Introducing Programming Concepts Through Montessori Materials

Steffi Rudel
University of the Bundeswehr Munich, Germany

Marlene Engels
University of the Bundeswehr Munich, Germany

Abstract

Programming is a skill already taught in many schools today. However, for historical reasons, it has not yet been included in the Montessori method. To close this gap, we developed learning materials for Montessori schools and present them in this article.

We show the design of a new learning material for introducing programming concepts in Montessori elementary level (6–12 years). The material is a concrete physical learning material and is supported by a modular lesson structure, which also takes the principles of Computational Thinking in account. Finally we discuss its suitability for Montessori education based on expert feedback. The article shows interesting examples of how programming can be taught in Montessori elementary level.

2012 ACM Subject Classification Applied computing → Education

Keywords and phrases Montessori Method, Programming, Montessori Material, Computational Thinking

Digital Object Identifier 10.4230/OASIcs.ICPEC.2026.5

Funding This work originates in the LIONS research project. LIONS is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr which we gratefully acknowledge. dtec.bw is funded by the European Union – NextGenerationEU.

Acknowledgements We want to thank the Montessori experts Christiane Salvenmoser and Saskia Haspel for evaluating the learning material as well as the Montessori teacher Christiane Daidrich for supporting the research.

1 Introduction

Computers and digital devices are ubiquitous today. Yet without the software that controls their operation, they are useless. Therefore, the ability to program – as part of computer science (CS) – is a vital skill that goes beyond mere knowledge of programming languages, libraries and frameworks. This ability could and should be taught from an early age to promote digital sovereignty of the individual.

Schools play an important role in teaching programming skills to children and young people. In addition to the standard state school system, reform pedagogical approaches, such as the teachings of Montessori [11] have been around for decades.

Maria Montessori's approach to learning in their time differed dramatically from the widespread authoritarian style. Instead, she believed that knowledge should not be "drilled" into children, but that motivation to learn is deeply rooted in every person (and therefore in every child). McNamara writes about this: "Maria Montessori believed that a goal of education should not be to fill children with facts, but rather to cultivate their own natural desires to learn." ([9] p. 95).

In the Montessori Method, the prevailing view is that learning is very much dependent on the *learning environment* and what is offered there. Therefore, learners acquire the knowledge themselves – the teacher is primarily responsible for preparing the environment and providing appropriate opportunities (so-called *Montessori material* the students work with independently).

However, the problem with CS is that computers had not yet been invented when Maria Montessori developed her approach and materials. Therefore, the ability to program or comprehend digital technology was not considered in the curriculum of Montessori Schools, nor are there dedicated Montessori materials for this topic. The proposed research aims to close this gap by creating suitable Montessori material.

Based on the teachings of Maria Montessori, the world-leading Association Montessori Internationale (AMI, `https://montessori-ami.org/`) divides the learning phases in Montessori education into the following stages, while our research focuses on elementary-aged children:

- Assistants to Infancy: Ages 0–3
- Primary: Ages 3–6 (Children's House / Casa dei Bambini)
- Elementary: Ages 6–12
- Adolescent: Ages 12–18

In summary, we address the following research question: **Which new Montessori material could support students aged 6–12 years in learning programming-logic?**

The contribution of the paper is as follows: First, we will provide an overview of the current state of the art (Section 2). Then we present a concrete physical learning material and a modular lesson structure (Section 3). After that, we discuss its suitability for Montessori education based on expert feedback (Section 4). The article concludes with a summary and an outlook (Section 5).

2 Related Work

In order to gain access to the subject area, a systematic literature search was carried out. We initially used the two databases EBSCO and ACM Digital Library one after the other and then conducted a snowball search.

The journal article by **Stefan Scippo and Fabio Ardolino** presents a long-term study in which children in grades 1–5 of primary school were followed in an Italian Montessori school [15]. Based on Montessori principles, technological materials were designed to promote computational thinking and encourage multiple solution paths, with built-in error control but flexible procedures. Over the years, students progressed from coloring binary patterns to working with the Ozobot for grammar tasks and storytelling, and later with LEGO WeDo for increasingly complex programming activities. In the final year, they created a website using WordPress. In summary, the article shows interesting examples of how programming can be taught at the Montessori primary level.

The study by **Mollie Elkin, Amanda Sullivan and Marina Bers** examines a robotics curriculum for Montessori classes (grades 1–3), aiming to integrate foundational programming and engineering concepts into early education [3]. Using a case study with a Montessori teacher and 19 students, the research combines qualitative and quantitative methods to analyze both classroom implementation and the teacher's development in robotics knowledge. Based on Montessori principles, a curriculum using LEGO WeDo was developed around the theme of Ancient Greece and structured into six learning units. The material is described

as self-corrective, supporting independent learning and engaging multiple senses, aligning closely with Montessori approaches, although a detailed comparison with Montessori material criteria is not provided.

