Proceedings of the Federated Conference on Computer Science and Information Systems pp. 649–652
DOI: 10.15439/2020F147
ISSN 2300-5963 ACSIS, Vol. 21

The Use of ARCore Technology for Online Control Simulations

Matúš Pohančenik, Jakub Matišák and Katarína Žáková
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Ilkovičova 3, 812 19, Bratislava, Slovakia
xpohancenikm@stuba.sk, jakub.matisak@stuba.sk, katarina.zakova@stuba.sk

***Abstract*—The paper describes an educational mobile application that controls the 3D model of towercopter using augmented reality for smartphones. It is developed using the ARCore technology that allows insertion of 3D objects into a real space via smartphone or tablet. The application serves as a simple guide for a real device which is placed in laboratory and enables to create simulations based on user input data. The application interface is connected with Scilab API simulation module that provides data for 3D model animations. Users can set their own controller parameters into the predefined control structures. Application is a part of virtual laboratory and can help students with understanding of problems connected with education process.**

***Index Terms*—ARCore, towercopter, augmented reality, computer based education, 3D model**

I. Introduction

AUGMENTED reality is a technology that enriches our physical world and adds digital information to it. In simple terms, it is a kind of a system combining physical and virtual worlds with the most accurate connection of virtual and real objects. Widespread adoption of augmented reality has been on the rise in recent years and today its use can be seen in a variety of areas from the entertainment and industry (e.g. [1]) to medicine (e.g. [2]), teaching (e.g. [3]) and to the military.

In the presented paper we focus on the field of interactive education. Many researches proved that interactive AR education can help with students results (e.g. [4]). We think that augmented reality has a huge potential to change the ways and methods of education, and has the opportunity to make it much more engaging. Most students are active users of smartphones that they use to access social networks or play games, but they don't use smartphones for study purposes. AR for smartphones with additional information has a huge potential to provide students better and easier understanding of complex information.

Augmented reality is still a relatively unknown concept in the educational process, even though it provides completely new and more interesting ways of learning. It makes it easier for teachers to get students' attention and motivate them, while students get new tools to visualize their subjects and complex concepts (e.g. [5]), as well as to gain practical skills.

The aim of this work is to create an application for mobile devices using AR, which will allow the rendering of complex objects and transform them into virtual 3D models, thus facilitating the understanding of abstract and complex content. Large numbers of people better absorb new information by direct seeing how they behave and function. The goal of our application is therefore to focus genuinely on visual learning and thus help students to better understand theoretical knowledge on real devices.

II. Application

The created application enables to simulate the behavior of a towercopter, which is a laboratory model of a flying machine with a one degree of freedom ([6], [7]) and should serve as an aid in teaching on subjects dealing with the basics of automatic control. In the next part of the work we will introduce the application in more detail. The paper describes architecture of the application, system requirements, used technologies and some challenges or problems we encountered during development.

A. Development Tools

The rapid growth of AR has required significant investment from leading technology companies. Technology that once seemed futuristic is now a reality thanks to the developments that have made it possible. It all started with technology giants Google and Apple, who introduced development tools for creating AR applications for Android and iOS devices [8].

After studying the issue of augmented reality, all available platforms and its development [9], we decided to develop the application using the Google ARCore framework, that operates on devices with the Android operating system. As a development environment, the Unity platform has been chosen because of its compatibility with the ARCore and previous experience. The application can be run on all Android devices that support the augmented reality and run at least version 7.0 of the Android operating system. Devices must have installed the ARCore, which can be downloaded from the official Google Play store.

This work has been supported by the Slovak Grant Agency, grant VEGA No. 1/0733/16, the grant APVV SK-IL-RD-18-0008, by Young Researchers grant program of the Slovak University of Technology, the grant 3DVinOE and by Tatra Banka Foundation grant program E-talent, the grant No. 2018et016. This support is gratefully acknowledged.

B. Application Requirements

The app should provide a full-fledged augmented reality experience. It should scan the environment and recognize vertical and horizontal planes and allows the user to place virtual 3D models of towercopter on these scanned surfaces. It should also be possible to move 3D models on these surfaces and rotate them.

The used devices must have a camera with a sufficiently high-quality image and sensors to monitor the movement of the device so that it is possible to determine as accurately as possible how the device moves in real time and thus ensure the most accurate mapping of the surrounding environment. Last but not least, the device must have a sufficiently powerful processor be able to make calculations related to mapping and definition of objects quickly and accurately. A complete list of supported devices can be found on the official Google ARCore website [10].

