Algo-AR: Development of an Augmented Reality-Supported Tangible Programming Tool to Improve Algorithmic Thinking Skills

Araştırma Makalesi/Research Article

Zeynep ÇİPİLOĞLU YILDIZ[1]*, Süleyman DOĞAN[2]

[1]Bilgisayar Mühendisliği, Manisa Celal Bayar Üniversitesi, Manisa, Türkiye,
[2] Bilgisayar Mühendisliği, Manisa Celal Bayar Üniversitesi, Manisa, Türkiye

zeynep.cipiloglu@cbu.edu.tr, suleyman@codecosoft.com

(Geliş/Received:08.12.2023; Kabul/Accepted:02.02.2024)
DOI: 10.17671/gazibtd.1398781

Abstract— The main purpose of this study is to develop an educational tool to help children acquire algorithmic thinking skills at an early age while having fun. The methodology combines modern technologies and approaches such as augmented reality (AR), gamification, and tangible user interfaces. In this application, the coding components consist of specially designed tangible command blocks in the form of jigsaw puzzle pieces. The application contains a 3D multi-level game environment, and the user is expected to control the game character by constructing an algorithm with physical command blocks. The constructed algorithm is scanned using a mobile AR application and converted into code that controls the game environment. The major design considerations during the development of this application were simplicity and accessibility. All design decisions were delineated extensively in the paper. In conclusion, an augmented reality-based gamified tangible programming kit is proposed to improve children's algorithmic thinking skills at an early age. The application requires only a smartphone and printable command blocks. Thus, an inexpensive, accessible, and entertaining educational tool is developed.

Keywords— algorithmic thinking, augmented reality (AR), gamification, heuristic evaluation, STEM, tangible user interfaces (TUIs)

Algo-AR: Algoritmik Düşünme Becerilerinin Geliştirilmesi İçin Artırılmış Gerçeklik Destekli bir Somut Programlama Aracı Geliştirilmesi

Özet— Bu çalışmanın temel amacı, çocukların eğlenirken algoritmik düşünme becerilerini erken yaşta kazanmalarına yardımcı olacak bir eğitim aracı geliştirmektir. Metodoloji, artırılmış gerçeklik (AG), oyunlaştırma ve somut kullanıcı arayüzleri gibi modern teknolojileri ve yaklaşımları birleştirmektedir. Bu uygulamada kodlama bileşenleri, yapboz parçaları şeklinde özel olarak tasarlanmış fiziksel komut bloklarından oluşmaktadır. Uygulama, çeşitli seviyeleri olan üç boyutlu bir oyun ortamı içermekte ve kullanıcının fiziksel komut blokları ile bir algoritma oluşturarak oyun karakterini kontrol etmesi beklenmektedir. Oluşturulan algoritma bir mobil artırılmış gerçeklik uygulaması ile taranmakta ve oyun ortamını kontrol eden koda dönüştürülmektedir. Bu uygulamanın geliştirilmesi sırasında tasarımda dikkat edilen başlıca hususlar basitlik ve erişilebilirlik olmuştur. Tüm tasarım kararları makalede kapsamlı bir şekilde açıklanmıştır. Sonuç olarak, erken yaştaki çocukların algoritmik becerilerini geliştirmek için artırılmış gerçeklik destekli oyunlaştırılmış somut bir programlama aracı önerilmiştir. Uygulama yalnızca bir akıllı telefona ve yazdırılabilir komut bloklarına ihtiyaç duymaktadır. Böylece ucuz, erişilebilir ve eğlenceli bir eğitim aracı geliştirilmiştir.

Anahtar Kelimeler— algoritmik düşünme, artırılmış gerçeklik (AG), oyunlaştırma, sezgisel değerlendirme, STEM, somut kullanıcı arayüzleri

1. INTRODUCTION

We live in an era of digital transformation where software development and programming skills are essential. Computational thinking (CT), algorithm design, algorithmic thinking and coding skills are extremely important for today and the future. Acquiring these skills becomes more difficult with age. Therefore, activities that develop computational and algorithmic thinking skills must be embraced in preschool and primary education curricula.

Many different approaches and tools have been developed to improve these skills. These approaches can be broadly divided into two categories "plugged" and "unplugged", in terms of whether they rely on computers. The plugged category includes traditional text-based or visual programming tools. There are also unplugged approaches, which are free of computers. In these applications, problem-solving activities are carried out using real-world objects such as paper and pencil.

Although text-based programming languages and tools are traditionally dominant for programming, visual programming languages are also widely used, especially by children, to gain programming/coding skills. In visual programming languages and tools, programming elements are represented by graphical components rather than textual commands, allowing users to control the program flow by directly manipulating these elements. The most common visual programming tools are Scratch [1], [2], Alice [3], and Blockly [4]. Visual programming languages make the programming process more understandable, concrete and fun because they consist of visual elements and work with the familiar drag-and-drop method. Nevertheless, like traditional programming languages, they still require a computer during the programming phase. However, overuse of computers is known to cause problems such as screen addiction for children.

Alternatively, tangible programming languages and tools have been developed. Such languages allow users to manipulate various physical objects to control virtual objects or physical robots [5]. They can be referred as hybrid programming environments. In many applications of tangible programming such as Algoblocks [6], E-Block [7], and LEGO Mindstorms [8], the control objects contain some electronic and/or mechanical components, which increases the cost and limits the accessibility. Durable and low-cost solutions have also been proposed for the use of tangible programming languages in the classroom environment [9], [10]. These studies generally use low-level image processing techniques, which can lead to computational costs and recognition errors, or some robotic components.