Montessori teacher **Sheri Milton** uses a practical example to show how mathematical skills and programming can be taught in a Montessori school (children aged 9–12 years) [10]. A project with the Turtle Math software is described. The program works with the LOGO programming language and figures can be drawn using a turtle. The article shows that programming skills in a Montessori context should never be taught in isolation but always in context and in conjunction with other skills.

John McNamara explains how STEM (Science, Technology, Engineering and Mathematics) subjects can be naturally integrated into Montessori education, based on a conference presentation [9]. He emphasizes that learning arises from understanding relationships rather than isolated details. Technology is presented as a supportive tool, not the focus, helping students explore questions and solve problems more efficiently. McNamara also highlights the importance of a prepared environment that encourages inquiry and experimentation, as well as giving students enough time to develop their own ideas rather than rushing tasks.

Luisa Greifenstein and Adrian Hanusch provide in their poster abstract initial insights into a systematic literature search on the keywords Montessori and CS education [5]. In the article, various sources are examined for the connection between Montessori education and CS. Some of the sources in the poster abstract discuss the teaching of programming skills in the Montessori context and were therefore relevant: one of the sources mentioned was already found in the previous search [3], a second was included in the search using the snowball method [1].

The article by **Oren Zuckerman, Saeed Arida, and Mitchel Resnick** presents "Froebel-inspired Manipulatives" (FiMs) and "Montessori-inspired Manipulatives" (MiMs), focusing on physical and digital tools for learning [18]. While FiMs emphasize real-world object design, MiMs aim to model abstract structures, particularly dynamic systems. The authors introduce digital MiMs – enhanced physical objects based on Digital Manipulatives [6] – used to explore concepts like change and probability. A study with children aged 4–11 informs the development of design guidelines for these materials.

Reinhard Fischer deals with the use of computers, software, and the Internet (at that time not nearly as widespread as today) as learning and working materials in Montessori education [4]. These are summarized as "new media". Fischer does not deal with programming in the article, but questions 3 and 4 of the text are interesting: "3 Montessori education is based on free work and has a tried and tested set of learning materials for this purpose, which must fulfill certain criteria. How do the new media behave when they are evaluated on the basis of the criteria of the Montessori materials? 4. What criteria must be used to modify new media so that they can be used sensibly in the context of free work?" (p. 196–197)

The practical report by Montessori teacher **Markus Wurster** is dedicated to "Algorithms and Computer Science" and deals with the concept of algorithms and their teaching, Scratch, LOGO and microcontrollers, but no explicit reference is made to the Montessori Method [17].

Mario Valle is one of the few authors to address the possible integration of new media into Montessori education [16]. The book is explicitly aimed at primary school teachers who teach at a Montessori school and questions whether and in what way new technologies (such as computers and software, tablets and apps, 3D printers, the Internet, robotics, or virtual and augmented reality) can be used sensibly in the Montessori school. He suggests that technology itself (when teaching skills in the field of new technologies) should be completely avoided until the age of around 8, as children up to this age must first acquire the capacity

for abstraction. On pages 76–78 he lists 14 “indispensable characteristics” (based on the work of Montessori) that a material must have as a Montessori material. He then applies these to a LEGO Mindstorms robot as an example. Valle also devotes a separate section to learn programming, as well as robotics (p. 132–135). For programming, he suggests various tools such as Kodu, Scratch or other block-based programming languages. He describes robotics as an artifact that contains the steps of design, assembly and programming and is therefore very suitable for the Montessori Method. He mentions the tools Bee-Bot, LEGO Mindstorms and LEGO WeDo as well as the KIBO robot.

In a practice-oriented article, **Steffi Rudel** describes how programming skills can be taught to pupils in a (voluntary) afternoon program at a Montessori school in grades 3–8 [13]. Several tools and platforms are mentioned, and selected examples of their use are presented. In grades 3 and 4, for example, analog programming is recommended as an introduction; later, digital learning units from the code.org platform and the Scratch learning environment can be used. The Minecraft Education Edition learning environment and Calliope Mini are recommended for grades 5 and 6. Although the article refers to quality features that learning materials in a Montessori school must fulfill, there is no dedicated consideration of whether the tools presented fulfill these.

The authors **Felienne Aivaloglou and Efthimia Hermans** deal with Early Programming Education in their research [1]. They investigate the factors that influence the performance of elementary school students and the effects of Early Programming Education on later career orientation. For this purpose, an 8-hour Scratch unit was conducted in two different elementary schools in the Netherlands, one of them was a bilingual Montessori school. However, no differences between the schools were considered either in the course design or in the evaluation.