The following list summarizes all requirements to the application. It should allow *adding objects to scene, objects manipulation, models simulation, graph rendering, displaying details of components and data saving and sharing.*

C. System Architecture

The ARCore app essentially works like a game. There is a certain scene on which each frame is drawn, where it updates images. This process is shown in Fig.1 in the ”Image Rendering” node. The whole life cycle of application is also shown in Fig. 1.

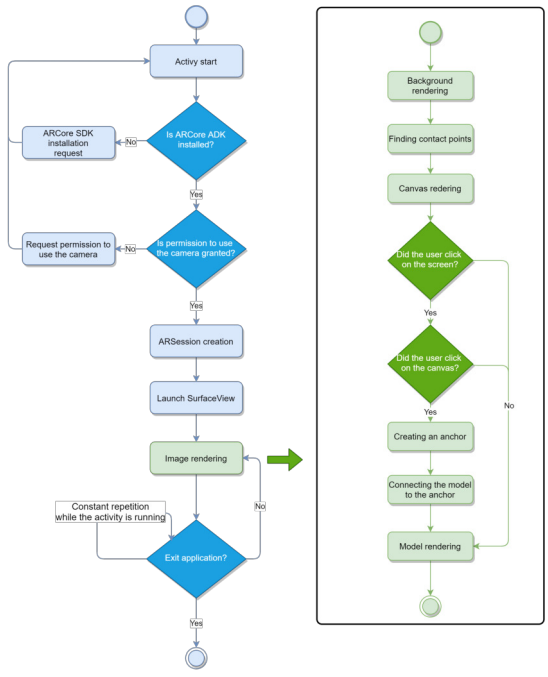

Fig. 1. Life cycle of the scene

The main entry point for the ARCore API is the Session class. This class manages the state of the system and handles the life cycle of the entire application. It allows the user to create a relation, configure it, start or stop it, but most importantly it allows access to the camera image and the position of the device. However, before creating a new session, it is necessary to verify whether ARCore is installed. It is also necessary to have to have the permission to access the camera equipment granted. If one of these conditions is not met, it is not possible to create a session and it is not possible to start and use the application [11].

The image rendering action itself consists of several smaller operations that must be performed sequentially: *background rendering, finding points of contact, canvas rendering and model rendering.*

D. 3D Model

In the application we use a virtual model of towercopter [6], which is a faithful copy of the real plant. The basic framework of this model was created based on [7].

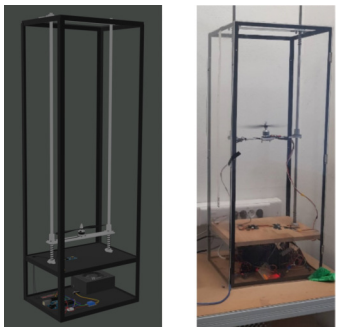

Fig. 2. Virtual and real towercopter model

The communication part consists of *ToF sensor VL53L1X, electronic speed controller EMAX Simon Series 25A, Keyes SR1y relay module, power supply, cables, Arduino UNO R3 (3D model obtained from the [12]).*

The towercopter model was originally exported from Blender in the .dae (Collada) format imported into Unity, but all colors and materials as well as complete cabling were lost. During the second export, we tried to use the .obj format. However, after the import, some materials and colors were lost again. Finally, we chose .fbx format, which was imported into Unity without any loss. For better clarity, we also created a second variant of the model, which has the communication part located next to the main frame. This option is intended to give users a more direct view to the individual components, especially on devices with smaller displays. All components can be seen in Fig. 3.

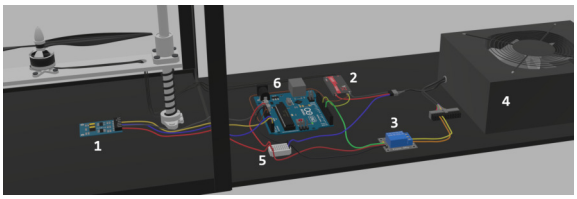

Fig. 3. Components

E. Components Interaction

Unity uses the so-called Game Objects as basic objects that represent models, props, and scenery. They do not perform any functionality on their own, they are just kind of containers for components that implement the actual functionality of the application. The most important objects used in application are as follows: *First Person Camera, Manipulation System, Object Generator, Event System, etc.* Of course, the application has to use more Game objects.

As it is usual during development, we did not miss a few problems that complicated our work when implementing some functions of the application.

The first problem we encountered was related to the interaction with the UI components of the user interface. More precisely, if we clicked on the button that was located above the scanned planes, this click also called the function that placed the 3D models in the virtual environment. Unity has a special EventSystem class implemented to detect such interactions, which is responsible for processing events in the scene. When looking for a solution, we found that this is a known bug that has been present on Android devices for a long time and still cannot be completely removed by developers. With this fact, we were forced to create our own method to control the interaction of UI components.

