Applying Augmented Reality to Learn Basic Concepts of Programming in U-Learning Environment

Denis Acosta, Margarita Álvarez(✉), and Elena Durán

Research Institute in Informatic and Information System, Faculty of Exact Science and Technology, National University of Santiago del Estero, Santiago del Estero, Argentina
{alvarez,eduran}@unse.edu.ar

Abstract. Learning the basic concept of programming, such as control structures, is considered difficult due to their complexity and require a high level of students' abstraction. Faced with this problem and taking advantage of ubiquitous learning that allows learning without being limited to a specific space or time, developing a ubiquitous learning app that helps students learn these concepts more tangible for them has been considered challenging. For this reason, we developed a software application (App) that, through augmented reality techniques, shows an object to the student, who must move in search of it. As a result, the App generates a program with the actions carried out. This article presents the ubiquitous learning App and the principal modules' development: the compilation module and the 3D manager module. The first one allows translating the student's cell phone's geographical coordinates into a programming language's instructions. The second one allows the student to visualize the objects using augmented reality. The tests carried out on both modules are also shown, demonstrating the feasibility of translating the actions and displaying the objects.

Keywords: Ubiquitous learning · Augmented reality · Control structure · Geo-localization

1 Introduction

By ubiquitous learning (UL) can be understood, one that is not limited to a specific context or is conditioned by a particular space, but the acquisition of new knowledge can occur in any situation in which the subject is, regardless of the moment and place [1]. The UL allows students to access all kinds of information through interaction with the objects surrounding it, whether physical or virtual [2]. Encourage collaboration, connect formal, non-formal, and informal spaces; and, relocates the location of learning, both inside and outside the classroom, and adapts the contents, presentations, and activities to the students' characteristics and context.

On the other hand, augmented reality (AR) is the technology that improves and enriches users' perception and interaction with the physical world by complementing it with virtual 3D objects that seem to coexist in the physical world [3]. This technology is generally used with ubiquitous and mobile technologies and has found a vibrant application in education. Thus, the combination of mobile and portable devices, AR and UL, provide immersive learning, enriched, situated, and fluid learning experiences [4].

The learning activities associated with programming are recognized that present a high degree of difficulty. Several studies have determined that the causes that generate this problem are related to specific characteristics that occur in the classroom and with particular cognitive skills relevant to learn the fundamentals of programming. Among them are abstraction capacity, an excellent logical-mathematical aptitude, and the facility for solving algorithmic problems [5].

Given the problems raised and taking advantage of UA and AR provide, we considered the challenge to develop an App supporting students in learning the basic programming concepts. This App shows the student the algorithms he/she performs when he/she moves in a particular environment, using the microlearning approach, which proposes learning with relatively short efforts and takes little time [6]. Thus, the App shows an object that the student must obtain using AR techniques. The student walks to the thing, and as a result, the App generates a program with the actions carried out. In [7], we present the App's architecture, which provides a compilation module. It takes inputs from the student's cell phone sensors, and then they are manipulated and processed by the sensor module. It translates them into a program written in the language designed to such an end. This last represented a significant challenge, as there is currently no tool to make this translation possible. In [7], we also present the development and validation of the compilation module. The App also consists of a 3D manager module that allows inserting and manipulating objects with AR, a user interface module, and the main module. In this paper, we present in detail these modules.

The interaction with the App provides the student with an immediate response, favoring the student experimentation in the real world and quickly forming a mental model thanks to the tool's answers.

In the following sections, we cite antecedents of related works, present the architecture, and design and develop the App modules. Finally, we have shown the tests carried out and expressed some conclusions regarding the work carried out and future actions.

2 The Background

This section presents the background of software applications for teaching the basics of AR-based programming; and some antecedents of works related to the learning of informatics and computational issues through UL.

In [8], the authors present a flexible U-learning mobile application for students to access the material. The authors adopted computational thinking (PC) to help students develop practical computing skills. They chose three classes of first-year students for the empirical study. They were divided into three groups: two experimental groups (UL&PC group and PC group) and a control group. Based on this study results, students who received UL treatment might have significantly better computer skills using PowerPoint and Word than those who did not. However, PC's treatment did not result in better development of the students' computer skills in this investigation.