The main objective of this work is to develop simple, comprehensible, accessible, and enjoyable material that will help children acquire algorithmic thinking and programming skills. We also avoid a fully plugged approach, as the target audience is preschool children. To achieve this, the following contemporary methodologies underpin our approach:

- *Gamification:* It is known that one of the best teaching methods for young children is the gamification technique [11], [12]. The aim of employing gamification is to make the learning process more fun, instructive, and memorable.
- *Tangible programming:* A key concern for children's physical and mental health is the overuse of computers. Parents generally limit their children's screen time. The adoption of a tangible programming approach makes the process almost "unplugged".
- *Augmented reality (AR):* The benefits of using AR technology are manifold. First, it requires only a smartphone or tablet, making it a low-cost and accessible kit. Second, it enables a flexible tool that is content-renewable, as virtual objects can be placed in the real world. Thirdly, it contributes to entertainment and engagement [13]. Finally, it provides a robust and computationally inexpensive method for detecting code blocks, compared to using low-level image processing operations.

In the proposed application, a maze-like game environment is superimposed on the user's real world using AR technology, and the game character is controlled by the player's algorithm constructed using tangible command blocks.

A common problem with tangible AR applications is that there are no well-defined design standards as there are for traditional desktop UIs. Therefore, we take a design-centric view throughout the paper. For the benefit of researchers in the field, the design trade-offs, implementation decisions, heuristic evaluation results, limitations and possible improvements have been detailed.

The rest of the paper is organized as follows: The next section provides a summary of the related studies in the literature. The third section explains the tools and methodologies used to develop the proposed application. In the fourth section, the results of the heuristic evaluation are explained. Comparison of our approach to other similar approaches is performed. The limitations and future work are then described. The last section summarizes and concludes the study.

2. RELATED WORK

The use of AR technology in education is known to make the learning process more enjoyable and memorable [14], [15]. AR technology has been used effectively in many different fields of education [14]–[17]. The use of AR technology for teaching algorithms and programming is relatively new. Experimental studies have shown that AR technology significantly contributes to learning and entertainment factors [13], [18]. Today, tangible programming tools have begun to be combined with AR technology [19]. For recent literature reviews about different applications of AR, serious games, and tangible interfaces in education, see the papers [20]–[22].

AR Scratch [23] provides features that extend the Scratch programming language to enable the development of simple AR applications. In the ARMaze [24] application, the visual codes on the cube-shaped programming blocks are recognized by an image processing library and the interpreted commands move the character around a maze using AR technology. Similarly, there are applications such as HyperCubes [25] and CodeCubes [18] where programming commands are represented by cubes.

Another example of tangible AR is the ARQuest [26] application, which enables team collaboration using a client-server architecture. In this application, users can design the game environment and present it to other teams. This application appeals to the 9-10 age group and does not contain advanced programming components such as loops and conditions.

The Code Bits [27] application consists of code blocks that are designed to be printed on paper to be affordable and accessible, as in our study. In the CodeBits application, there are no commands such as loops and conditions, and the commands are scanned and transferred to the game environment one at a time. Although this is good for debugging purposes, it can limit usability.

Command blocks in the Code Notes [28] application are in the form of cards and consist of English phrases such as “Turn position to the left, draw a tree” instead of visual codes. Text recognition techniques are used to interpret the commands in the cards. It is stated that the application is mainly aimed at the 12-13 age group. Similarly, the Kart-ON [29] application aims to provide an affordable, extensible, and expressive programming environment using paper programming and relying on text recognition.

The literature review shows that although there are studies on the use of tangible programming tools to teach algorithmic thinking and coding skills to children, these studies have not yet reached saturation point. Most of these tools involve microcontrollers and mechanical components and/or are paid for. This reduces accessibility due to cost. At this point, combining AR technology with tangible programming tools is an alternative. In these studies, programming commands generally consist of blocks in the form of cubes, and visual codes consist of abstract shapes, as they aim to facilitate recognition by the artificial vision system rather than user perception. In addition, current studies generally fail to explain the rationale behind design choices and provide limited insight for further research.

The key contributions of this study and the advantages of the proposed solution can be summarized as follows:

- An affordable and simple educational tool has been developed to improve children's algorithmic thinking skills.
- The primary user group is 5-10 year old children. The application and command blocks have been designed with the target user group in mind.
- The proposed solution offers a compromise between plugged and unplugged approaches to algorithmic thinking activities by combining AR and tangible programming.
- The tool contains no electronic or mechanical components. This makes it cheap and safe.
- The solution does not rely on low-level image processing techniques that can degrade the system's recognition accuracy and interactivity.
- It has a broader command gamut compared to similar tools, including conditions and loops.
- The game environment is presented in 3D and is attached to the real world using AR technology to enhance realism and engagement.
- The results of the usability analysis, design trade-offs, and implementation decisions are explained to help other researchers and designers.

3. METHODOLOGY

This section first describes the use of the proposed kit. It then summarizes the tools and techniques used to develop the proposed system. This is followed by the details of the rationale behind the design and implementation choices, together with possible design trade-offs.

3.1. Overview

In the proposed application, there is a 3D game environment, such as a maze, and the game character must be controlled to move to the target location. The steps performed by the user are shown in Figure 1, and an example view during gameplay is shown in Figure 2.

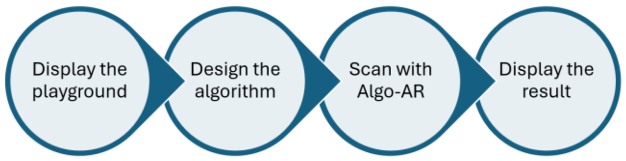


Figure 1. Steps of the usage of the application

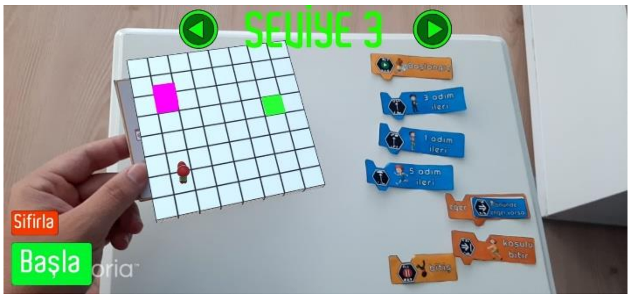


Figure 2. A sample view during gameplay

First, the player inspects the problem on the virtual grid that is displayed on top of the physical gaming platform using AR technology. The player then places specially designed command blocks as physical jigsaw puzzle pieces to create the required algorithm. These physical command blocks are scanned and converted into code using the proposed mobile AR application, and the game character moves in the digital environment according to these codes. With the help of AR technology, the playground is displayed in the user's physical environment, increasing fun, motivation, and engagement.

