Building, Controlling and Simulating We-Do Robots

Ahmed Wael[1], Ghada Ahmed[1] and Nada Sharaf[2]
[1]*The German University in Cairo, Egypt*
[2]*The German International University, Egypt*

Keywords: WeDo, Robotics, Simulator, Programming, Children, Computational Thinking.

Abstract: Robotics is now integrated in the STEM curricula for teaching mathematical concepts, mechanics, physics and programming skills. However, the presence of the robots and hardware components is always a challenge. It could be costly and in case of online learning, it is hard to use hardware elements. This is why there is an increasing need of simulators that display the robot in different environments. Such simulators could provide feasible solutions. The work presented in the paper introduces “Let’s LEGO”, which is a simulator for building and constructing robots, controlling them programmatically and seeing them in action based on the Lego WeDo 2.0 electronic kit. The work presented in this paper builds on the current need of introducing computational thinking to children from an early age. The goal of this project is to investigate methods for developing a Simulator for creating and constructing robots, controlling them programmatically, and viewing them in operation. Let’s LEGO was evaluated by instructors responsible for teaching programming and robotics to children in grades 2-4 (7+ years old). They reported that they thoroughly enjoyed the experience and would like to include it in their teaching techniques. The aim is to strengthen how technology-mediated learning may be used to provide a better and a more enjoyable learning experience to students with different profiles.

1 INTRODUCTION

Around the world, educational reform is taking place, and one of the foundations of the reform is the introduction and integration of assistive technology methods into educational institutions. Motivated by the possibility of higher economic, social, educational, and technical advantages, both emerging and developed countries are implementing this reform, with a strong emphasis on technology integration in education. This research is part of a cluster aiming at examining how developing nations may absorb, adapt, and utilize the information obtained by countries that have already jumped on the bandwagon of technology assistive methods’ integration in their own educational systems.

Nowadays, we can witness the heavy use of technological devices by children either for educational or entertainment purposes. It is inevitable to have them play games, chat with their friends, have online meeting or solve their homework using their computers and smart phones. However, learning the true power of those smart devices should be the desired goal for our children. They should be the real producers rather than being consumers.

”A computer is a bicycle for your mind” -Steve jobs. In the previous quote, Steve jobs referred to the use of computers to amplify the power of human brains. This can be attained by learning how computers are programmed. Programming teaches children how to think differently and effectively. A programmer needs to use logical thinking to be able to divide a large problem and break it down into smaller conquerable pieces in order to solve it effectively. This divide and conquer process is called decomposition and is one of the key features of computational thinking. Computational Thinking (CT) is a collection of cognitive abilities, habits, and methods that are necessary for problem solving with a computer and should be increasingly prevalent in education (Barr and Stephenson, 2011).

Enhancing the CT for children include some key skills such as decomposition, pattern recognition, and algorithmic design (Selby and Woollard, 2013). Lego models can be a great way to teach children decomposition, by showing them the final model and asking them to figure out the steps needed. Lego models as well can be of great use to teach students about pattern recognition. They can use their building skills to explore the repeated patterns for similar parts of models.

Wael, A., Ahmed, G. and Sharaf, N.
Building, Controlling and Simulating We-Do Robots.
DOI: 10.5220/0011087300003182
In *Proceedings of the 14th International Conference on Computer Supported Education (CSEDU 2022) - Volume 1*, pages 376-382
ISBN: 978-989-758-562-3; ISSN: 2184-5026

Pattern recognition is always an important skill for teaching the looping concept in programming. Following the instruction manual for building a Lego model as well gives an example for student on algorithmic design. Thus, we can deduce how building Lego models and programming them can contribute in teaching students main aspects of Computational Thinking in an enjoyable way. However, those kits can be expensive and difficult to attain for personal use (Cardoso et al., 2018). Moreover, with the current Covid-19 situation, some parents prefer online learning which can be difficult when hardware (in our case programmable Lego kits) are included. All of these reasons were the main inspirations for Let's LEGO. The aim of this project is to investigate the feasibility and applicability of creating a simulator for building and constructing robots, controlling them programmatically, and observing them in different environments. The work aims at building a simulator for the well-known Lego Education WeDo 2.0 Robotics construction set, aimed at children aged 7-9. The reason behind that was the lack of presence of any (free) simulators for this specific version of the Lego Educational Kits, to the best of our knowledge. The WeDo set is simple yet versatile, making it a great candidate to teach children robotics and programming (Usengül and Bahçeci, 2020).