In [9], a customized UL support system based on multiple information sources is proposed to encourage college students to take computer programming courses. It includes a new technique that integrates student learning problems, knowledge level, and learning style to personalize learning paths that offer students their programming concepts.

In [10], the requirements necessary to support UL with existing software tools for teaching and learning computer programming to children between 4 and 10 years of age are identified. Twenty-two tools were analyzed and contrasted with UL's five known characteristics: permanence, accessibility, immediacy, interactivity, and awareness of the context. On a final note, they recommend adding UL features in kids' programming tools.

Boonbrahm et al. [11] developed a tool to learn the primary flow of commands and control structures, including sequence, selection, and iteration structures. Using the tool, students construct a program flow diagram using AR markers. The developed software captures the image of the flow chart that the student has built, processes the program, and shows the result of the command's execution. The tool identifies the command, the variable value, and the Boolean operator. Then, the software simulates each command's result, and thus the student can check if the program's logic is correct.

In [12], educational material has been developed using AR to teach basic programming concepts. The theme developed was Control structures, and AR exploration activities were carried out using previously printed markers. It was also evaluated by the teachers of the first-year chairs related to the teaching of programming.

In work presented by Tan and Lee [13], they argue that using AR in teaching basic programming concepts is very appropriate. The article analyzes a survey of students enrolled in computer programs at Sunway University in 2016. The results indicate that students have moderate participation when traditional methods are used to teach programming concepts. Moreover, the results were determined, too, that 80% of the students agreed that the AR learning method is useful because it provides more fun, interest, and a basic understanding for students to learn to program.

Da-Ren Chen et al. [14] study show the use of AR technology to create virtual objects for mobile devices. This study provides students with contextual information related to the outdoor learning environment. The ubiquitous system developed is for the tourism area. Context-awareness was used to allow students to follow learning activities along a predetermined path, using AR features. The goal is to provide students with a friendly, interactive interface and engaging means to stimulate intrinsic motivation and learning performance.

In the background review carried out, we did not find works presenting UL experiences with AR in programming concepts or software applications for this purpose.

On the other hand, although there are applications that translate geographic coordinates, no antecedents have been found for converting such units to a programming language such as the one proposed in this work.

3 U-Learning App

3.1 App Description

The purpose of the App is to support the teacher in introducing programming concepts related to basic control structures: sequential, selection, and iteration, and to allow the student to transition to the development of more complex algorithms.

For the design of the App, the microlearning approach has been considered [6]. To this end, we divided into three levels the teaching of basic control structures in such a way as to introduce a basic control structure at each level, increasing the level of complexity of the exercises that the student can perform. In level 1, simple activities are carried out to learn the concept of instruction and instruction sequence. In level 2, exercises are included where the conditional structure gives rise to instructions to turn right and left to avoid obstacles. At level 3, the activities introduce the concept of repetition [7].

3.2 App Architecture

The App's objective is for the student to walk to where the virtual object is located. The App returns to the student a program formed by a set of atomic instructions with steps that he carried out until he found the object. To achieve this goal, we design the App architecture consisting of five main modules. In Fig. 1, the modules and the interrelationships between them are shown.

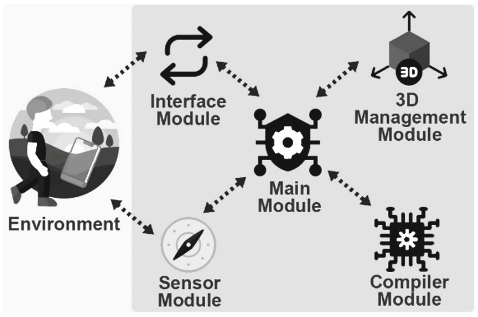


Fig. 1. App architecture

The **interface module** takes care of the student's communication with the App. It determines the complexity level to offer the learning activities according to it and finally shows the student the resulting program using representative symbols.

The **sensor module** is responsible for sensing the environment of the device. With the help of GPS, it can obtain geo-referential data (latitude and longitude). On the other hand, the gyroscope and accelerometer work together to carry out the tasks of anchoring the object found in the real environment (calculating the X, Y, and Z axes) and relocating it in case of moving (left, right, up, or down) in the environment (calculating the axis of rotation).

