

# SCHOOL-SCOUT.DE



Unterrichtsmaterialien in digitaler und in gedruckter Form

## Auszug aus:

*Coding made easy: Space and Shape*

Das komplette Material finden Sie hier:

[School-Scout.de](https://www.school-scout.de)



© Copyright school-scout.de / e-learning-academy AG – Urheberrechtshinweis

Alle Inhalte dieser Material-Vorschau sind urheberrechtlich geschützt. Das Urheberrecht liegt, soweit nicht ausdrücklich anders gekennzeichnet, bei school-scout.de / e-learning-academy AG. Wer diese Vorschauseiten unerlaubt kopiert oder verbreitet, macht sich gem. §§ 106 ff UrhG strafbar.

## Table of contents

Foreword .....	4
Overview of the learning environments .....	5
Didactic background .....	6
Presentation of the programmable materials .....	11

## Learning environments

### Drawing plane shapes

Teaching Notes .....	14
Drawing plane shapes (Dash®) .....	19
Program template (Dash®) .....	21
Shapes flashcards (Dash® and Scratch) .....	22
Drawing plane shapes (mTiny) .....	23
Program template (mTiny) .....	25
Drawing plane shapes (Scratch) .....	26
Program template (Scratch) .....	28
Word bank: Drawing shapes .....	29

### Frieze patterns

Teaching notes .....	30
Module 1 (mTiny): Teaching notes .....	32
Continue frieze patterns .....	34
Describe frieze patterns .....	35
Draw frieze patterns with mTiny .....	36
Frieze patterns: Templates .....	37
Frieze patterns: Ideas workshop .....	38
Module 2 (Scratch): Teaching notes .....	39
Draw simple frieze patterns .....	41
Draw complex frieze patterns .....	42
Frieze patterns: Templates .....	43
Design and examine frieze patterns .....	44
Module 3 (Scratch): Teaching notes .....	45
Draw spiral patterns .....	47
Solution notes .....	48

### Coordinate system “Corolina Amusement Park”

Teaching notes .....	50
Where is what in the Corolina Amusement Park? .....	54
Nano’s paths through the amusement park .....	55
Code for Nano’s paths through the amusement park .....	56
Programming Nano’s paths through the amusement park .....	57
Map: Corolina Amusement Park .....	60
Nano’s vocabulary cards .....	61
Solution notes .....	62

### Networks and paths in Cornerstown

Teaching notes .....	64
Map of Cornerstown (Dash® and mTiny) .....	71
Eva’s way to school .....	72
Eva’s and Leo’s way to school .....	73
Program robot kids .....	74
Action cards (mTiny) .....	75
Blank action cards (mTiny) .....	76
Coordinate cards (mTiny) .....	77
Dash® in Cornerstown: Preparation .....	78
Dash® in Cornerstown: Let’s go! .....	79
Dash® in Cornerstown: Ideas workshop .....	80
mTiny in Cornerstown: Preparation .....	81
Map squares (mTiny) .....	82
mTiny in Cornerstown: Let’s go! .....	83
mTiny in Cornerstown: Ideas workshop .....	84
Ozobot® in Cornerstown: Preparation .....	85
Colour codes (Ozobot®) .....	86
Map of Cornerstown (Ozobot®) .....	87
Blank map of Cornerstown, DIN A4 (Ozobot®) ...	88
Blank map of Cornerstown, DIN A3 (Ozobot®) ...	89
Action cards (Ozobot®) .....	90
Ozobot® in Cornerstown: Let’s go! .....	91
Ozobot® in Cornerstown: School routes .....	92
Ozobot® in Cornerstown: Ideas workshop .....	93
Map of the Cornerstown Zoo (Ozobot®) .....	94