The paper is organized as follows: Section 2 introduces some related work. Section 3 shows more details about let's LEGO. The experimental design is introduced in Section 4. We finally conclude with directions to future work.

2 RELATED WORK

This section discusses virtual prototyping for the latest Lego Mindstorms robot, the EV3. Lego Mindstorms is a platform that allows users to operate a programmable and customizable 'brick.' This brick serves as the control center and power station for the robot and is compatible with a variety of extra connectable modules for a variety of purposes, such as LEDs, motors, colour sensors, and buttons. These sensors and actuators can be controlled by creating code and uploading it to the brick. As of 2013, the Mindstorms EV3 is the third and most recent addition to the Mindstorms family, and its software includes an upgraded brick that runs Linux locally (Cornelissen, 2019).

The Mindstorms series has proven to be successful in education, where it is used to teach programming and computer science (Klassner and Anderson, 2003). However, there are some drawbacks to using this type of instruction. Mindstorms robots are expensive, making it difficult to provide each student with their own robot. The frequent swapping of the robot between students limits the amount of time available for each student to actively work with it. Furthermore, working with Mindstorms might be inefficient owing to the substantial effort required to upload code to the brick on a regular basis. Given the popularity of the Mindstorms series in education, as well as its short comings, simulation software for the EV3 series might be beneficial. As the fundamental idea behind these robots is real-life interaction (driving around, flashing LEDs, handling physical input, etc...), the simulation component would be distinct from traditional virtual prototyping in that it would need a completely visual depiction of the brick and its linked components (Cornelissen, 2019).

A real-world EV3 brick interfaces with its hardware to set and retrieve variables based on software execution (Cornelissen, 2019).

In the simulation, this concept must be reproduced. When programming languages are incompatible, a dilemma arises: the simulation software is created in Unity using C#, and Python is utilized to operate the simulated robot. One solution is to include an interpreter within the simulation program, allowing the simulated brick to run suitable code natively. A more achievable approach is to develop a mechanism to communicate between the simulation software and a user-written application running on the same computer (Cornelissen, 2019).

The idea of the work presented in the paper is to provide an easy to use simulator for lego WeDo. WeDo has proved to help in teaching children about robotics and programming (Chalmers, 2018). It targets an age group that is younger than EV3. Since the aim is to introduce programming and computational thinking as early as possible, this was the motivation to fill int he gap and provide a free simulator that can allow children to learn about WeDo without having to own a kit.

There is a difference between 'input' and 'output' values. The Python application assigns input values, while the simulation retrieves result values. A LED module's 'Colour' value counts as input, while an ultrasonic sensor's 'Distance' value counts as output. The objective is to achieve structural equivalence between the simulation (in C#) and ev3dev2 in terms of hardware modelling. In this manner, despite the language barrier, a connection between the two processes may be established. When this model is implemented on the simulator side, it generates a 'virtual' brick that represents the current state of the simulated hardware. The simulation may then be performed using these variables. There is no locally saved representation of the brick in Python. To retrieve and set values, the re-implemented ev3dev2 methods interact directly with the simulation. For Communication between the two processes, a generic framework must be built. In this connection, the Python software that is being executed (which is effectively replacing the EV3 brick operating system) functions as the client, while the emulated hardware acts as the server. The link must first be established. This is accomplished by a simple handshake in which the client provides its name and the simulator responds with its own. The communication can then begin (Cornelissen, 2019).