The **main module**: When starting the App, this module downloads all the data necessary for the App's operation: object database, level register, and initial settings. It is also in charge of coordinating all the tasks with the other modules and facilitating the interaction between them, providing support for the App's operation. As the main task, it sends all the data collected throughout the activity to the compiler to translate it and gives the algorithm (program) obtained on the way to reaching the object as a result.

The **3D management module**, with the help of the ARCore tool, renders the object to be viewed and adapts it based on the graphic capabilities of the user's device, providing a three-dimensional model of the object ready to be processed on the screen.

The **compilation module** contains the three parsers' tasks (lexical, syntactic, and semantic) to translate geographic locations. These locations are the product of collecting information that the main program performs from the student's actions. These actions may or may not be optimized based on the complexity of the task accomplished, granting fully customized algorithms.

3.3 Sequence Diagram of the App

The user's interaction, the mobile device, the App's modules, and sensors are model in the sequence diagram (Fig. 2).

When the user starts the App, the GPS determines which is the user's current location. Based on this data, the system can determine if a geo-referenced object is close to its location displayed on the map during the activity. In proximity to it, the user can select it and start sensing the context to obtain information from the environment and anchor the object in the real world. Previously, the App performs the task of rendering the object's three-dimensional model. The data obtained from the environment mapping

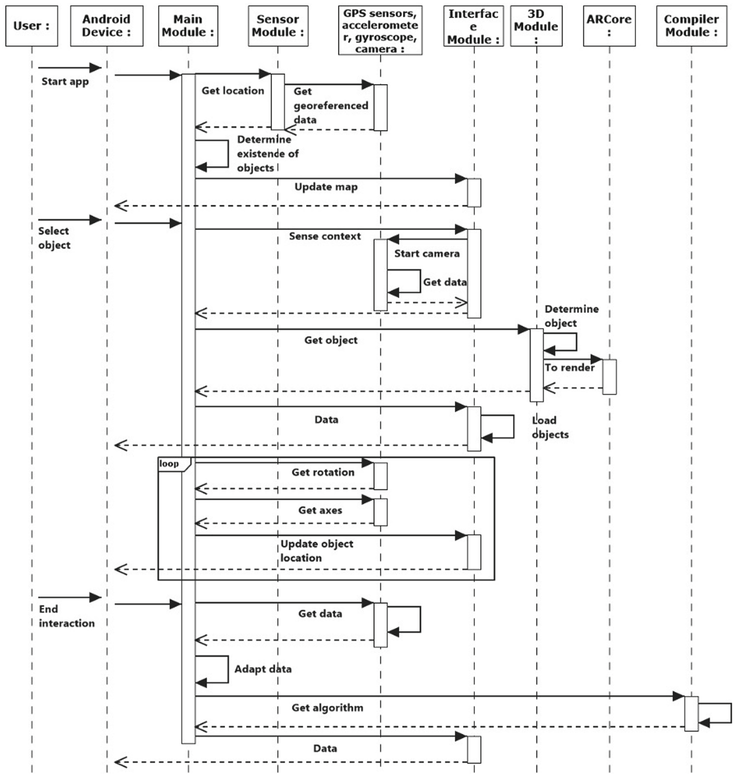


Fig. 2. Sequence diagram of the App

(axis values) can be anchored and updated in movement events. Finally, when the user ends the activity, all the data obtained throughout the experience is collected and adapted to the compiler to return the representative algorithm to the activity carried out.

4 Design and Development of the Modules

4.1 Compilation Module

The methodology followed for the compiler's design and development consisted of the following steps: Definition of the source and object languages, Design and construction of the analysis phases: lexical analysis, syntactic analysis, and semantic analysis. Then, the synthesis phase was developed: the code generator.

Source Language and Object Language. Since a compiler is a program that reads a program written in a source language and translates it into an equivalent program in an object language, both languages are defined in this section.

We design source language with a simple syntax so that the rest of the modules do not require a complicated interface to communicate with the compiler. In this way, we guarantee a completely modular and independent design of the App.

The compiler's input data are each of the geographic locations (latitudes and longitudes) obtained by the different devices capable of sensing the geographic context (GPS, Accelerometer, and Gyroscope). Therefore, the source language is composed of the user level and the set of geographic points.