3 Creation of a new Montessori Material

3.1 Preliminary Considerations and Design Principles

Objective and Approach

In order to close the gap described in the introduction, a Montessori material for elementary grades at Montessori schools was created. The material is designed for young school children, so it can be assumed that these children are still at the early stages of developing their programming skills. The material is therefore designed to teach the underlying concepts rather than a specific programming language.

The Design Science Research (DSR) approach was used for the development [12]. The central element of the DSR is an artifact that represents the result of the research and development process. The artifact of this work is a Montessori material that playfully introduces students of the Montessori elementary level (grade 1–6) to basic concepts of programming-logic.

The initial concept for the learning material was developed based on the literature review and a desk research of existing commercial programming games. We tried to implement the design principles of a Montessori material (see below) from the very beginning, but it was already clear that these characteristics would need to be refined during the design cycles based on expert feedback.

Requirements for a Montessori Material

In order to understand why specific research for Montessori in the area of programming is necessary and why the teaching methods and materials of the standard school system cannot be transferred without closer examination, the following background is important:

Firstly, the Montessori Method puts the child at the center, not the teacher or what he or she wants to teach. The approach is always "child-centered", and the teacher does very little to "teach" the children, instead guiding them through observation and feedback. In Montessori education, children acquire knowledge and skills independently using the Montessori material, which was developed by Maria Montessori and is nowadays used in all Montessori schools worldwide. This *Montessori material* is used by the children in a so-called prepared environment, teachers using the Montessori Method are learning guides rather than instructors.

Design Principles of Montessori Material

There are some "essential characteristics" that Montessori learning materials must fulfill. They were first described by Maria Montessori herself ([11], p. 177–185), although strictly speaking, this description applies specifically to sensory materials (materials for the first development stage, 0–6 years). However, these characteristics are also reflected in the Montessori materials designed for older children and are frequently cited in the secondary literature (for example [7] or [14]).

- **Isolation of difficulty**: A material should always contain only one new difficulty so the child can focus their attention specifically on it.
- **Limitation / Order**: Materials are structured, complete, and present only once in the room to promote order, care, and social consideration.
- **Activity**: The material encourages action and enables independent learning.
- **Inviting nature / Aesthetics**: The material should be designed to be clear, orderly, beautiful, and child-friendly so it arouses interest and concentration.
- **Self-control**: The child should be able to recognize and correct mistakes on their own as much as possible without constant supervision by adults.

However, it is important to distinguish which developmental stage the material is used in. For example, in the first developmental stage (ages 0–6), self-regulation should be embedded within the material itself or within the process; in the second developmental stage (ages 6–12), self-regulation is primarily achieved through group discussion. Another example is the aesthetic of the material – in the first developmental period, the focus is indeed on "beautiful" material, whereas in the second developmental period, the tool-like nature of the material is much more central.

3.2 The Artifact

The Elements

The material consists of a playing field, a playing piece in the shape of a dragon, physical command blocks, task cards, and optional game accessories such as walls, stars, and mountains.

The material was deliberately designed as a physical playing field to make abstract programming concepts tangible through hands-on activity. The playing field provides clear orientation through coordinates while offering ample opportunities for tasks of varying

difficulty. The game piece was introduced so that movements and program execution can be visually tracked. The command blocks were designed to interlock, making the structure of a program visible and tangible. At the same time, this supports the step-by-step planning and assembly of sequences.

In the center is the three-dimensional playing field with 8×8 squares (Figure 1). Each square is uniquely named by a combination of letters (A–H) and numbers (1–8). In addition, walls can move around the playing field. These can be placed in the spaces between two adjacent squares and block the piece's path (Figure 1).

Figure 1 Game board with coordinates and plug-in walls as a modifiable obstacle system.

A game piece, in this case a dragon, is guided via coordinates from a starting position to a target position (Figure 2). The piece always starts on square A1 in the upper left corner of the playing field, facing square A2. There are also stars that can be collected and mountains that can be moved by the piece.

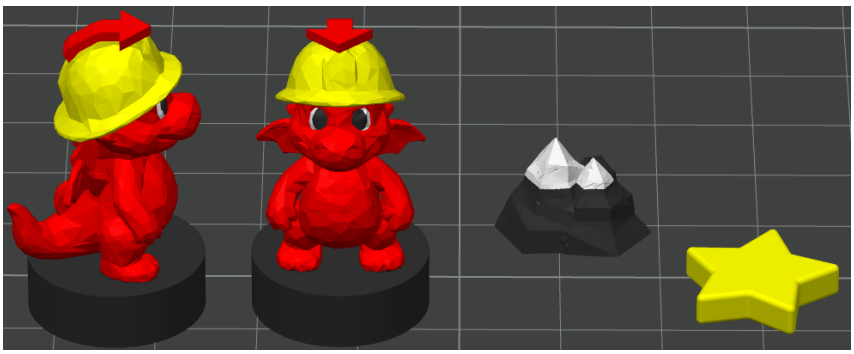

Figure 2 Game piece "dragon", stars, and mountains.