3.2. Command Blocks

The first step was to determine the commands that would be used to control the character's movements. The most necessary commands have been implemented for this prototype, but it is planned to increase the variety of commands in future developments. The commands in the current prototype are: *Start (Başlangıç), End (bitiş), Move n steps (n adım ileri), Turn right/left (sağa/sola dön), Repeat n times (n defa tekrarla),* and *If (eğer)*. Therefore, the current prototype involves the most important concepts of programming, including loops and conditions. The users organize the command blocks to build their algorithms. The algorithms must be placed between the *start* and *end* commands. The same background colour is used for the complementary commands (i.e. start-end). The visual codes to be used for the reocgnition of the blocks were generated using the VuMark tool[1]. These command blocks are shown in Figure 3 and printable command blocks are available in the Appendix.

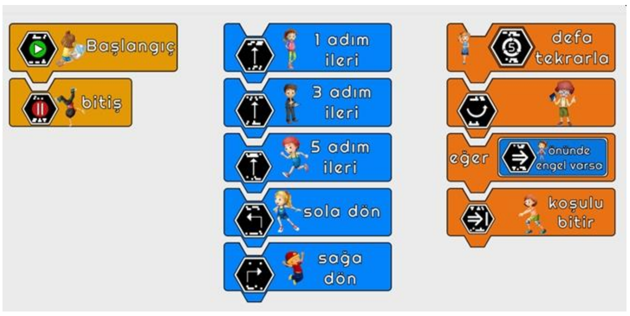


Figure 3. Command blocks

3.3. Mobile AR Application

A mobile AR application called Algo-AR was developed using the Unity[2] game engine and the Vuforia SDK[3]. The playground was designed as a 3D grid-like structure in which some cells may have obstacles. The game has several levels of increasing difficulty. Using command blocks, the player is expected to create an algorithm that will transport the game character from its current location to the target location without hitting obstacles. The installation file of the application and sample videos on how to use the application can be found in the Appendix.

Figure 4 shows an example of the process of converting command blocks into game control. After the users physically create the necessary algorithm using the command blocks, they scan this algorithm with the camera through the Algo-AR application. For the recognition of the command blocks, we use marker-based AR technology. In marker-based AR, where the digital content will be placed is pre-defined to the system with a marker. For this purpose, the image targets of the command blocks and the corresponding text codes were matched using the Vuforia SDK. Target images are recognized by the detector of the Vuforia platform. The recognized text codes corresponding to the image targets are converted by the application into C# functions that provide the necessary parameters for the character's movement, and these functions are executed in the game by the Unity game engine.

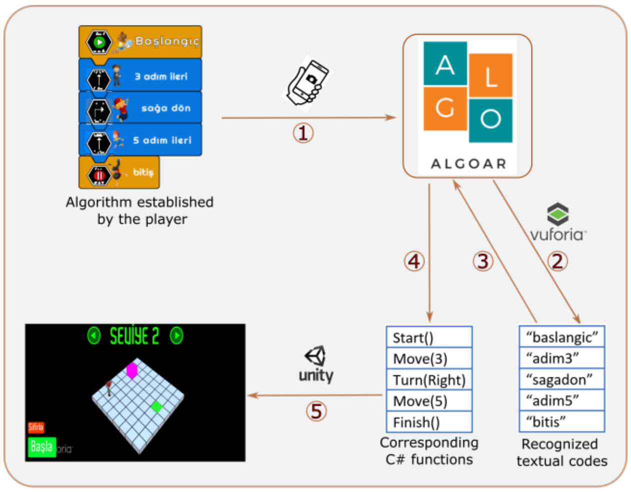


Figure 4. The process of converting the command blocks to game control

```
Definition of the Game Loop:
Start
   L = SelectLevel()
   InspectPuzzle(L)

   Algorithm = ConstructAlgorithm()
   RunAlgorithm(Algorithm)

   DisplaySolution()
End


Function ConstructAlgorithm:
   Start
       PuzzlePieces = DetectPuzzlePiecesFromCamera(Puzzle)

       Display(PuzzlePieces)

       CommandList = AddCommandToList(PuzzlePieces)

       Return CommandList
   End
```

Figure 5. Pseudocode describing the game loop process.

[1] https://library.vuforia.com/objects/vumarks

[2] https://unity.com/

[3] https://developer.vuforia.com/

In Figure 5, a description of the game process is given in a pseudocode format. When the game starts, the player selects the level, inspects the puzzle, constructs the algorithm using the blocks then scans the algorithm with the camera. Detection of the puzzle pieces by the system and the construction of the command list from the detected pieces is also given in the second part (Function ConstructAlgorithm) of the pseudocode. The returned algorithm is executed on the game board and the resulting animation is displayed.

3.4. Design Rationale

This section explains critical implementation issues and major design goals that were considered in the development of the current prototype.