The program resulting from the compilation is write in the object language. The characteristics considered for selecting the object language's instructions or sentences are expressed in [7]. In this case, the object language consists of the following atomic instructions: move forward (represents a forward step made by the student on his or her walk to the object), backward (represents a backward step that the student performs to meet the objective of the exercise), turn left and turn right (they allow to deviate from the linear path), if there is an object then (represents the fork when an obstacle is presented or not) and repeat (with this instruction the student visualizes a repetitive control sentence).

Construction the Compiler Phases. The lexical components and the patterns or regular expressions that these components generate have been defined for constructing the lexical analyzer. The specified lexical components are punctuation symbols and numeric constant.

For the Syntactic Analyzer phase, context-free grammar is defined using the BNF notation, which is the input to the ANTLR generator.

```
start:
  LLAVE_A
    CORCHETE_A nivel CORCHETE_C
    CORCHETE_A puntos CORCHETE_C
  LLAVE_C;
nivel: NUMERO;
puntos: tupla (COMA tupla)*;
tupla: PARENTESIS_A latlong COMA latlong PARENTESIS_C;

latlong: NUMERO PUNTO NUMERO;
```

In the semantic analysis phase, type verification is performed on the numeric constants to determine if they are of the latitude type or the longitude type.

Object Program Generation. The procedure described in Fig. 3 is carried out for the object program's generation. It applies three algorithms: the relationship algorithm, the classification algorithm, and the ranking algorithm.

Relationship and classification algorithms have been developed to determine the relationship between two pairs of geographic points to determine whether the student has stepped forward, backward, or turned in some direction.

Each time two consecutive elements are stored (the process that the parser executes when it traverses the abstract syntax tree), a comparison is made to determine the relationship between them (vector). The relationship is classified according to its equivalent in atomic instructions.

A vector in the plane is an ordered pair of real numbers (a, b) belonging to the space R^2 [15]. In this case, "a" is considered a set of latitudes, and the component "b" is the set of longitudes. They are the input to the Relationship Algorithm, and the results indicate the different directions that a vector can have. From the vectors obtained, the classification algorithm is applied, which establishes the relationship between said vectors. The results presented after classifying a pair of vectors are divided into two categories: parallel and orthogonal.

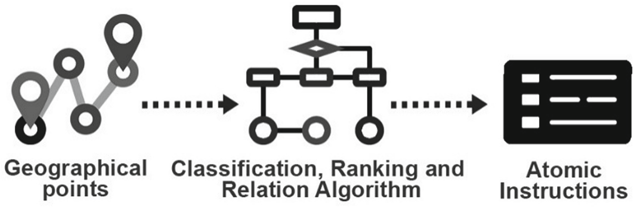


Fig. 3. The process to obtain the program

To determine when a pair of vectors belong to one set or another, we considered [15]:

The parallelism of vectors. Let u and v be non-zero vectors of R^2, the vector u is parallel to the vector v if and only if there is a non-zero scalar c such that

$$u = c.v. \tag{1}$$

Orthogonality. Let u and v be non-zero vectors of R^2; it is said that u is orthogonal to v if and only if the scalar product of u and v is equal to zero,

$$c.v = 0 \tag{2}$$

Angles between vectors. Let u and v be non-zero vectors of R^2 and let the scalar product then there exists and is unique an α [0, π] such that:

$$\cos(\alpha) = \frac{u.v}{\|u\|\|v\|} \tag{3}$$

For the classification of vectors, we apply the following rules:

- If both calculated vectors are parallel, we assumed that the user advanced in a straight line, and we consider the direction of the vectors to determine whether he advanced or retreated,
- If the vectors are perpendicular, the intersection between them is calculated to obtain the angle that joins them. If the angle is close to 90°, it is assumed to be a change of direction.
- If the upper vector has a left direction, we assume that the user has turned to the left.
- If the upper vector has the right direction, we assume that the user has turned to the right.

Finally, each of the rules is mapped with the equivalent atomic instruction (Table 1).

Table 1. Mapping relationships to atomic instructions.