The walls, stars and mountains serve to make the tasks more varied and stimulate different thought processes. The dragon is controlled indirectly via a sequence of physical command blocks, which must first be assembled into a complete program (figure 3). The blocks are

color-coded and designed in a way to be connected to each other. Each program begins with a red start block (“Start”) and ends with a red stop block (“Ende”). The actual sequence of commands is located between these two blocks. The yellow blocks represent simple movement commands such as *walk* (“gehe”) or *turn left/right* (“drehe links”, “drehe rechts”), which must be executed immediately by the game character.

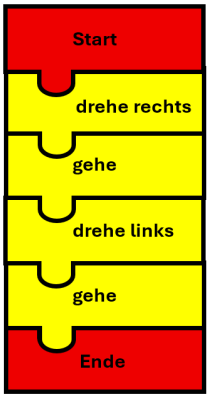

Figure 3 Example of a program sequence consisting of command blocks.

White building blocks, on the other hand, represent control structures such as if ...then ... conditions (Figure 4, “wenn...dann...”). This allows more complex programs to be formulated.

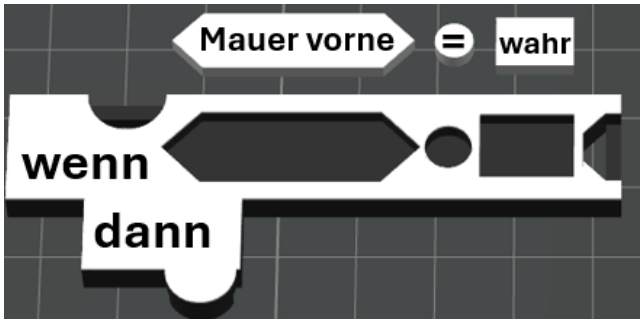

Figure 4 Building blocks for the if ... then ... conditions.

Simple conditions such as *wall in front* (“Mauer vorne”) are inserted into the hexagon. To formulate more complex conditions, the white building block can be extended with a plus sign or an O sign to add another simple condition.

The Design

The material has various properties to make it easy to handle. The prototype was produced using a 3D printer. The colors were deliberately designed to be distinct – the basic programming blocks are yellow with black (go, turn left, turn right), whereas the programming blocks for the conditions are white. This color coding allows learners to easily navigate the programming. The programming blocks are stored thematically in separate boxes, which are secured with a magnetic closure and reflect the colors of the blocks. The game board is labeled throughout with numbers and letters to clearly identify each square. This is important for the tasks, so it is clear which square the piece starts on and where it should move to. Both the game board and the game piece are fitted with magnets, ensuring that the piece stands securely on the individual squares and cannot simply slip out of place.

The optional walls for the game board come in two different colors and are each stored in their own box, which has been designed in the corresponding color and size. To make it easier to position the walls correctly, two templates are provided, which are also designed in the corresponding colors. Each template has a tab so that it can easily be lifted off the playing area without removing the walls. Furthermore, the abbreviation A1 is marked in the top left corner of both templates, so the students know which way round the template must be placed on the playing area.

The dragon figure wears a helmet featuring an arrow. This makes it easy to see which direction the figure is facing – this is important for programming, as "go" always means in the direction the figure is currently looking.

Modular Structure of the Lessons

The Montessori material is divided into five modules that build on each other. Each module introduces a central idea of programming and deepens it through appropriate tasks. The order of the modules creates a didactically meaningful learning path that supports the transition from concrete action to abstract thinking. The framework developed by Brennan and Resnick was taken into account in the design and structure of the modules [2]. The authors argue that Computational Thinking (CT) consists not only of programming concepts, but of three equal dimensions:

The first dimension is formed by **CT Concepts**, which include basic programming concepts such as sequences, conditions, events, operators, data (variables, lists), and loops. These elements form the conceptual vocabulary that learners must understand and apply when programming.

The second dimension is formed by **CT Thinking Practices**. This dimension describes ways of working and thinking when dealing with the program and possible problems that may arise. These include, in particular, iteration (design – testing –improvement), debugging, and decomposition. The focus is not on *what* is learned, but on *how* learners approach problems.

