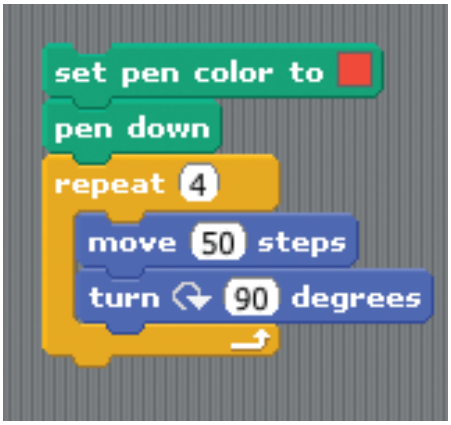
Ruby

Writing a program to draw shapes using a text-based programming language

Already knowing how to create a program to draw geometrical shapes using a graphical programming language helps when it comes to doing the same thing using a text-based programming language.

Think-IT

3.3.2 Look at the two sets of instructions below, and match the graphical instructions to the text-based instructions.




```
SetPenColor 3
PenDown
Repeat 4 [forward 100 right 90]
```

The graphical programming software used in the example is Scratch and the text-based programming language is Logo.

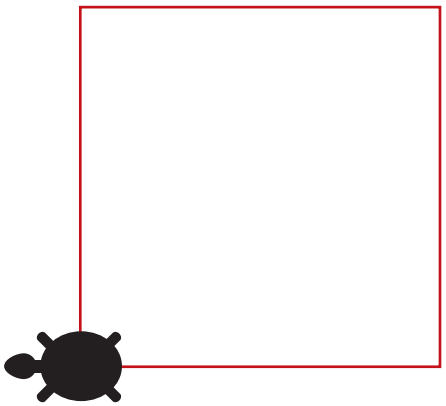
Plan-IT

3.3.3 Write algorithms for each of the following shapes:

- A red square
- A yellow triangle
- A green pentagon
- A shape of your choice

Most programming languages allow you to use iteration to clarify the processes being used. For example, we can create a new instruction to draw a square and name it 'square'. Then we can re-use that instruction elsewhere in our program.

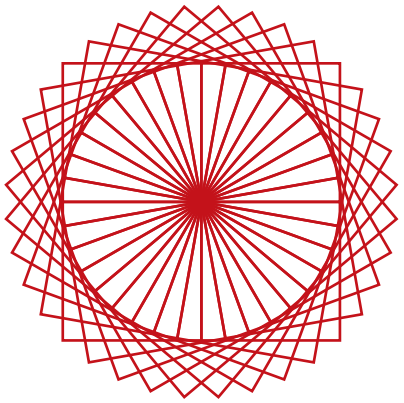
In LOGO we can define the instruction 'square' as follows:



```
to square
repeat 4 [forward 50 right 90]
end
```

We can now use the instruction 'square' more than once in our program. For example:

```
repeat 36 [right 10 square]
```



Compute-IT

3.3.4 Using a text-based programming language, enter the code required to draw the shapes you wrote algorithms for in Plan-IT 3.3.3

Challenge

Do you remember the challenge at the beginning of the unit, to create an artwork based on drawing and positioning shapes found in Celtic or Islamic art?

Plan-IT

3.3.5 a) Design an artwork based on drawing and positioning shapes found in Celtic or Islamic art.

b) Create an algorithm for your artwork. Use paired programming, as well as **abstraction**, **decomposition** and **iteration**.

Compute-IT

3.3.6 Using a text-based programming language, enter the code required to draw your artwork.

COMPUTE-IT 1

COMPUTING FOR KS3

The Compute-IT series has been edited by two experienced subject leaders with years of experience in developing curriculum programmes of study and awarding body specifications, and resources for use by teachers and students.

Developed by an expert author team including members of the Computing at Schools Group and the Network of Excellence Master Teachers programme, this complete offering for Computer Science and ICT at Key Stage 3 consists of three student's books, three teacher packs and a bank of digital resources and assessments delivered via the Dynamic Learning website. Together, these form a cohesive and supportive learning package structured around the key strands of Computing.

Computational thinking, creativity and flexibility lie at the heart of the series' approach, making Computer Science and ICT for Key Stage 3 easy to teach, and fun and engaging to learn.

Each unit in the student books provides a sound basis for the development of computational thinking skills and features activities that are designed for use in class or as homework. The topics covered are developed further through practical activities and digital files provided via the accompanying Teacher Packs and Dynamic Learning website. All of these are supported via comprehensive lesson plans, guidance on assessing students' work and mark schemes.

The series also provides a perfect foundation for students choosing to study GCSE Computer Science.

HODDER EDUCATION

www.hoddereducation.co.uk

ISBN 978-1-471-80192-1