1) *Cost:* The proposed tool requires only a smartphone or a tablet and command blocks that can be printed on a piece of paper. This makes it an inexpensive kit.
2) *Accessibility:* As a direct consequence of the low cost, this tool is easily accessible to people on low incomes. In addition, since visual codes are used, illiterate children can also use this material. Although the primary audience of the application is children, it is suitable for users of all ages who are new to algorithmic skills.
3) *Durability:* The proposed tool does not require complex and fragile mechanical or electrical components as many popular tangible programming tools do. Thus, there is no problem with the breakdown of these components. In the worst-case scenario, if the printed command blocks are torn, they can be reprinted. It is also possible to use durable materials such as wood or plastic for the command blocks. Although this will increase the cost, it will enhance the overall user experience.
4) *Safety:* As the target user group is children, special attention must be paid to the safety of the material. As mentioned above, the kit does not contain any electrical components, so it is quite safe to use.
5) *Engagement*: Fun is an important factor in increasing people's motivation and learning rate. The more people feel involved in the application, the more they are motivated to learn. To increase fun and motivation, the application is designed as a game with levels of increasing difficulty. Additionally, the game environment and the characters are designed in 3D, which also contributes to engagement. Nevertheless, more visual/audio effects and game levels are required to raise the fun factor of the game. In other respects, the usage of the application requires a focus switch between physical command blocks and the virtual content on the screen, which may damage engagement.
6) *Unplugged usage:* Most similar applications aimed at teaching children to program or design algorithms require coding with a computer or mobile phone. Such applications can be referred to as “plugged-in” tools. However, social/mental/physiological problems or digital addiction may arise if too much time is spent with digital devices, especially at a young age. Completely “unplugged” tools are also available to enrich computational thinking without computers [30], [31]. We offer a compromise between plugged and unplugged tools by combining AR and tangible programming. Since the algorithm construction is done using physical command blocks, the plugged usage time is very short. When the game environment is projected on a shared display, parents or teachers can control the application with their smartphones. This can be considered unplugged use for children. This allows for cooperation as well as competition. Therefore, depending on the age of the users, the usage process can be adjusted as plugged or unplugged. This is another flexibility of the proposed tool.
7) *Design of the command blocks:* An image target was created to represent each command block. When creating image targets, it is important to have different images for different commands and unique details so that the application can easily recognize the images. However, completely abstract shapes such as QR codes were avoided so as not to complicate children’s perceptions. Instead, black-and-white image targets were created using the VuMark tool to identify the commands. Additional visual images were also added for human perception. Although the image targets uniquely identify the commands, text equivalents for each command were also included in the command blocks. Turkish was preferred as the text language for the presentations, but in practice, no modification is required to change the text language. For this reason, command blocks can be used in any language, even without text.
8) *Accuracy and efficiency:* Instead of training our models with low-level image processing and computer vision techniques, the marker-based AR method was found to be suitable for recognising the command blocks. This is because in this problem, unlike a standard object recognition problem, the images to be recognized are specified and fixed at the beginning. In addition, due to the variable environmental conditions in mobile environments and the real-time interactivity requirement of the application, a robust solution with low computational cost is needed. Therefore, employing marker-based AR techniques was found to be the most appropriate solution.

3.5. Design Trade-offs

In this section, we summarize the major trade-offs that we encountered during the design and implementation of the proposed tool. In some of these situations, we force the user to meet a specific design goal, while some trade-offs are flexible and left to the user’s control.

1) *Cost/affordability vs. usability:* Command blocks can be printed on a piece of paper for cost-effectiveness,

but from a usability perspective it is easier to use other firm but more expensive materials such as wood or plastic. At this point, the application does not have a restriction and the user has control over this choice. Another issue related to this trade-off is the use of mobile AR or AR glasses. While AR glasses allow for hands-free use, they are not very accessible due to their cost. For this reason, we have traded off cost against usability in this question.

2) *Cost/affordability vs. durability:* As mentioned earlier, the user can choose the material for the command blocks to be printed. This can be a piece of paper (for low cost) or 3D printed using other durable materials.

3) *Error prevention (affordance) vs. usability:* The Jigsaw puzzle-like form of the command blocks caters to affordance and hence error prevention. However, this may limit usability somewhat when paper is the material for the command blocks. The tool still works even if the user cannot place the blocks correctly concerning the interlocking mechanism. However, the user is in control of the choice for this compromise and may prefer to use a different material for a better user experience.

4) *Simplicity vs. expressiveness:* A wider range of commands, including other programming concepts, would provide a more expressive tool. However, given the age of the target audience, we preferred to keep it simple and understandable. Similarly, the number of command blocks in a particular algorithm is deliberately limited to simplify comprehension and avoid visual clutter on the screen. This was also suggested by the usability experts who evaluated our prototype.

5) *Accessibility vs. expressiveness:* Several alternatives were considered for the design of the command blocks. Firstly, a text-based approach was considered to provide an expressive and extensible tool, but this method was abandoned bearing in mind the illiterate children and the method's dependence on the text language. The idea of assigning distinctive images to command blocks was found to be more appropriate and non-abstract visuals were determined for children.

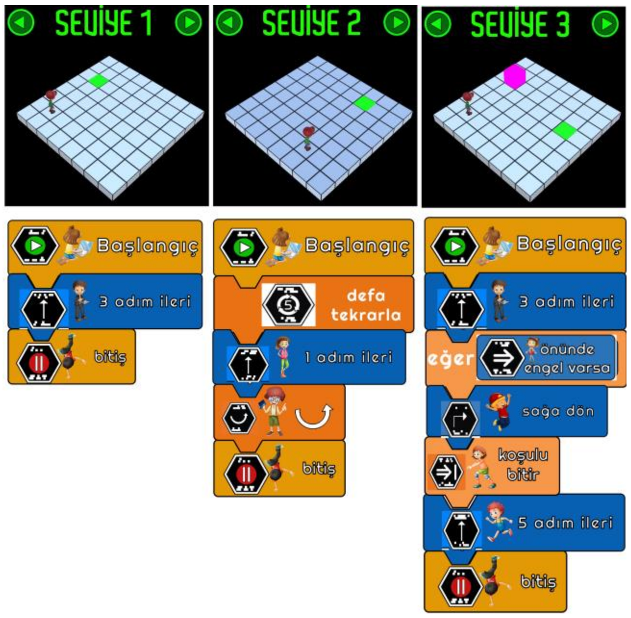
6) *Accuracy vs. usability:* For the scanning process of the algorithm, we contemplated three options: *i)* scanning the whole algorithm and playing its result on the game at once, *ii)* scanning and playing the blocks one by one (the idea in [27]), or *iii)* scanning the blocks one by one and playing the whole recognized algorithm at once. Scanning the whole algorithm (option *i*) would be more usable and intuitive, as it was also suggested to us by a heuristic evaluator, but it requires more computation and low-level image processing operations. This could reduce the recognition accuracy and efficiency. Therefore we preferred to scan the blocks sequentially one at a time. Nevertheless, since the second option limits usability more than the last, the current prototype supports only sequential scanning and full replay (option *iii*). However, we plan to evaluate these options from both usability and efficiency perspectives and update the next prototype of the application accordingly.