The last dimension is called **CT Perspectives**. This dimension refers to attitudes and viewpoints that learners develop when dealing with technology. These include the expressive perspective (coding as a means of expression), the collaborative perspective (learning through exchange, feedback, and collaboration), and the reflexive perspective (recognizing patterns, structures, and systems in one's own environment).

To support self-directed learning, task cards have been added to the learning materials. The task cards explain the respective task that needs to be solved next. Each card first contains the task with possible additional conditions, followed by a visual representation of the playing field. On this, the figure is shown on the starting square A1 looking to the right. The goal is marked by a black pin symbol; walls, stars, and mountains are drawn in according to the task. On the back of the task card, there are one or more possible solution sketches that can be used for self-checking. However, it is not always possible to show all theoretically correct solutions. The solution is therefore only an aid and not an absolute correction aid. These task cards are organized into six modules. Each of the modules takes up aspects of the three dimensions and combines them into a coherent learning process.

Module 1: Basic movement commands (imperative programming): This module introduces students to the basics of imperative programming. The focus is on the principle of sequencing: commands are executed in a fixed order with the game character.

Module 2: Interaction with objects (state changes): In the second module, the command sequences already familiar from Module 1 are expanded to include the concept of state changes.

Module 3: Truth values: In the third module, the focus is on understanding truth values. Students learn that statements in programming logic can be either true or false.

Module 4: If–then conditions (decision logic): In the fourth module if–then structures are introduced. Children learn that programs do not only run linearly, but can branch depending on conditions.

Module 5: Testing and debugging: The fifth module is entirely devoted to testing and debugging. The focus here is no longer on learning additional commands, but on consciously reflecting on errors and systematically correcting them.

Sequence of a Learning Unit

A typical unit begins with reading the task card. The students build the walls on the field. They can either build the walls freehand, use the template to build them, or check if their construction is correct using the template. The dragon is then positioned on the starting square A1, facing A2. Then, either alone or in teams, they consider how the task could be solved. The students basically have two approaches at their disposal.

1) The program (solution to the task) is planned in a purely abstract manner. The command blocks are put together according to this plan. Only then is the command chain traced by actively moving the dragon.
2) The program is developed with the help of the dragon and the game board, and the command blocks are strung together in the process. If the execution of the program does not lead to the desired result, the faulty piece of code is specifically searched for, the command sequence is adjusted and tested again. A red arrow is included with the game to help with the testing process. This arrow may only be moved down one command block at a time. This prevents commands from being executed twice or omitted during the test phase (Figure 5).

The cyclical approach (plan–test–correct–test–...) is typical in the field of software engineering and is taught to children in a playful manner.

3.3 The Design Cycles

As part of the DSR, several improvement cycles have already been completed during the development of the artifact. Through an ongoing comparison with existing educational games in the field of programming, as well as discussions with experts, the following adjustments were made:

Lowering the Difficulty Level and Reducing the Number of Modules

The game was originally planned with the following modular structure:

- Module 1: Simple commands
- Module 2: Conditions
- Module 3: Loops

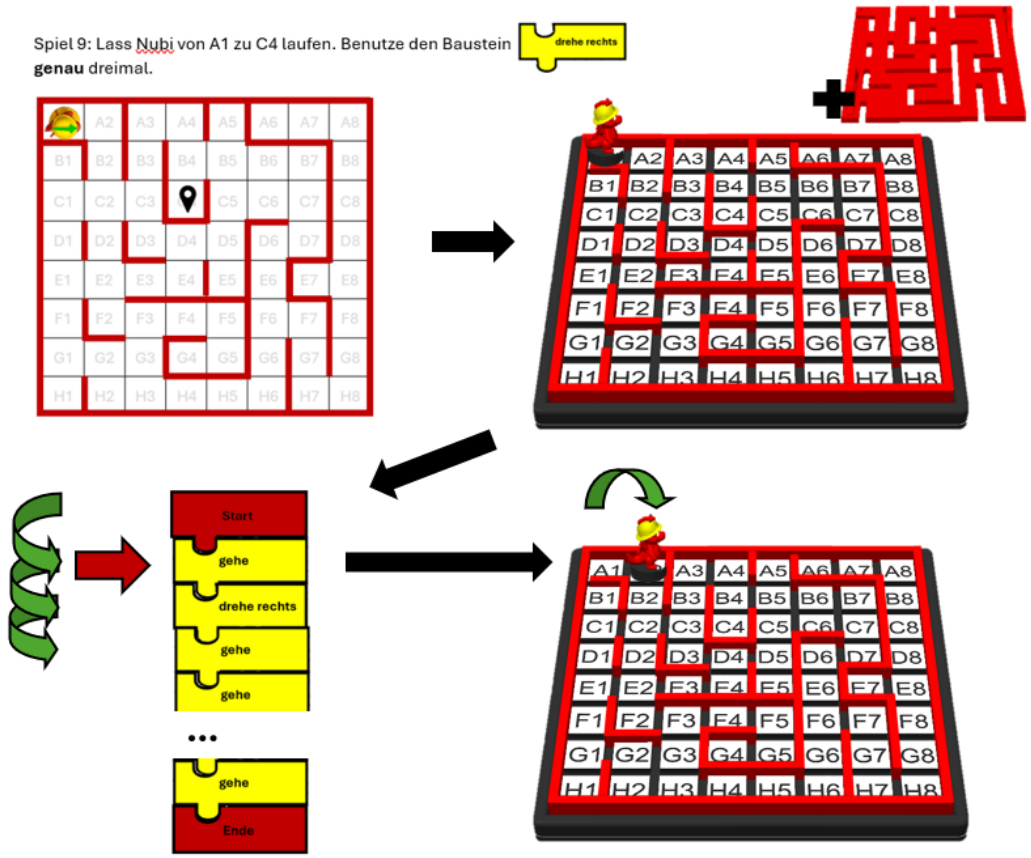

Figure 5 Example sequence of a learning unit.

- Module 4: Attributes and values
- Module 5: Developing custom functions
- Module 6: Basic concept of iteration
- Module 7: Debugging and testing

Analysis of this structure revealed that the transition between the modules would have been too rapid, thereby significantly increasing the cognitive load on the children. Young school children are in the concrete operational stage [8], in which abstract and nested structures can only be processed to a limited extent. In particular, loops, functions, and debugging represent abstract meta-levels of programming, the understanding of which requires a solid foundation in simpler concepts. Proceeding too quickly would have overwhelmed the children and hindered their learning success.

Simplification of Language

Another point of criticism concerned the labeling of the command blocks. These were heavily based on traditional programming standards and included elements such as parentheses and semicolons. While such syntactic elements are necessary for professional programming, they are not required for understanding basic algorithmic concepts. For children, however, they present an additional extrinsic difficulty that distracts from the actual learning objective: recognizing logical relationships and sequences. Therefore, these syntax elements in the command blocks were simplified: For example, **gehen();** became **gehe** ("gehen" means go).

Optimizing the Walls

Originally, the walls consisted of one-, two-, and three-piece combinations, which made it difficult to place them using the reference images. In addition, small gaps on the game board caused the walls to shift slightly. This problem was solved by reducing the number of wall combinations that appear in all games to two variants (red and white). Sturdy wall pieces were developed using a 3D printer to match these two combinations. As an additional aid, stencils were included with the game. These can either be used to check the walls that have already been placed or laid out on the playing field at the start so the walls can be placed directly using the stencil. The templates are color-coordinated with the respective wall combination.

4 Reflection and Suitability for Montessori Education

The material was developed to introduce Montessori students of the elementary phase to programming concepts. Testing with children is currently pending, as approval has not yet been granted by the research university's data protection and ethics committee. Therefore, adult experts were consulted to assess the suitability of the material for Montessori education. The Montessori experts are Christiane Salvenmoser and Saskia Haspel. They are the founders and directors of the Montessori Academy in Austria and have been working for decades as Montessori instructors, consultants, coaches, and evaluators. They served on the board of the Austrian Montessori Society for more than 25 years and train Montessori educators through certification courses. The workshop was supported by the german Montessori teacher Christiane Daidrich.

The materials were evaluated during a workshop held on April 30th, 2026 in Vienna. The material was first demonstrated and explained, after which the experts tested it on their own. This approach mirrors the way Montessori materials are used in a classroom: The teacher provides an introduction, and then the children work independently with the material. The experts' comments were recorded in writing and evaluated for this article.

The material was evaluated both from a general, educational perspective and specifically in relation to the Montessori method. Several aspects were cited as **strengths in a general sense**: The materials are designed to be tactile and easy to handle. The color-coded separation of the programming blocks into individual storage boxes supports a modular learning experience. In the experts' view, the material clearly prepares children for programming at various difficulty levels, and the cards allow children to work at their own pace. As **biggest weakness** was mentioned that navigating the game piece's paths requires children to have good spatial orientation. A solution should be found for this, as spatial orientation is not a prerequisite for effective programming.

However, in order to actually use the learning material as **Montessori material**, they suggested a few more aspects.