3 Let's LEGO

This section provides a thorough overview of how to build, operate, and simulate Lego robots virtually. Unlike past researchers, the proposed approaches involved physically constructing robots and replicating them through bluetooth or cable. As a result, by virtualizing it, many of the disadvantages of these approaches may be eliminated.

3.1 Let's LEGO Engine

Let's LEGO simulator is implemented on top of a game engine. Unreal engine, CryEngine, and Unity 3D were among the possible game engines. Unity[1] is a well-documented game engine featuring 2D and 3D functionality, as well as a multi-platform editor (Cornelissen, 2019). This editor includes all the tools needed to create games and other interactive applications, such as physics and collision systems. In conjunction with Unity, most developers code in C# or JavaScript. Unity applications may be exported to various platforms with similar functionality, including Windows, macOS, Linux, Android, iPhone, and many others. Unity employs what are known as GameObjects. Different components have been added to these GameObjects. Components hold data about parameters like position, rotation, physics, 3D representation, and much more. Scripts may also be added to a GameObject, allowing developers to control elements of that GameObject and other variables through code (Cornelissen, 2019). With all of these capabilities, Unity is the clear choice for implementing the Lego robot simulator.

[1]https://unity.com/

3.2 Flow of Let's LEGO

In this Unity simulator, four scenes were developed, each with a distinct purpose. Starting with the start menu screen, moving on to building the Lego robot, then a scene to control this built robot using code blocks, and finally the simulation of the robot controlled by the code blocks added in the previous scene, which can be re-coded by returning to the control scene, as shown in Figure 1. Throughout this paper, each scene will be discussed in further detail, along with scripts written in C# and integrated with them.

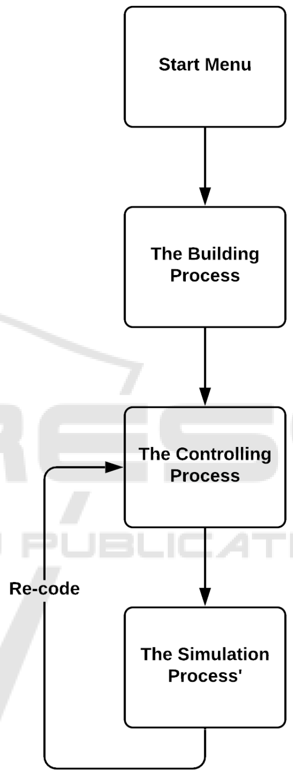


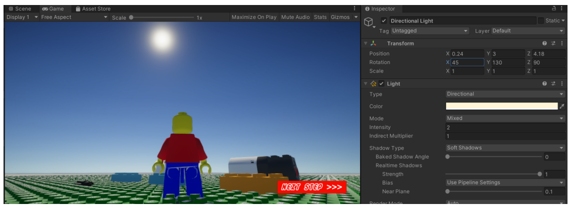
Figure 1: Flow of the Simulator.

3.3 Building The Robot Scene

The key purpose of Let's Lego simulator is to make the user interested and engaged as if they were actually creating a robot with a WeDo Construction Set. A detailed review of creating the game environment, the player, and the bricks with their associated scripts is provided in this section.

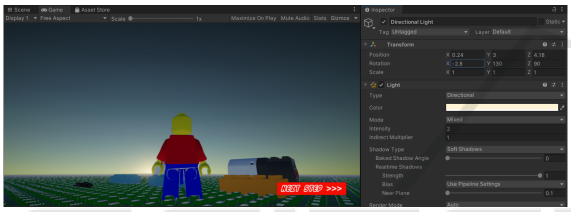
3.4 Environment

The game's environment has a directional light; which is a sun-like object that gives light to the whole area dependent on its rotation angle, which can be seen in Figure 2 and Figure 3. The ground is a plane with a mesh collider linked to it to keep items above it from falling. The scene's main camera is set to be attached to the player, giving the user a First-person or Third-person perspective by zooming in or zooming out. This camera adjusts its direction and angle dependent on the mouse movement in this 3D environment, bringing it closer to the real-life, as seen in Figure 4.



```
Inspector
Directional Light                          Static
Tag Untagged            Layer Default
Transform
Position      X 0.24    Y 3      Z 4.18
Rotation      X 45      Y 130    Z 90
Scale         X 1       Y 1      Z 1
Light
Type                    Directional
Color
Mode                    Mixed
Intensity               2
Indirect Multiplier     1
Shadow Type             Soft Shadows
  Baked Shadow Angle    0
  Realtime Shadows
    Strength            1
    Bias                Use Pipeline Settings
    Near Plane          0.1
Render Mode             Auto
```