4. HEURISTIC EVALUATION

To assess the usability of the proposed system for the target user group and to identify design problems at an early stage, we carried out a heuristic evaluation process, which is a usability testing method in which experts assess the usability of the system against a set of pre-defined criteria. It can be argued that heuristic evaluation is limited in the sense that the evaluators are not part of the target audience. However, it is a very powerful tool because it provides a cost-effective way of detecting design problems early in the development process.

4.1. Evaluation Methodology

Three usability experts independently assessed the proposed application according to Nielsen's usability heuristics [32]. They benefitted from the Heuristic Evaluation Workbook provided by Nielsen Norman Group[4]. They are informed about the purpose and target users of the application. In their evaluations, they were asked to consider children aged 5-10, especially pre-schoolers, as the target user group. The evaluators first inspected the application freely, then they were shown the main functionality and finally, they played the game levels shown in Figure 6. They used command blocks printed on paper but were also informed about the possibility of using other materials.



4 https://media.nngroup.com/media/articles/attachments/Heuristic_Evaluation_Workbook_1_Fillable.pdf

Figure 6. Top – sample game levels, and bottom – their solutions

4.2. Findings

The evaluators' findings are explained below according to the predefined usability criteria. These findings are also summarized in Table 1. This table lists the key strengths of the tool and recommended improvements for each usability criterion.

1) *Visibility of system status* ("The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.")

When the application recognizes a command block, visual and auditory feedback is presented to the user, and the recognized algorithm is displayed online. In Figure 7, the algorithm constructed by the player using physical command blocks is shown on the left. On the right, the recognized commands are displayed virtually as they are scanned.

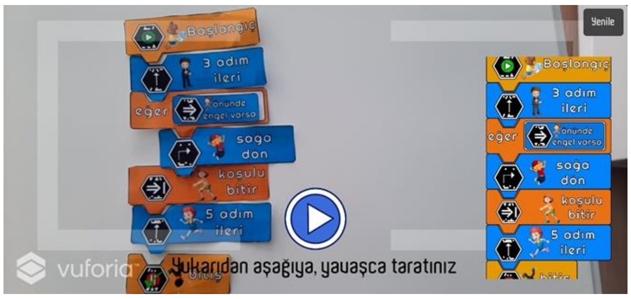


Figure 7. A screenshot from the gameplay, while scanning the constructed algorithm

2) *Match between the system and the real world* ("The system should resemble the experiences that users already had. The design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.")

The command blocks are designed as interlocking jigsaw puzzle pieces and are used based on the intuitive drag-and-drop metaphor. This is consistent with real-life puzzle games and popular visual programming languages. It also prepares children for block-based programming tools, which are generally the next level in the learning process. The use of the same colour for the complementary command blocks (i.e. start-end) is also a good choice. Although the images on the command blocks generally match universal visual codes such as arrows, play buttons, numbers, etc., the visual code for the *if* command block is somewhat ambiguous.

3) *User control and freedom* ("Users should be able to reverse their action if done by mistake.")

The application provides a visible "Sıfırla (Reset)" button to reset the algorithm. The choice of red colour for this button is considered appropriate as it is a dangerous operation that requires attention. However, an Undo operation should also be provided to allow users to reverse their actions taken in error.

4) *Consistency and standard* ("Follow platform and industry conventions. Similar system elements should look similar.")

AR interfaces and tangible user interfaces are relatively new fields. As a result, there are no widely accepted industry standards as there are for 2D interfaces. On the other hand, the puzzle-like form of the command blocks is consistent with common visual programming conventions.

5) *Error prevention* ("Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.")

Another advantage of the jigsaw puzzle form of the command blocks is that it provides affordance and thus prevents possible syntax errors. Furthermore, the feedback (see item 1) made available to the user when a command block is recognized by the application prevents potential errors.

6) *Recognition rather than recall* ("Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design should be visible or easily retrievable when needed.")

The application is easy to learn because the command blocks are designed like puzzle pieces that everyone is familiar with from everyday life or common visual programming languages. The simple design of the tool contributes to learnability as well. Expressive and memorable visual codes have generally been assigned to the command blocks. Text codes are also helpful in identifying the meaning of a block, but they are helpless for illiterate children. The control buttons, "Başla (Start)" and "Sıfırla (Reset)", should also have familiar icons rather than text, taking into account illiteracy. These buttons are designed to be visible, but in the context of AR, background and lighting conditions may affect visibility. Similarly, the nature of AR requires switching focus between real and virtual content, which can generate additional cognitive load.

7) *Flexibility and efficiency of use* ("both new and experienced users should be able to efficiently use the system")

The target user group consists of young children and the application does not generally require shortcuts for experienced users. On the other hand, the developed tool offers some flexible options that users can customize according to their needs. First, the user can print command blocks on a piece of paper or any other material. Second, the language of the expressions in the command blocks can be changed or completely removed. Third, although the command blocks were originally designed as puzzle pieces, they can also be used flat, as it can be difficult to use puzzle pieces when paper command blocks are preferred.