In Montessori education for school-age children (starting in Elementary), learning a subject area can be divided into **three learning phases**. In the first phase, the *introductory phase*, the material is shown to the learners and the rules and usage are explained. The learners then familiarize themselves with the material. The second phase, *exploration*, involves investigating the material and thereby acquiring knowledge independently. In the third phase, *integration*, the newly acquired knowledge is finally linked to existing knowledge and, if appropriate, presented in a suitable manner (for example, as a presentation, a poster, or a product). A key principle of Montessori education is that the duration and intensity of each phase are determined solely by the child, not by the teacher. Some children may grasp

a concept after just a few exercises, while others may take much longer and repeat things over and over. What matters most is that there is sufficient freedom in the exercises and that no rigid guidelines hinder the child's learning. However, the current design of the task cards only partially allows self-directed learning and exploration of topics. Since all lessons within a module build upon one another, individual task cards cannot simply be skipped if a child has grasped the topic more quickly. On the other hand, repeating a task card for reinforcement is rather unappealing, as this stifles the child's spirit of inquiry and does not encourage joyful learning.

Therefore, it was suggested to replace the task cards by **activity cards**. These could be organized into subcategories according to the modules, allowing the students to decide how many of the cards they would like to work on for a given topic. **Self-checking**, which is a key pillar of Montessori materials, should not be resolved by printing the solution on the back. The question is: What do we want to achieve with the learning child? Is it about the solution being exactly correct? Or isn't it more about the process of evaluation and debugging? The Montessori experts therefore suggested that program verification should instead be carried out through teamwork (one child programs, the other verifies). A fixed division of roles would be helpful here, so each child is aware of its task. Debugging, if something is wrong, can then take place together through dialog.

Here is an example to illustrate: Child 1 draws an activity card that says: *Place the game piece on square A1; he is facing A2. Now create a program that makes the dragon move to square C4.* Child 1 reads the card silently and creates the program using the programming blocks. Child 2 (who does not know the destination) then executes the exact program using the game piece. Now, the two children look together to see which square the game piece ends up on and compare it with the activity card. If the square is not the same, they work together to figure out how the program needs to be changed.

Also, the modules could be effectively expanded by allowing students to create and solve their **own tasks** through teamwork. It would also be quite conceivable to add **challenges**, such as:

- Can you solve the task using a different program?
- Can you solve the task using fewer programming blocks?
- Can you solve the task using the longest possible program?
- How many ways can you find to reach the goal?

In addition, it was suggested to create so-called **Definition materials** for each module. In Montessori education, these materials are provided to learners for specific subject areas and encourage them to look up information on their own. This reduces the workload for the Montessori teacher and supports the learners' independence and self-efficacy ("Help me to do it myself"). The definition material consists, among other things, of a small ring binder in 14x14 cm format. On a double-page spread, the left side features a relevant illustration, while the right side contains the explanation along with the technical terms.

It was also mentioned that the material needs a catchy **name** (all Montessori materials have a name that is easy for children to remember and that clearly identifies the material). Last but not least the teachers should be provided with a clear instruction for the **presentation** of the material.

5 Conclusion and Outlook

As the literature review shows, research to date has only dealt rudimentarily with the topic of programming in Montessori schools. Montessori materials have also hardly been systematically created yet. To close this gap, a new learning material specially for the

Montessori method was developed. The DSR was applied, and the structure of the lessons was chosen to be modular. Finally, since approval for a study involving students had not yet been granted, the material was evaluated by adult Montessori educators.

The Montessori experts highlighted many positive aspects of the learning materials and explicitly emphasized the relevance of the topic for Montessori pedagogy. In their view, incorporating this topic into Montessori pedagogy is highly desirable. From the experts' perspective, Montessori materials represent the right approach, but they must absolutely be embedded within other methods of Montessori pedagogy. Examples include the Definition materials or Cosmic Education, which places all of the learners' experiences into context.

As the research progresses, we will first conduct a more in-depth evaluation of the learning material. To this end, we will carry out empirical tests with students at Montessori schools and incorporate the findings into subsequent design cycles for the material. In doing so, we will also take into account other Montessori components, such as *Cosmic Education* and the *Definition materials*.

Montessori pedagogy represents a promising opportunity for learners to prepare for the digital future. By expanding the materials to include the subfield of programming, Montessori pedagogy is enriched with contemporary aspects, thereby silencing critics who argue that Montessori is no longer up to date. In the further course of the research, the goal is not only to improve the programming materials through additional design cycles. It is also intended to examine whether other CS aspects, such as Artificial Intelligence, should be integrated into the curriculum.

However, the materials developed are not only valuable for Montessori education. Even today, elements of the Montessori Method are increasingly being adopted in the standard state school system. One example is independent and self-regulated learning: Modern societies increasingly demand skills such as problem-solving, initiative, and self-direction. These are fostered by the self-directed learning that is central to Montessori ("Help me to do it myself"). The learning material is also suitable for traditional schools to help children experience and learn the basic principles of programming in a self-directed way. The learning materials can therefore support teachers shifting their role from "instructor" to "learning guide". Especially in an increasingly fast-paced, digital world, the goal is no longer simply imparting knowledge to learners. Rather, it is to equip them with the skills to acquire relevant knowledge independently.