Figure 2: Directional Light with angle 45° (Daylight Mode).



```
Inspector
Directional Light                          Static
Tag Untagged            Layer Default
Transform
Position      X 0.24    Y 3      Z 4.18
Rotation      X -2.8    Y 130    Z 90
Scale         X 1       Y 1      Z 1
Light
Type                    Directional
Color
Mode                    Mixed
Intensity               2
Indirect Multiplier     1
Shadow Type             Soft Shadows
  Baked Shadow Angle    0
  Realtime Shadows
    Strength            1
    Bias                Use Pipeline Settings
    Near Plane          0.1
Render Mode             Auto
```

Figure 3: Directional Light with angle -2.8° (Night Mode).

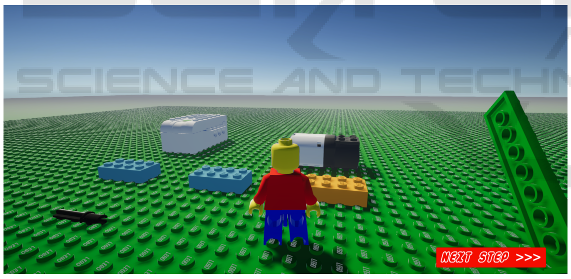


Figure 4: 3^{rd} Person View Camera.

3.5 Player

This scene contains a comprehensive examination of the player's animation, movement, and use in the construction of the robot. The player was designed to offer the user a sense of realism. To begin, the player is a 3D model taken from a GitHub MIT-licensed source (cit, a), which is made up of the head, hands, body, chest, and legs as shown in Figure 5. Handles were introduced to the player at each joint so that they could be utilized in the animation phase when the Animator is linked to the player. The player may move and jump in two modes. The movement state has five states: idle, forward run, backward run, left and right. Each state includes a unique animation, such as tilting forward or back, and left or right.

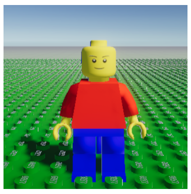


Figure 5: The Lego Player Model.

3.5.1 Lego Bricks

The Lego bricks are designed using Studio 2.0 (cit, b). Studio 2.0 is a program that includes a library of 3D Lego bricks of various sizes, forms and colors.Studio 2.0 was used to construct the 3D bricks required for each model from the Lego models that is driven by a motor.

3.6 Script

After the bricks were created, a script named Drag3D was applied to each one. Drag3D is mostly done in the update function, which is called once every frame. It has a method that is invoked when the mouse is pressed, and it places the GameObject in the same location as when the mouse is pressed. It stays connected to the mouse wherever it moves as long as it remains down till it is linked to its attachment point. The bricks are added one by one until the model is complete. When the player is done with the model building, the "Next Step" button can be clicked to move to the next scene.

3.7 Programming the Robot Scene

A thorough examination of how the blocks are programmed in order to generate the appropriate output, as well as how these code blocks are connected, moved, and copied, is provided in this section. WeDo 2.0 software is used to program an actual Lego model after it has been built by linking code blocks with a wire. Some blocks are in charge of the motor's speed, while others are in charge of the motor's direction, and yet others are in charge of the motor stopping or changing direction, and so on. These programming blocks are combined to produce a set of instructions that are executed by the connected device. The final result is displayed on a Lego model built with real Lego components rather than virtual Lego parts.

3.7.1 Code Blocks

The code blocks were adopted from the WeDo 2.0 program in order to provide the user with the familiar WeDo appearance and the same button design. The necessary buttons for the prospective code blocks, summarized in table 1, are added on the screen, as shown in Figure 6.