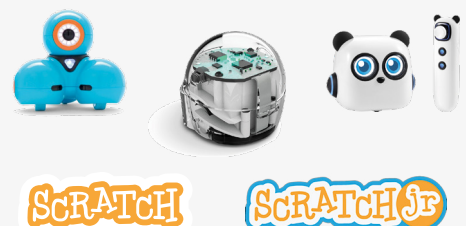
### Attachment

mTiny driver’s licence .....	95
------------------------------	----



### Download

- ScratchJr project “Corolina Amusement Park” (.sjr)
- Colour codes (Ozobot®) (.pdf)
- Solution video: Ida’s way to school (Ozobot®) (.mp4)
- Solution video: Noa’s way to school (Ozobot®) (.mp4)
- Action and coordinate cards (Dash®) (.pdf)
- Map of Cornerstown (Dash® and mTiny) (.pdf)
- Map of Cornerstown (Ozobot®) (.pdf)



## Foreword

What are the benefits of using programmable tools, such as robots, in primary school? While these tools have limited lifespans, the knowledge of interacting with them is crucial. Moreover, it is rewarding for students to witness a robot responding correctly and as desired. The process of achieving the desired outcome usually involves multiple trials and test runs, which helps prepare students for future professional fields.

Geometry and activities related to geometric ideas are particularly well-suited for incorporating programmable tools. Unlike in arithmetic, geometric algorithms can be expressed using everyday language. At the same time, students can be gradually introduced to appropriate mathematical terminology, enabling them to communicate their ideas more effectively and clearly.

First and foremost, Mathematical language is functional: it should support co-constructive work processes and learning processes. This includes three aspects:

*purpose, clarity, and targeting.*

These aspects apply not only to communicating ideas to others but also to programming, which is a specific way of interacting with machines. To achieve the desired behavior, machines need to be programmed, meaning they need to be given specific instructions in a precise manner.

Specifically:

- ◆ **Purpose.** It is important to consider the type of activity that is required from the machine and what can be expected from it. This is initially done in natural language. From the perspective of the machine and its designers, this can be referred to as pseudocode.
- ◆ **Clarity.** The pseudocode needs to be examined to ensure that the intended meaning is clear and possible misunderstandings are eliminated. Implicit assumptions should be reflected upon, taking into account the ways in which the machine under consideration works.
- ◆ **Targeting.** The clarified pseudocode must be translated into the machine's programming language. Syntactical peculiarities of the programming language need to be considered; otherwise, error messages and a refusal to act will occur. Machines are not generously accommodating dialogue partners.

While it may initially seem unfamiliar, those seeking to interact with machines and achieve specific behaviors must adapt to their unique dialects; otherwise, the desired results may not be attained.





During the process of programming a machine, two types of experiences are acquired:

1. Even though the peculiarities of programming a machine may initially seem off-putting and demotivating, persevering is worthwhile. Not only because the desired outcome is eventually achieved, but also because adapting to new machines becomes easier and faster with experience.
2. The transfer to forms of interpersonal dialogue is promising: there too, an assessment of what the dialogue partner already knows and understands is necessary and potential misunderstandings need to be identified and avoided. Furthermore, existing language can be adapted and modified in an agreed-upon way such as by introducing technical terminology.

Learning Geometry in primary school requires language development that starts from everyday language and elementary experiences and is increasingly characterized by agreed-upon specifications. This explains the central benefit of programming in primary school.

Bernd Wollring, Berlin 20 July 2022

## Overview of the learning environments

Learning environment	Focus	Grade	Programmable materials	Duration
Drawing plane shapes	Drawing quadrilaterals, describing construction steps, modifying programming based on given specifications, and checking properties of figures	Grade 2 to 4	mTiny, Dash <sup>®</sup> and Scratch 	2 to 4 hours
Frieze patterns	Frieze patterns, spiral patterns, and tessellations	Grade 1 to 4	mTiny, Scratch 	2 to 4 hours per module
Coordinate system “Carolina Amusement Park”	Algorithms, point coordinates in the plane, coordinate systems, orienting oneself in two-dimensional space, describing paths	Grade 4	ScratchJr 	2 to 4 hours
Networks and paths in Cornerstown	Orientating in two-dimensional space, describing paths, preformal familiarisation with the coordinate system	Grade 2 to 4	Dash <sup>®</sup> , mTiny, Ozobot <sup>®</sup> 	3 to 4 hours

Ozobot © 2021 Ozo EDU, Inc; Dash © makeunder.com; Logo Scratch © Scratch Team, CC BY-SA 2.0, available at: <https://commons.wikimedia.org/wiki/File:Scratchlogo.svg>;  
 Logo ScratchJr © Lrex, CC BY-SA 4.0, available at: <https://de.scratch-wiki.info/wiki/Datei:ScratchJrLogo.png>; mTiny © education.makeblock.com

## Didactic background

### *Algorithms in primary school mathematics lessons*

In discussions on digital education, it is not uncommon for the media and school councils to use terms such as “21<sup>st</sup>-century skills” or “computational thinking”. But what exactly do they mean in the context of the teaching of mathematics?