References

1 Efthimia Aivaloglou and Felienne Hermans. Early Programming Education and Career Orientation: The Effects of Gender, Self-Efficacy, Motivation and Stereotypes. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 679–685, Minneapolis MN USA, 2019. ACM. doi:10.1145/3287324.3287358.

2 Karen Brennan and Resnick Resnick, Mitchel. New frameworks for studying and assessing the development of computational thinking. In *Annual American Educational Research Association meeting AERA*, 2012.

3 Mollie Elkin, Amanda Sullivan, and Marina Bers. Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education: Innovations in Practice*, 13:153–169, 2014.

4 Reinhard Fischer. Neue Medien und Montessori-Pädagogik. In Harald Ludwig, Christian Fischer, Reinhard Fischer, and Montessori-Vereinigung, editors, *Verstehendes Lernen in der Montessori-Pädagogik*, volume 8 of *Impulse der Reformpädagogik*, pages 195–222. LIT, Münster, 2003. Hrsg. Buch: Ludwig, Harald; Fischer, Christian; Fischer, Reinhard.

5 Luisa Greifenstein and Adrian Hanusch. Combining Montessori Pedagogy and Computing Education: First Insights from a Systematic Literature Review. In *Proceedings of the 19th WiPSCE Conference on Primary and Secondary Computing Education Research*, page o.P., Munich Germany, 2024. ACM. doi:10.1145/3677619.3678102.

6 J. Patten, James Patten, H. Ishii, Jim Hines, Hiroshi Ishii, Gian Pangarp, J. Hines, and G. Pangarp. Sensetable: A Wireless Object Tracking Platform for Tangible User Interfaces. In *Proceedings of SIGCHI 2001*, pages 253–260, Seattle Washington USA, 2001. Association for Computing Machinery ACM. doi:10.1145/365024.365112.

7 Michael Klein-Landeck and Tanja Pütz. *Montessori-Pädagogik: Einführung in Theorie und Praxis*. Montessori Praxis. Herder, Freiburg, 4. auflage edition, 2019.

8 Saul McLeod. Piaget's Theory and Stages of Cognitive Development, April 2025. doi:10.5281/ZENODO.15241970.

9 John McNamara. How the Montessori Upper Elementary and Adolescent Environment Naturally Integrates Science, Mathematics, Technology, and the Environment. *NAMTA Journal*, 2(41):82–97, 2016.

10 Sheri Milton. Star Power - Connecting Montessori and Technology. *Montessori Life*, 11(4):39, 1999.

11 Maria Montessori. *The Discovery of the Child*. Kalakshetra, Adyar, Madras, India, 1948.

12 Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and S. Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24:45–77, January 2007.

13 Steffi Rudel. Programmieren lernen an einer Montessori-Schule: Der EmiLe Computer Club (ECC). In Daniel Demmler, Hannes Krupka, and Hannes Federrath, editors, *Lecture Notes in Informatics (LNI) - Proceedings*, volume P-326, pages 1213–1219, Bonn, 2022. Gesellschaft für Informatik. doi:10.18420/INF2022_104.

14 Schumacher, Eva. *Montessori-Pädagogik verstehen, anwenden und erleben*. Individuelles Lernen mit Montessori. Beltz Verlag, Weinheim, Basel, 2016.

15 Stefano Scippo and Fabio Ardolino. Computational thinking in Montessori primary school. *Ricerche di Pedagogia e Didattica. Journal of Theories and Research in Education*, 16(2):59–76, 2021. Epistemological intersections between human and natural sciences in the thought and works of Maria Montessori [Special Issue]. doi:10.6092/ISSN.1970-2221/12163.

16 Mario Valle. *Montessori-Pädagogik und neue Technologien: Eine mögliche Integration?*, volume 33 of *Impulse der Reformpädagogik*. LIT, Berlin, 2019. Title of the original Italian edition 2017: La pedagogia Montessori e le nueve tecnologie. Un'integrazione possible?

17 Markus Wurster. Internet und digitale Strukturen im Grundschulalter. *Montessori. Zeitschrift für Montessori-Pädagogik*, 2, 2019.

18 Oren Zuckerman, Saeed Arida, and Mitchel Resnick. Extending tangible interfaces for education: digital montessori-inspired manipulatives. In Wendy Kellog, Shumin Zhai, Gerrit van der Veer, and Carolyn Gale, editors, *CHI05: CHI 2005 Conference on Human Factors in Computing Systems*, pages 859–868, Portland Oregon USA, 2005. ACM. doi:10.1145/1054972.1055093.