Relation	Sense	Atomic instruction
Parallels	Upward	ADVANCE
	Downward	GO BACK
Orthogonal	Left	TURN LEFT
	Right	TURN RIGHT

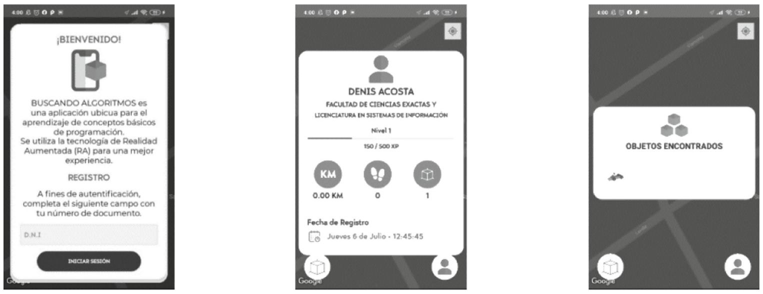
As the last stage, the compiler will adapt the set of atomic instructions based on the student's level. The ranking algorithm is applied to do this, which takes the atomic instruction set as input data and returns an adapted set based on its level as a response. This adaptation produces a reduction of instructions or addition of new ones, such as "*repeat*" and "*if there is an obstacle then*". For the conditional statement, the compiler, when detecting the tuple (0, 0), infers an obstacle. For example, if there are three points as input: (−64.25139158964157, −27.80154927681387), (0, 0), (−64.25133660435677, −27.801606219132715), the App deduces that between point 1 and point 3, there is an obstacle.

4.2 Interface User Module

This module allows communication with the student to request the App's operation's data and show him the tool's results. To begin the execution, the student must log in to the application to obtain their data (Fig. 4a).

The user interface then enables options that allow access to the user profile and object registry. The first option shows the student information about previous activities (for example, the current level of knowledge, level of experience, traveled kilometers, number of steps, number of found objects, and a record of the date on which the student started using the App (Fig. 4b)). The second option permits the student to access objects found in previous sessions (Fig. 4c).

When the exercise begins, this module displays an interactive map on the main screen. The map will update based on the user's location. When the student is close to


¡BIENVENIDO!

BUSCANDO ALGORITMOS es una aplicación ubicua para el aprendizaje de conceptos básicos de programación.
Se utiliza la tecnología de Realidad Aumentada (RA) para una mejor experiencia.

REGISTRO

A fines de autentificación, completa el siguiente campo con tu número de documento.

D.N.I

INICIAR SESIÓN

DENIS ACOSTA

FACULTAD DE CIENCIAS EXACTAS Y LICENCIATURA EN SISTEMAS DE INFORMACIÓN

Nivel 1

150 / 500 XP

0.00 KM 0 1

Fecha de Registro

Jueves 6 de Julio - 12:45:45

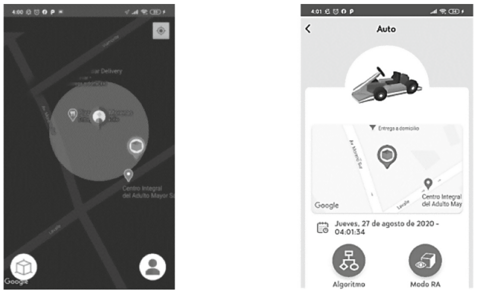
OBJETOS ENCONTRADOS

(a) Login interface (b) User profile (c) Object registration

Fig. 4. User interfaces.

the object, the interface module prompts him to sense the current context through his device's camera and identifies a surface on which to anchor and insert the object on the map. The map contains two markers. A blue one indicates the user's current position, and a red one indicates the presence of an interactive object (Fig. 5a).

The interface module shows, as a result, a screen where the image of the object, the geographical position where it was found, the resulting algorithm, and the object with AR (Fig. 5b) are displaying.



Auto

Entrega a domicilio

Centro Integral del Adulto May

Jueves, 27 de agosto de 2020 - 04:01:34

Algoritmo

Modo RA

(a) Marker map (b) Result interface

Fig. 5. Result interfaces.

4.3 3D Management Module

The 3D management module selects from the object database the object that it will show to the student. We classified the objects into three categories based on the level of knowledge. Then, the user can visualize particular objects based on the level of experience or knowledge they have.