The German Standing Conference of the Ministers of Education and Cultural Affairs has issued a strategy paper on the goals of school education in the digital age (KMK, 2016). However, the competency requirements for modern education mentioned in the paper are formulated in a cross-curricular and topic-independent manner. Among the required competencies is the competent use of algorithms. As there are no dedicated computer science lessons in elementary school, teachers will have to find ways to integrate coding tasks into their teaching of other subjects. In particular, students should learn to:

- ◆ recognise and formulate algorithms, [...]
- ◆ recognise and formulate algorithmic structures in digital tools used, [and]
- ◆ plan and use a structured, algorithmic sequence to solve a problem (KMK, 2016, p. 18).

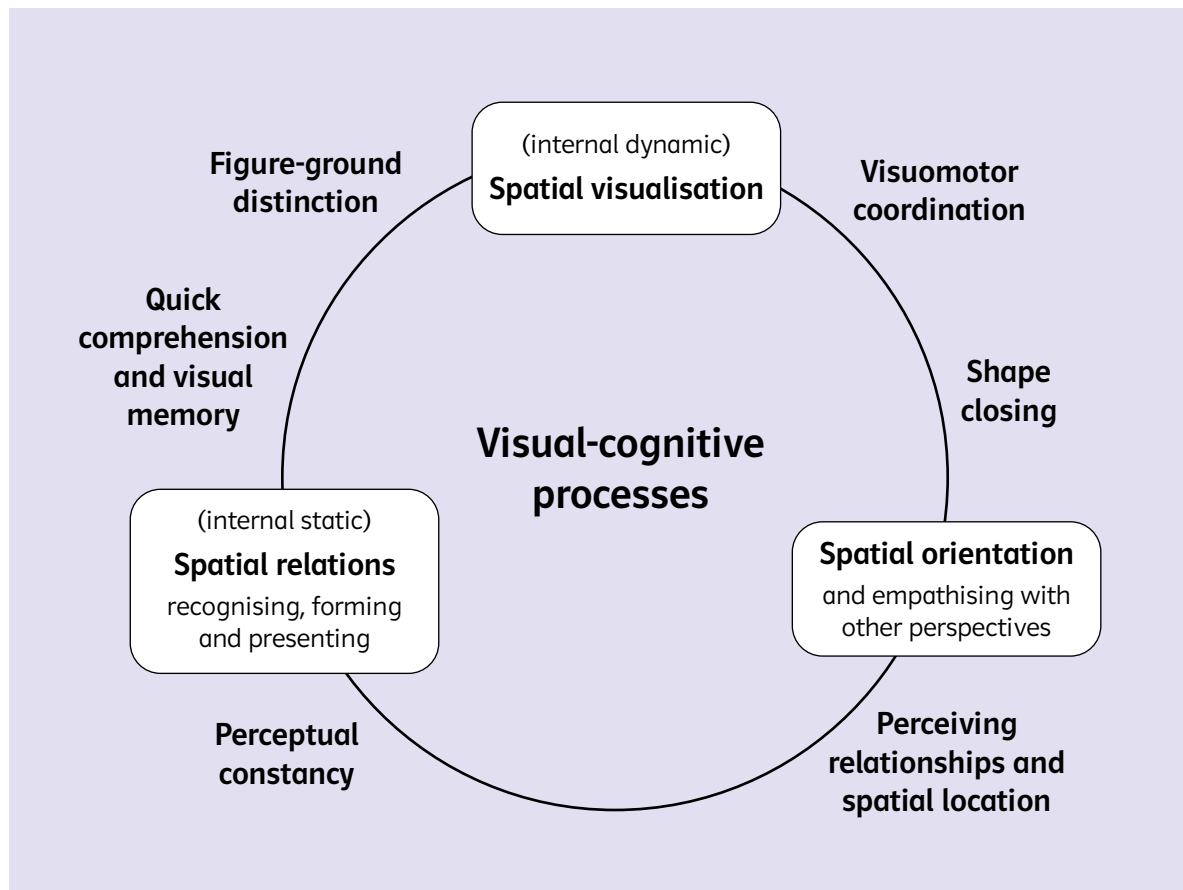
The concept of an algorithm is a fundamental idea in mathematics education, and as such mathematical algorithms are a suitable starting point for the implementation of the KMK guidelines. Winter (1976) describes fundamental ideas as mathematical concepts that have strong connections to reality, are characterised by internal conceptual richness, allow for different approaches, and can be revisited in different grades. The definition of an algorithm as a precise description of activity in the form of unambiguous instructions provides numerous points of contact with elementary school mathematics education, such as written methods for addition, the calculation of greatest common divisor and least common multiple, or descriptions of geometric constructions (Möller, Eilerts, Collignon & Beyer, 2022).