Table 1: Blocks Used (cit, c).

Block Name	**Description**
Motor Power	Sets the motor power to the specified level and starts the motor.
Motor On For	Starts the motor for a chosen amount of time specified in seconds.
Motor Left	Sets the motor to turn the axle clockwise and starts the motor
Motor Right	Sets the motor to turn the axle counter-clockwise and starts the motor.
Motor Off	Stops any movement of the motor.
Play Sound	Plays a sound. The sound is chosen from a list available within the software.
Wait For	Use this block to tell the program to wait for something to happen.
Start	When used, always placed at the beginning of a program string.

3.8 Scripts

Similar to Drag3D, a script named DragAndDrop was made for the same object, and it is to be able to drag the block using the mouse to the attachment point that is added to each block in this scene when it is the same as the block location. However, an extra item is included in this script, and it is to instantiate a new block of the same type if it is attached and above the line -2 where the set of programming blocks are added side by side.

The goal of this script, called ”CodeBlocksArray,” is to create a sorted array with all the blocks arranged from left to right. If the following set was connected in the following order, Start - Motor Speed 8 - Wait 5 - Motor Speed 4 - Motor Left, the aim is to do these movements in the same order each time. However, because the array is first generated by searching for objects with a specific tag, ”mygameobjects,” they are added to an array in the order retrieved rather than the order required. That is why array sorting is critical, since it uses the X position of the GameObject (Block) from left to right to provide the correct result. In order to transfer variables and data across scenes, the sorted array is then transferred to another array in another script named ”GameMaster” that is not destroyed when the next scene is loaded. The sorted array has now been sent and is ready to be utilized in the final simulation scene.

3.9 Simulation Scene

The last scene of the simulator will be reviewed in this section. The final scene for The Cooling Fan model is about Max and Mia, two friends who are feeling overheated due to the hot weather and need some air from the cooling fan, as seen in Figure 7. The code blocks array is moved here, since the GameMaster script is not destroyed when the scene is loaded. A script named Simulator is created, and when run, it transfers the components’ names from the GameMaster array into a new array called ”steps.” It has two flags, one for each blade of the Cooling Fan, which are used to control whether the blades rotate or not. Each item in ”steps” is tested to determine what should be done; for example, if the item is Motor Stop, the two flags are set to false, indicating that both blades will not rotate.