One evaluator made the following suggestions in addition to the common opinions listed above. The application was designed for mobile AR, which requires holding the mobile device. Adapting it for AR glasses is also suggested to allow hands-free and comfortable use. Scanning the whole algorithm at once could be more efficient and usable.

8) *An aesthetic and minimalist design* ("Interfaces should not contain information that is irrelevant or rarely needed.")

The interface and the system have been designed in a fairly simple and minimalist manner. However, the game grid and the look and feel of the UI elements could be more aesthetically pleasing. Limiting the overall command gamut is a sound decision, given the age of the target audience and the short-term memory limitations of the human brain. Similarly, the total number of command blocks required to construct a particular algorithm should also be limited. Otherwise, long algorithms will result in visual clutter on the screen, and the scanning of the algorithm will become more difficult as the length of the algorithm increases.

9) *Help users recognize, diagnose, and recover from errors* ("Error messages should be expressed in plain language, precisely indicate the problem, and constructively suggest a solution.")

Although it is very rare, if the user inadvertently moves the mobile device during the sequential scanning process, the application may fail to recognize some command blocks. In such cases, the user is informed of the situation as the recognized blocks are displayed on the screen. However, the only way to recover from this situation is to reset the algorithm. An Undo operation must also be made available to the user. In addition, the errors in the constructed algorithm could be shown to the user.

10) *Help and documentation:*

More detailed documentation, including the meaning of the blocks, sample usage scenarios, etc. could be provided. Online textual instructions on how to use the system should also be provided as audio messages, taking into account the illiterate population in the target user group.

Table 1. Summary of the heuristic evaluation (C1-C10 refers to the evaluation criteria, in the above order)

	Strengths	*Recommendations*
C1	Visual/auditory feedback	
C2	Intuitiveness Use of metaphors	Revision of the visual codes
C3	Visible control buttons	Undo operation
C4	Puzzle-like form Drag-and-drop metaphor	
C5	Affordance Feedback	
C6	Easy to learn Familiar elements Simplicity contributes to learnability Memorable visual codes	Audio messages for illiterate children Familiar icons for control buttons
C7	Flexible options for command blocks	Adaptation for AR glasses Scanning the algorithm at once
C8	Simple and minimalist design Limited command gamut	Refining the look-and-feel
C9		Undo operation Showing the algorithm errors
C10		More detailed documentation Audial instructions

5. COMPARISON TO OTHER APPROACHES

In this section, we perform a qualitative comparison of our kit with other alternative tools for cultivating computational thinking. We make this comparison in the context of our target user group (preschoolers) and our main design goals: "unplugged use", "simplicity", and "accessibility/affordability".

In Table 2, interface types and main methodologies of similar studies are listed. The interface type refers to the usage of a tool and we categorize it as *Unplugged*, *Plugged*, and *Hybrid*. Unplugged interfaces rely on traditional methods and do not use any computers, plugged interfaces require a computer and/or electronic components, while hybrid interfaces require computers only for specific stages. Algo-AR falls into the hybrid interface category by combining tangible programming and AR as mentioned before, since a mobile phone is only required for displaying the results. Limiting screen time is one of our crucial goals because of the target users. Compared to totally unplugged activities, AR increases the fun factor.

Table 3 evaluates similar studies in the literature in the context of the design goal of simplicity. Preschool children are the target user group, therefore the design, usage and comprised programming concepts should be simple. As indicated in the table, some of the tools are text-based and not suitable for illiterate children, some contain complex programming concepts, and some are sophisticated for little children. Algo-AR has a simple interface and limits the complexity of algorithms and programming concepts.

Accessibility and affordability are also among our primary design objectives. Therefore, the studies in the literature are also elaborated from this perspective and the summary is given in Table 4. Unplugged activities can be considered as the most accessible category. There are also free visual programming tools that only require a computer. Some of the tools, such as Algoblocks and E-Block, necessitate special hardware for the command cubes, which limits their accessibility. On the other hand, most of the tools in the hybrid interface category, rely on a mobile device and paper-based command blocks. This provides a low-cost and accessible solution.

Table 2. List of the studies according to interface type and methodologies

Studies	*Interface*	*Tools/Methodologies*
[30] [31]	Unplugged	Paper, pen, etc.
Scratch [1], [2] Alice [3] Blockly [4]	Plugged	Visual programming
Algoblocks [6]	Plugged	Tangible programming
E-Block [7]	Plugged	Tangible programming, Microcontrollers
HyperCubes [25]	Hybrid	Tangible programming, AR, Spatial tracking
CodeCubes [18]	Hybrid	Tangible programming, Marker-based AR
ARQuest [26]	Hybrid	Tangible programming, Marker-based AR
Code Bits [27]	Hybrid	Tangible programming, Marker-based AR
Code Notes [28]	Hybrid	Tangible programming, Computer vision, Text recognition
Kart-ON [29]	Hybrid	Tangible programming, AR, Text recognition
Algo-AR	Hybrid	Tangible programming, Marker-based AR

Table 3. List of the studies according to simplicity design goal

Studies	*Simplicity*
[30] [31]	Simple to use, appeals to all ages of children
Scratch [1], [2] Alice [3] Blockly [4]	Simple drag-and-drop metaphor Not for illiterate children in general Comprises many programming concepts Not suitable for preschoolers
Algoblocks [6]	Appeals to primary and secondary school Includes conditionals, loops, parameters, and basic movement commands Requires command cubes connected with cables which limit usability
E-Block [7]	Appeals to 5-9 age Includes basic movement commands
HyperCubes [25]	Appeals to late elementary and middle school Includes visual and sound inputs Requires constructing command cubes Complex for preschoolers since it includes abstract markers and parameters are adjusted by a menu
CodeCubes [18]	Appeals to 13-14 age Includes basic movement commands Requires constructing command cubes
ARQuest [26]	Appeals to primary school (9-10 age) Includes basic movement commands Designed for collaborative usage
Code Bits [27]	Preschoolers can use Includes basic movement commands Commands are transferred to the application one-by-one
Code Notes [28]	Based on English text Not suitable for illiterate children Includes a wide range of commands
Kart-ON [29]	Based on English text Not suitable for illiterate children Includes a wide range of generalizable commands
Algo-AR	Simple command gamut (loops, conditionals, basic movement) Simple algorithm complexity Simple drag-and-drop metaphor Suitable for illiterate and preschool children