As a contemporary approach to promoting these skills, Wilson, Hainey & Connolly (2012) report on the promising use of programmable materials such as coding apps or robots. These help children gain insights into the concepts of algorithms and are motivating and action-oriented. Kyriakides & Meletiou-Mavrotheris (2018) find that programming activities help learners to deal better with abstract mathematical ideas and to formulate them more concretely. They also improve problem-solving skills, logical reasoning, understanding of arithmetic and algebraic processes, and geometric skills.

### *Promoting spatial perception and imagination*

With regard to geometric skills, the development of visual-spatial perception and processing is particularly important. Previous studies in this field indicate that inadequate promotion of spatial-visual skills can be partly responsible for children’s difficulties in dealing with arithmetic content and visualisation. Furthermore, this area of competence is considered a significant component of human intelligence and is closely related to academic performance - not only in mathematics education (Franke & Reinhold, 2016).

There are various models for describing visual-spatial perception and processing. The following is a three-component model based on Franke & Reinhold (2016). In practice, the three components cannot be clearly separated from each other because tasks always address all areas. However, they can serve as a basis for the diagnosis and promotion of visual-spatial perception and processing.



*Three-component model of spatial abilities according to Franke & Reinhold, 2016, p. 85.*

The first component is called “Spatial relations”. This comes into play especially in thinking processes that work with mental rotations and reflections of rigid configuration in different positions. The focus of the spatial imagination processes can be described here as internally static.

Another component in this model is “Spatial visualisation”. This comes into play especially in thinking processes with mental changes of parts of a figure or the figure as a whole. This means that something is spatially shifted, disassembled, folded, or enlarged.

The third component in the model is “Spatial orientation”. This area includes both the orientation in the perceived space and the mental transfer into another perspective. Consequently, it is also about grasping spatial relationships in relation to one’s own body and its spatial orientation, imagining other spatial positions, or making a right-left distinction.



## Designing and teaching with learning environments

The starting point of the working materials presented in this book is the concept of good tasks and their extension to learning environments. Good tasks enable learning the same subject at different levels of intensity and allow the consideration of different abilities, skills, solution ideas, strategies, and ways of presentation. Learning environments thus combine several of these (sub-)tasks under a mathematical or subject-related guiding idea into a flexible, comprehensive task (Wollring, 2009; Hirt & Wälti, 2016). The concept of learning environments

- ◆ represents central goals and contents of mathematics teaching,
- ◆ provides rich opportunities for mathematical activities for learners,
- ◆ can be adapted to the specific conditions of a class due to its flexibility,
- ◆ integrates mathematical and pedagogical aspects of teaching and learning (Krauthausen, 2018, p. 257f.).

Wollring (2009) formulated six guiding ideas for the design of learning environments. They describe the wholeness of learning environments based on the following aspects:

1. Subject and meaning
2. Articulation, communication, and social organisation
3. Differentiation
4. Logistics of managing learning materials and time
5. Evaluation and assessment
6. Connections with other learning environments

In this sense, learning environments are a framework for teachers that can be adapted and specified to achieve specific learning goals in their classrooms. The guiding ideas can be applied to analogue and digitally supported learning environments.