4 EXPERIMENTAL DESIGN

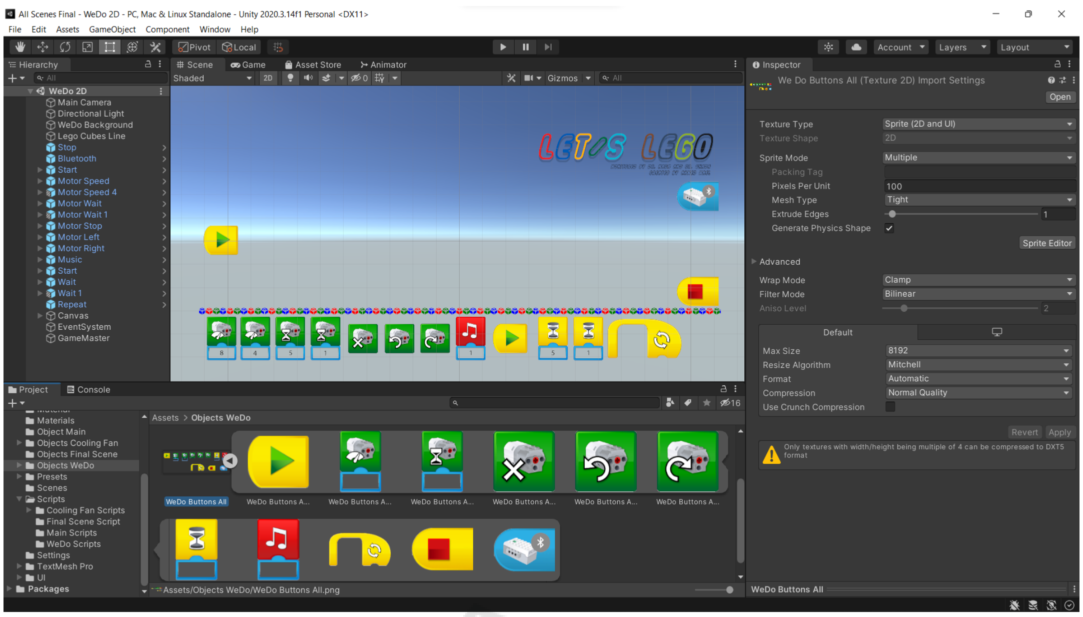
4.1 Testing Procedure

The testing determines the effectiveness/likeliness of the virtual robot simulator and if it can replace the real robot construction. A test is conducted on prospective instructors who are interested in instructing or assisting youngsters to learn programming. These prospective instructors are well aware of the software’s benefits and drawbacks, as well as the implications of utilizing a tool like this with children. The major goal of the testing is to determine if this tool can stand on its own and offer the youngster with the entire programming abilities supplied by robot building and programming.

4.1.1 The Followed Method

The System Usability Scale (SUS) is a basic ten-item likert scale that provides a global perspective of subjective usability assessments (Brooks, 1986). The standard SUS consists of the ten items (odd-numbered items worded positively; even-numbered items worded negatively) (Lewis and Sauro, 2009).
The test was held with 25 participants. 64% of the participants were males and 36% were females of an average age of 22.5. To ensure reliable results, those who completed the survey were either graduated or undergraduate engineers with a solid programming experience. The sequence was as follows:

1. In an attempt to familiarize the participant with Let’s LEGO, the user was given the opportunity to be introduced to the program and start using it.
2. SUS was applied to the program, encompassing the aforementioned survey items and adding a question to evaluate the software’s dependability, whether used alone or as an aid to the physical one.


All Scenes Final - WeDo 2D - PC, Mac & Linux Standalone - Unity 2020.3.14f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
Scene Game Asset Store Animator
Inspector
Project Console
Assets > Objects WeDo

We Do Buttons All (Texture 2D) Import Settings

Texture Type	Sprite (2D and UI)
Texture Shape	2D
Sprite Mode	Multiple
Pixels Per Unit	100
Mesh Type	Tight
Extrude Edges	1
Generate Physics Shape	✓
Wrap Mode	Clamp
Filter Mode	Bilinear
Aniso Level	2
Max Size	8192
Resize Algorithm	Mitchell
Format	Automatic
Compression	Normal Quality

Only textures with width/height being multiple of 4 can be compressed to DXT5 format

Assets/Objects WeDo/WeDo Buttons All.png

Figure 6: Let's Lego Code Blocks.

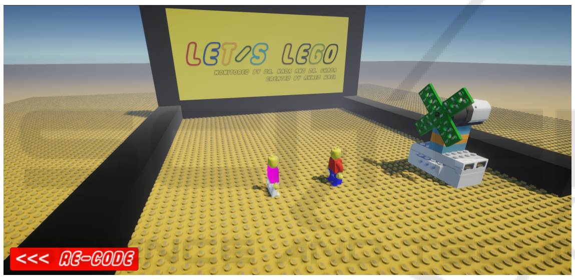


Figure 7: Max And Mia With The Cooling Fan.

4.2 Results

Figure 8 displays the average value for each question. For odd-numbered questions, the higher values are better, and vice versa for even-numbered questions. The SUS formula was used to obtain the maximum, minimum, and median values as shown in Table 2.

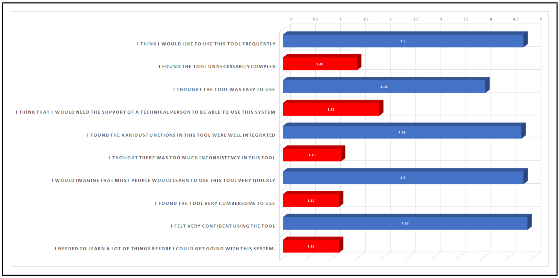


Figure 8: SUS Average Results.