Each object has associated an occurrence probability. The module defines a number randomly for selecting objects, and chose that one close to said value will be. Also, it performs a check to determine that the student in previous sessions has not found the object. In case this had not happened, the module will be executed again to prevent the user from encountering repeated objects.

Once the object has been downloaded and the student is within the visibility radius, the module will see a 3D object marker on the map. The user can touch it and display a dialog that will indicate that she is about to enter RA mode to project the object in context.

Finally, the 3D management module coordinates with the sensor and interface modules to determine the anchor point based on the device's rotation axes and update it in case of user movement.

5 Implementation

For the interpreter program's construction, we use the ANTLR tool (ANother Tool for Language Recognition). It is a generator of analyzers to read, process, execute or translate structured text. The regular expressions of the defined lexical components and the context-free grammar generated by the described language have been entered into ANTLR.

For the development of the 3D management module, the ARCore tool, developed by Google, was used to build AR applications. This tool allows files with extension FBX, SFB, GLT, GLTF, OBJ. It uses three technologies to integrate, through the phone camera, virtual content with the real world. They are position tracking relative to the real world, environmental understanding for detecting the size and location of flat surfaces such as the floor or a table, and light estimation, which allows the phone to estimate the current lighting conditions of the environment).

We developed the interface module with the XML language, typical of Android Studio. It is a tag language that matches each meta-tag with an internal Android class. In this way, it is possible to control what elements are inserted into the user interface and manipulate them during execution. It has the advantage that each label is adaptable to each device's screen, complying with the property of a responsive interface design.

We developed the sensor module with the Java language, also typical of Android Studio, which contains specialized libraries to access the internal components such as GPS, accelerometer, and gyroscope.

Finally, we developed the main module under the structural facade design pattern. It allows structuring several subsystems or modules, minimizing communication and dependencies between them, providing a single communication interface through the main module.

6 Tests

6.1 Compiler Module Test

We conducted tests for all levels of complexity.

Table 2 shows a test carried out for the level of complexity 1. In (Fig. 6a), we can observe the set of latitudes and longitudes captured by the App when the student makes the journey towards the object. Also, the output of the compilation module, the resulting program, and finally, the interface presented to the student is shown.

Table 3 shows a test carried out for the level of complexity 2. The student may encounter an obstacle on his path (Fig. 6b), so the resulting program introduces a conditional structure concept.

Table 4 shows a test carried out for the level of complexity 3. When the student performs the same operation several times, for example, moving forward (Fig. 6c), the resulting program contains the sentence repeat, thus introducing the concept of the repetitive control structure.

Table 2. Test performed for complexity level 1.

Input	Output	Interface

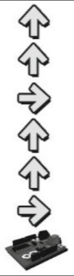
{
[1]
[(-64.25157129764557,-27.801613336920475),
(-64.25148211419582,-27.80157715482788),
(-64.25138555467129,-27.8015510562618),
(-64.25134532153606,-27.801586645213977),
(-64.25128899514675,-27.80155639460536),
(-64.25122663378716,-27.80151843304538),
(-64.25117500126362,- 27.80154868366457)]
}
	{
	[ADVANCE,
	ADVANCE,
	TURN RIGHT,
	ADVANCE,
	ADVANCE,
	TURN RIGHT]
	}
		

Table 3. Test performed for complexity level 2.

Input	Output	Interface

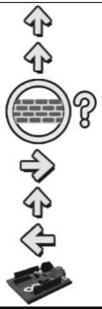
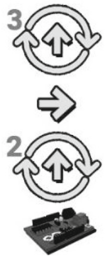
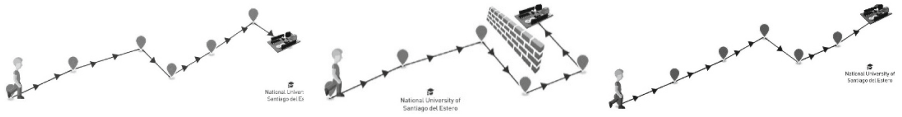
{
[2]
[(-64.25157129764557,-27.80161570951628),
(-64.25148278474808,-27.801581306871892),
(-64.25139158964157,-27.80154927681387),
(0,0),
(-64.25133660435677,-27.801606219132715),
(-64.25126284360884,-27.801596728748322),
(-64.2511984705925,-27.801581306871892)]
}
	{
	[ADVANCE,
	ADVANCE,
	IF THERE IS OBJECT
	THEN,
	TURN RIGHT,
	ADVANCE,
	TURN LEFT]
	}
		

Table 4. Test performed for complexity level 3.