In addition to these basic characteristics for the overall design of learning environments, four experiences for promoting *computational thinking*, according to Kotsopoulos et al. (2017), were taken into account during the design of subtasks within each learning environment. The tasks were designed in such a way, that they enable an increasingly deeper engagement with the learning material.

Pedagogical experiences from Kotsopoulos et al. (2017), ascending according to complexity:

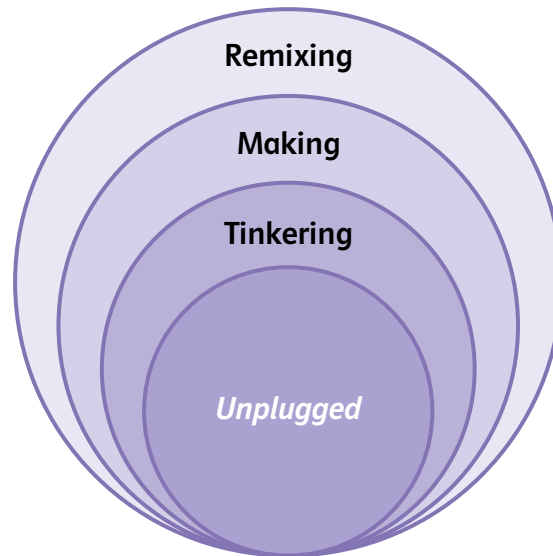
- ◆ The first stage is called “Unplugged”. Here, experience is gained by engaging with materials without digital support.
- ◆ At the “Tinkering” level, existing objects are modified by tinkering and fiddling around under the question “What if ...?”.
- ◆ This is followed by experiences in the third stage, “Making”. Here the focus is on activities that create new objects.
- ◆ The most complex experiences are made possible by the “Remixing” level. Here, already existing objects are modified (in parts or as a whole) and used in other projects or for other purposes

In short

### Learning environments

are large, flexible tasks, which usually consist of several subtasks and are bound together by certain inner-mathematical or relevant central ideas.

(Hirt & Wälti, 2016, p. 13)



*Four pedagogical experiences to promote computational thinking, according to Kotsopoulos et al, 2017, p. 159.*

### **The scaffolding of programming**

Depending on the children's prior knowledge, the level of difficulty can be varied with the help of an appropriate scaffolding structure (detailed in Möller, Eilerts, Collignon & Beyer, 2022) for instance by choosing a suitable programming template to guide student activities.

- ◆ To get started, a faulty program can provide a suitable template for independent correction.
- ◆ For advanced students, the pre-structured template can be used, in which the command category is given by the colour of the blocks, but the concrete action still must to be identified.
- ◆ For the very experienced, a template can be dispensed such that a working program that achieves the given goal must be coded by students entirely on their own.

**In short**

**Scaffolding** describes a way in which a child can be supported to progress during a task or activity. The basic idea is that any support (for instance in the form of hints or further material) is just enough to enable the learner to cope with the task at hand and thus allows the student to reach the next level of competence on their own. (Tavassolie & Winsler, 2018).

In addition, depending on previous knowledge, you can choose between linear or loop-based programming for the templates. For the linear templates, the optimisation and reduction of the programming blocks through the use of loops can follow as a deepening task.



# SCHOOL-SCOUT.DE



Unterrichtsmaterialien in digitaler und in gedruckter Form

## Auszug aus:

*Coding made easy: Space and Shape*

Das komplette Material finden Sie hier:

[School-Scout.de](https://www.school-scout.de)



© Copyright school-scout.de / e-learning-academy AG – Urheberrechtshinweis

Alle Inhalte dieser Material-Vorschau sind urheberrechtlich geschützt. Das Urheberrecht liegt, soweit nicht ausdrücklich anders gekennzeichnet, bei school-scout.de / e-learning-academy AG. Wer diese Vorschauseiten unerlaubt kopiert oder verbreitet, macht sich gem. §§ 106 ff UrhG strafbar.