Table 4. List of the studies according to accessibility design goal

Studies	*Accessibility/Affordability*
[30] [31]	Requires basic materials such as paper, pen So, very cheap and accessible in general
Scratch [1], [2] Alice [3] Blockly [4]	Requires a computer Freely accessible in general
Algoblocks [6]	Requires a computer and special wired command cubes, so not much accessible
E-Block [7]	Requires a computer and command blocks that include microcomputers, infrared transmitters and receivers, batteries, wireless modules, and LEDs. So they are not much accessible
HyperCubes [25]	Requires a mobile device and paper cubes
CodeCubes [18]	Requires a mobile device and paper cubes
ARQuest [26]	Requires a computer, a mobile device and paper tokens
Code Bits [27]	Requires a mobile device and paper tokens
Code Notes [28]	Requires a mobile device and paper tokens
Kart-ON [29]	Requires a mobile device and paper tokens
Algo-AR	Requires a mobile device and paper blocks

6. LIMITATIONS AND FUTURE WORK

Despite its many benefits, the current prototype has some shortfalls and potential for improvement. These improvements can be considered under five main categories: refining the usability aspects, reinforcing the pedagogical aspect, enriching the content, increasing the supportability, and conducting comprehensive evaluations.

Firstly, the user interface and usability of the tool should be improved in light of the results of the heuristic evaluation. These improvements should include refining the visual codes on the command blocks in terms of expressiveness and discoverability, adding an undo operation, considering illiteracy in the design of the UI elements, and improving the documentation.

Second, the current prototype supports programming concepts such as condition and loop that are not found in many similar applications. Considering that people will not develop very complex algorithms during the learning phase, it can be said that the current application has functional competence in terms of targeted acquisition of algorithmic thinking. However, more programming elements such as arithmetic operations, functions and variable definitions can be added. Correct solutions and errors should be displayed on the application. It could also be useful to enable the teacher/parent to track the progress of the students.

Third, more levels should be added to the game to increase the fun and motivational factors. The following improvements can be considered, which will also affect the difficulty level of the game: time limit, types of obstacles, bonus items, and shortest path problems. The application should be made more impressive and customizable by enriching it with more animations, visual effects, sound/music, and virtual environment/character themes. Game themes and difficulty levels should be determined according to age groups, taking into account the views of educators and pedagogues.

Fourth, the current prototype works on Android devices, but it is possible to easily deploy it for other operating systems. Its use with AR glasses, however, should be specifically tested and the application should be adapted to the needs of such use. Since one of the main design goals is affordability, we have not yet considered the use of AR glasses.

Finally, although the usability aspect has been assessed by some experts, the tool has not yet been tested by the target user group. Thus, comprehensive user studies are required to evaluate different aspects of the user experience and to draw generalizable conclusions. The application should be extensively tested with different age groups and its long-term effects in different dimensions (learning, entertainment, motivation, cognition, etc.) should be statistically analysed.

7. CONCLUSION

A prototype application has been developed to improve children's algorithmic thinking skills. The main design goals are simplicity, affordability, entertainment, and unplugged usage. The application has a gamified structure and is intended to be particularly interesting for children. Although the primary audience of the application is children, it is suitable for users of all ages who are just starting to learn algorithm construction. The proposed application has been developed using modern technologies such as AR and tangible programming. A design-oriented view of the work could be of great benefit to the researchers in the field. In summary, a low-cost, flexible, unplugged educational material for the development of algorithmic thinking skills at an early age is proposed and elaborated.

APPENDIX

Materials related to this study can be downloaded from the links obtained by scanning the QR codes in Figure 8. The downloadable materials include sample videos demonstrating the use of the game, the installation file of the Algo-AR application for Android devices, and the printable command blocks.

Sample videos | Algo-AR APK | Printable command blocks

Figure 8. OR-codes for accessing the sample material of the proposed tool

REFERENCES

[1] J. Fagerlund, P. Häkkinen, M. Vesisenaho, J. Viiri, "Computational thinking in programming with Scratch in primary schools: A systematic review," *Comput. Appl. Eng. Educ.*, 29(1), 12–28, 2021.

[2] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, M. Resnick, "Scratch: a sneak preview [education]," **Second International Conference on Creating, Connecting and Collaborating through Computing**, Kyoto, 104–109, 2004.

[3] M. Conway, S. Audia, T. Burnette, D. Cosgrove, K. Christiansen, "Alice: lessons learned from building a 3D system for novices," **SIGCHI Conference on Human Factors in Computing Systems**, Netherlands, 486–493, 2000.

[4] N. Fraser, "Ten things we've learned from Blockly," **IEEE Blocks and Beyond Workshop (Blocks and Beyond)**, USA, 49–50, 2015.

[5] A. Strawhacker, M. U. Bers, "'I want my robot to look for food': Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces," *Int. J. Technol. Des. Educ.*, 25, 293–319, 2015.

[6] H. Suzuki, H. Kato, "Algoblock: a tangible programming language, a tool for collaborative learning," **4th European Logo Conference,** 297–303, 1993.

[7] D. Wang, Y. Zhang, S. Chen, "E-block: A tangible programming tool with graphical blocks," *Math. Probl. Eng.*, 2013, doi: 10.1155/2013/598547.

[8] F. Klassner, S. D. Anderson, "LEGO MindStorms: Not just for K-12 anymore," *IEEE Robotics and Automation Magazine*, 10(2), 12–18, 2003, doi: 10.1109/MRA.2003.1213611.