Input	Output	Interface

{
[3]
[(-64.25157062709332,-27.801612150622535),
(-64.2514868080616,-27.801581306871892),
(-64.25141505897044,-27.801553428858966),
(-64.25135537981987,-27.8015231782411),
(-64.25130508840084,-27.801555801456097),
(-64.25124809145927,-27.801533854930724),
(-64.25118573009968,-27.801497079662074)]
}
	{
	[REPEAT 3,
	ADVANCE,
	TURN RIGHT,
	REPEAT 2,
	ADVANCE]
	}
		



(a) Walkthrough for level 1. (b) Walkthrough for level 2. (c) Walkthrough for level 3.

Fig. 6. Student journey of the tests performed.

6.2 3D Management Module Tests

We tested the 3D Management Module for portability, and visual distinction [16] in different environments, with have varying lighting levels, various objects, and different platforms and devices.

1. **Portability Test:** since the users of the App are the students, and they use a wide variety of devices, then the 3D management module was tested with different operating systems, versions, and devices.
   We carried out 32 tests with Android versions that allow the ARCore library: Android 7.0 (Nougat), 8.0 (Oreo), 9.0 (Pie), and 10. We carried out tests with the following range devices: Samsung, Xiaomi, Huawei, and Motorola (each with different OS versions and models). From the tests carried out, we can conclude that the rendering of the different 3D objects is useful thanks to each device's hardware characteristics and the support of Google ARCore Services installed by default in the mentioned devices and complements the rendering. There is no current support for devices with Android versions less than the above mentioned because they do not meet the hardware conditions to run the Google ARCore tool.
2. **Visual Distinction:** AR applications can lose visual distinction based on the physical world in which they operate. We carried out the following tests:

   a) *In an environment with intense light:* we carried out two tests, one outdoor in broad daylight (Fig. 7a) and the other indoor with harsh lighting (Fig. 7b).
   b) *In an environment with low or intermediate lighting:* we carried out a test at night (Fig. 8a) and another indoors at night and with little lighting (Fig. 8b).

From the tests carried out, we can conclude that:

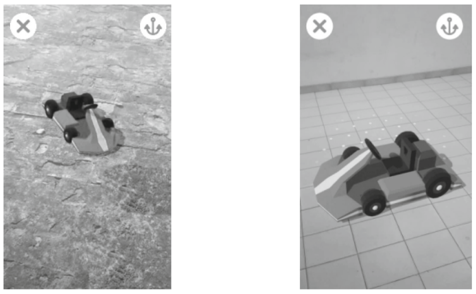

(a) Outdoor, daylight (b) Indoor, with lighting

Fig. 7. Visual distinction tests in bright light.

- It has been possible to carry out tests in scenarios with different lighting characteristics, which is a primary factor when rendering a virtual object in the real environment.
- The anchorage of objects and their three-dimensional characteristics such as height, width, and depth have been precisely maintained.

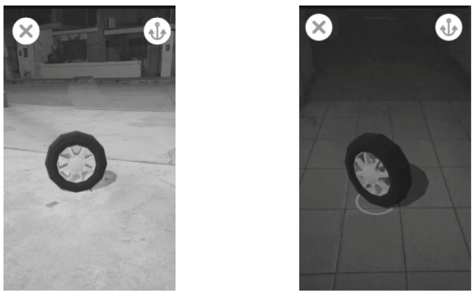

(a) Outdoor, at night (b) Indoor, at night, and low lighting

Fig. 8. Visual distinction tests with low or intermediate lighting.

7 Conclusions

In this work, we present a UL App for teaching the basic concepts of programming to overcome the learning problems of the ideas that include the control structures.

We describe the main modules, such as the compilation module and the 3D manager module. The compilation module allows translating the recorded geographical locations of the student's activities in the ubiquitous environment. The tests carried out on the compiler show that it is feasible to have a UL App that accurately converts coordinates into instructions in a language, which students can then use to program. The compilation module is useful, novel, and widely applicable since there is no history of this type of translation from geographic location to program instruction. It can also be displayed graphically and extend to generate more complex instructions later.