Finally, after exploring all the functionalities Let's LEGO software, the participants were asked about the possibility of relying solely on the software to provide students with a sufficient understanding of building, controlling, and simulating robots. 64% of the participants chose "yes" as an answer, and the rest 36% were not that sure, hence choosing "no".

Table 2: SUS Result.

Maximum Value	100
Median Value	92.5
Minimum Value	77.5

5 CONCLUSION & FUTURE WORK

Educational robots proved to help students in different STEM disciplines. They strengthen students' creativity, programming skills, problem solving skills and much more through the process of designing models, assembly and coding their functionalities. However, having those robots in classrooms with the sufficient amount can be expensive and impractical sometimes, especially with the new urge of online learning in the COVID-19 situation. Let's Lego aimed at establishing a simulator for creating, constructing robots, controlling them programmatically, and viewing them operating virtually in different scenarios and environments. The process of building the simulator, starting from the design to the implementation is all discussed in detail throughout the paper. The simulator was based on the well known WeDo construction kit, which has been widely used in robotics and programming classes for children starting 5 years old. Let's Lego was tested with computer science instructors who are interested in teaching children. Based on the preliminary results of the testing, the software can be sufficiently dependable in teaching robotics classes for children and provides the learner with a sufficient grasp of building and programming a robot.

Looking forward, this research provides a good starting point for different possibilities to extend the work. First of all, the simulators need to provide all the building models found in the WeDo software, so that the student has a wide range of robots to construct. Another aspect that can be added is an open environment to design and assemble any robot with different ranges of functionalities e.g. moving, lights and sounds. Furthermore, adding different environments for experimenting with the robot would be a good idea. For example, after building a car, the student can virtually try how it moves in the desert or a jungle and experiment different mechanics and physics rules. Finally, long-term experiments with students and teachers in classrooms are needed to test the system thoroughly.

REFERENCES

"legogame (mit licensed)". Available online: https://github.com/ditzel/LegoGame/blob/master/LEGO/Assets/Object/Character/figure.fbx. (accessed on May 2021).

"studio 2.0". Available online: https://www.bricklink.com/v2/build/studio.page. (accessed on May 2021).

"wedo user guide". Available online: https://le-www-live-s.legocdn.com/sc/media/files/user-guides/wedo/wedo-user-guide-80b6e879549d1be595355dc8b6dee075.pdf?la=en-us. (accessed on May 2021).

Barr, V. and Stephenson, C. (2011). Bringing computational thinking to k-12: What is involved and what is the role of the computer science education community? *Acm Inroads*, 2(1):48–54.

Brooks, J. (1986). Smart phone applications for people with brain injury. *United Kingdom: Agency for Clinical Innovation*.

Cardoso, A., Ferreira, H. S., and Sousa, A. J. (2018). Programming for young children using tangible tiles and camera-enabled handheld devices. In *11th annual International Conference of Education, Research and Innovation.*

Chalmers, C. (2018). Robotics and computational thinking in primary school. *Int. J. Child Comput. Interact.*, 17:93–100.

Cornelissen, L. (2019). Simulating lego mindstorms ev3 robots using unity and python. *Radboud University Nijmegen*.

Klassner, F. and Anderson, S. D. (2003). Lego mindstorms: Not just for k-12 anymore. *IEEE robotics & automation magazine*, 10(2):12–18.

Lewis, J. R. and Sauro, J. (2009). The factor structure of the system usability scale. In *International conference on human centered design*, pages 94–103. Springer.

Selby, C. and Woollard, J. (2013). Computational thinking: the developing definition. Project report.

Usengül, L. and Bahçeci, F. (2020). The effect of lego wedo 2.0 education on academic achievement and attitudes and computational thinking skills of learners toward science. *World Journal of Education*, 10(4):83–93.