[9] M. S. Horn, R. J. K. Jacob, "Designing tangible programming languages for classroom use," **International Conference on Tangible and Embedded Interaction**, Louisiana, 159–162, 2007, doi: 10.1145/1226969.1227003.

[10] E. Naude, A. Fowler, R. Lemon, C. J. Sutherland, "Kupe's Journey: Building a Low-cost, Screen-free Robotic Programming Environment for Children," **20th International Conference on Ubiquitous Robots (UR)**, USA, 710–715, 2023, doi: 10.1109/UR57808.2023.10202226.

[11] S. Tobias, J. D. Fletcher, and A. P. Wind, "Game-based learning," *Handb. Res. Educ. Commun. Technol.*, pp. 485–503, 2014.

[12] G. Lampropoulos, E. Keramopoulos, K. Diamantaras, G. Evangelidis, "Augmented reality and gamification in education: A systematic literature review of research, applications, and empirical studies," *Appl. Sci.*, 12(13), 6809, 2022.

[13] A. Gardeli, S. Vosinakis, "The effect of tangible augmented reality interfaces on teaching computational thinking: A preliminary study," **21st International Conference on Interactive Collaborative Learning (ICL2018)**, Greece, 673-684, 2020.

[14] A. Theodoropoulos, G. Lepouras, "Augmented Reality and programming education: A systematic review," *Int. J. Child-Computer Interact.*, 30, 100335, 2021.

[15] S. A. Hassan, T. Rahim, S. Y. Shin, "ChildAR: an augmented reality-based interactive game for assisting children in their education," *Univers. Access Inf. Soc.*, 21(2), 545–556, 2022.

[16] Y.-C. Chien, Y.-N. Su, T.-T. Wu, Y.-M. Huang, "Enhancing students' botanical learning by using augmented reality," *Univers. Access Inf. Soc.*, 18, 231–241, 2019.

[17] Z. Çipiloğlu Yıldız, M. Türker, R. Ak, "Mimari Miras Eğitiminde Artırılmış Gerçeklik ve Fotogrametri Desteği," *Bilişim Teknol. Derg.*, 14(2), 137–149, 2021, doi: 10.17671/gazibtd.792539.

[18] B. Cleto, C. Sylla, L. Ferreira, J. M. Moura, "CodeCubes: Coding with Augmented Reality," **First international computer programming education conference,** Portugal, 7:1-7:9, 2020, doi: 10.4230/OASIcs.ICPEC.2020.7.

[19] S. Washbrooke, N. Giacaman, "Play, Code, Learn: Fostering Computational Thinking in Primary Aged Learners Through Interactive Play," **IoT, AI, and ICT for Educational Applications: Technologies to Enable Education for All**, Ed.: S. Papadakis, Cham: Springer Nature, Switzerland, 135–162, 2024.

[20] M. G. Rios, M. Paredes-Velasco, "Using Augmented Reality in programming learning: A systematic mapping study," IEEE Global Engineering Education Conference (EDUCON), Austria, 1635–1641, 2021, doi: 10.1109/EDUCON46332.2021.9454149.

[21] M. Liang, Y. Li, T. Weber, H. Hussmann, "Tangible interaction for children's creative learning: A review," Conference on Creativity and Cognition, 1–14, 2021.

[22] J. M. Cerqueira, B. Cleto, J. M. Moura, C. Sylla, L. Ferreira, "Potentiating Learning Through Augmented Reality and Serious Games," Springer Handbook of Augmented Reality, Eds.: A. Y. C. Nee, S. K. Ong, Cham: Springer International Publishing, 369–390, 2023.

[23] I. Radu, B. MacIntyre, "Augmented-reality scratch: a tangible programming environment for children," Conference on Interaction Design for Children, Italy, 2009.

[24] Q. Jin, D. Wang, X. Deng, N. Zheng, S. Chiu, "AR-maze: A tangible programming tool for children based on AR technology," ACM Conference on Interaction Design and Children, Norway, 611–616, 2018, doi: 10.1145/3202185.3210784.

[25] A. Fuste, C. Schmandt, "Hypercubes: A playful introduction to computational thinking in augmented reality," CHI PLAY 2019 - Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play, Spain, 379–387, 2019, doi: 10.1145/3341215.3356264.

[26] A. Gardeli, S. Vosinakis, "ARQuest: A tangible augmented reality approach to developing computational thinking skills," International Conference on Virtual Worlds and Games for Serious Applications (VS-Games), Austria, 1-8, 2019, doi: 10.1109/VS-Games.2019.8864603.

[27] S. Goyal, R. S. Vijay, C. Monga, P. Kalita, "Code bits: an inexpensive tangible computational thinking toolkit for K-12 curriculum," International Conference on Tangible, Embedded, and Embodied Interaction, Netherlands, 441–447, 2016.

[28] A. Sabuncuoğlu, M. Erkaya, O. T. Buruk, T. Göksun, "Code notes: Designing a low-cost tangible coding tool for/with children," ACM Conf. Interact. Des. Child., Norway, 644–649, 2018, doi: 10.1145/3202185.3210791.

[29] A. Sabuncuoglu, T. M. Sezgin, "Kart-ON: An Extensible Paper Programming Strategy for Affordable Early Programming Education," ACM Human-Computer Interact., 6(EICS), 1–18, 2022.

[30] T. Bell, J. Alexander, I. Freeman, M. Grimley, "Computer science unplugged: School students doing real computing without computers," New Zeal. J. Appl. Comput. Inf. Technol., 13(1), 20–29, 2009.

[31] A. Juškevičiene, G. Stupuriene, T. Jevsikova, "Computational thinking development through physical computing activities in STEAM education," Comput. Appl. Eng. Educ., 29(1), 175–190, 2021.

[32] J. Nielsen, "Enhancing the explanatory power of usability heuristics," SIGCHI conference on Human Factors in Computing Systems, USA, 152–158, 1994.