The 3D manager module is a complete and efficient subsystem that will allow the rendering and manipulation of three-dimensional models. These models can be viewed on any Android device compatible with the implemented technology, leaving the possibility of working not only with objects but with more complex scenarios totally in 3D thanks to the development of the said module.

As future work, we will carry out the App integration tests and tests in real contexts to evaluate students' learning in programming courses. In this way, both geolocation and UA and AR result in powerful tools that enable more prosperous and motivating learning spaces.

References

1. Burbules, N.C.: Ubiquitous learning and the future of teaching. Encounters **13**, 3–14 (2012)
2. Martínez, L.V., del Moral Pérez, M.E.: Geolocalización y realidad aumentada para un aprendizaje ubicuo en la formación inicial del profesorado. @tic revista d'innovació educativa **21**, 40–48 (2018)
3. Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B.: Recent advances in augmented reality. IEEE Comput. Graphics Appl. **21**(6), 34–47 (2001). https://doi.org/10.1109/38.963459
4. Bozkurt, A.: Augmented reality with mobile and ubiquitous learning: immersive, enriched, situated, and seamless learning experiences, January 2017. https://doi.org/10.4018/978-1-5225-1692-7.ch002
5. Insuasti, J.: Problemas de enseñanza y aprendizaje de los fundamentos de programación. Revista Educación y Desarrollo Social **10**, 234–246 (2016)
6. Salinas, J., Marín, V.I.: Pasado, presente y futuro del microlearning como estrategia para el desarrollo profesional. In: Campus Virtuales, Huelva, España, vol. III, no. 2, pp. 46–61 (2014)
7. Acosta, D., Álvarez, M., Durán, E.B.: Compilador para traducir ubicaciones geográficas en instrucciones atómicas en una aplicación de aprendizaje ubicuo de programación. In: XXVI Congreso Argentino de Ciencias de la Computación (CACIC) (Modalidad virtual, 5 al 9 de octubre de 2020), pp. 650–659 (2020)
8. Tsai, C.-W., Shen, P.-D., Tsai, M.-C., Chen, W.-Y.: Exploring the effects of web-mediated computational thinking on developing students' computing skills in a ubiquitous learning environment. Interact. Learn. Environ. **25**(6), 762–777 (2017)
9. Chookaew, S., Wanichsan, D., Hwang, G.-J., Panjaburee, P.: Effects of a personalised ubiquitous learning support system on university students' learning performance and attitudes in computer-programming courses. Int. J. Mob. Learn. Organ. **9**(3), 240–257 (2015)
10. Hosanee, Y., Panchoo, S.: The analysis and the need of ubiquitous learning to engage children in coding. In: Fleming, P., Lacquet, B., Sanei, S., Deb, K., Jakobsson, A. (eds.) ELECOM 2018. LNEE, vol. 561, pp. 268–276. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-18240-3_25
11. Boonbrahm, S., Boonbrahm, P., Kaewrat, C., Pengkaew, P., Khachorncharoenkul, P.: Teaching fundamental programming using augmented reality. Int. J. Interact. Mob. Technol. (iJIM) **13**(07), 31–43 (2019)

12. Salazar Mesía, N.A.: Realidad Aumentada en la Enseñanza de Conceptos Básicos de Programación. Universidad Nacional de La Plata, Facultad de Informática (2015)
13. Tan, K.S.T., Lee, Y.: An augmented reality learning system for programming concepts. In: Kim, K., Joukov, N. (eds.) ICISA 2017. LNEE, vol. 424, pp. 179–187. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-4154-9_22
14. Chen, D.R., Chen, M.Y., Huang, T.C., Hsu, W.P.: Developing a mobile learning system in augmented reality context. Int. J. Distrib. Sens. Netw. **9**(12), 594627 (2013)
15. Kolman, B., Hill, D.: Algebra Lineal, 8th edn, pp. 216–224. Pearson Educación, México (2006)
16. Scheibmeir, J., Malaiya, Y.K.: Quality model for testing augmented reality applications. In: IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York City, NY, USA, pp. 0219–0226 (2019). https://doi.org/10.1109/UEMCON47517.2019.8992974