Another complication was encountered when implementing the screen for a visualization of individual component. The screen is not designed as a separate scene because its change would cause the lost of the entire main AR scene. Therefore the detailed component screen is only displayed as a panel that covers the entire main stage. However, when rendering the panel together with the 3D objects of the individual components, this panel was magnified several times. The source of this behavior was an AR script, which is responsible for rendering the background from the camera's point of view. We were not able to eliminate the problem, due to the fact that this script is responsible for a very important functionality of application and we did not want to make extensive changes to it. As an alternative we chose the deactivation of the main application screen whenever the component detail is displayed. This means that while the detailed screen is displayed, the application does not draw a real image from the device's camera. After closing the detail, the script is run again without any loss of objects in the AR scene.

F. Gestures

The application allows to manipulate virtual objects on scanned planes. This functionality is provided through gestures. All supported gestures are shown in the Fig.4.

Individual gestures are introduced in the following list (from the left side): *selection, movement on the screen, rotation, vertical movement (lifting), resize.*

Fig. 4. Gestures [13]

G. Components Details

Last but not least, the application supports the display of basic information about the individual components of the towercopter connection. After clicking on one of components, a panel containing the rotating 3D model of the component, its name and the basic information that characterizes it is displayed.

H. Simulation Process

The main function of application includes the ability to simulate towercopter behavior based on user-specified input parameters. The application enables this functionality thanks to the connection with the API simulation module, which provides us with real-time data necessary for the most accurate display of the simulation [14]. It uses the simulation software SciLab/Xcos, in which it is possible to create and simulate behavior of the system using block diagrams.

The device must have access to the Internet in order to run the simulation and perform it successfully. It is also necessary that at least one virtual object (in this case a towercopter) is placed on the scanned planes. After pressing the "Simulate" button in the main menu, the form will be displayed on the left side of the screen. First, users are allowed to choose the type of the controller that should be used in the simulation. At present, application supports the behavior of a PID controller, but it allows easy expansion with possible additional block diagrams in the future. After selecting the controller, a form for the change of the simulation parameters is displayed. The form is pre-filled with the predefined values of the controller and additional simulation parameters. Items in this form are generated automatically based on the selected controller. In the presented experiment, the following parameters can be set: *P, I, D, height (cm), sampling period (s), simulation time (s).*

After submitting the form, the simulation data are obtained through the HTTP GET request to the web interface of the API simulation module. The request is implemented asynchronously so that the application can still be used until a response with data arrives. The output data are returned in JSON format and contains two variables: *time* and *height*. Then the data are processed. The individual values are gradually used to modify the coordinates of individual moving objects (platform, propeller, engine ...) and in this way the process of rendering the movement of the towercopter 3D model can start. Fig.5 shows the sequence diagram that graphically describes the process of obtaining simulation data.

The simulation of the towercopter movement is plotted in real-time on a graph. Since Unity is primarily a game engine, it does not contain any of its own libraries that would support the drawing of graphs (it allows only to use assets), so we had to design own implementation from scratch.

When the simulation starts, the graph is displayed at the bottom of the screen (Fig.6 left). It can be minimized or possibly completely closed. After the simulation, the user is allowed to share the simulation data through all applications that support this feature on Android devices or save them directly to the device memory. The export format is .csv that is

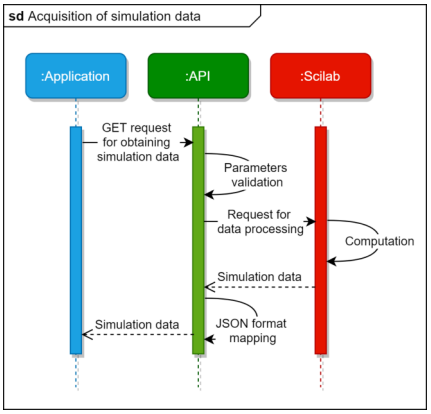

Fig. 5. Obtaining of simulation data

compatible with many often used programs e.g Matlab, Scilab or MS Excel. The graph plots the elevation curve, with the X-axis of the graph representing time in seconds. In Fig. 6 (right) the application from user's point of view can be seen.

One of the challenges was the creation of moving cables during takeoff. Unity could address individual cables as separate objects, but it cannot address individual joints, because model was created in Blender. Therefore we modified the basic model so that the part of the cables that would move on its own was replaced with a component based on the Bézier curve of three points.

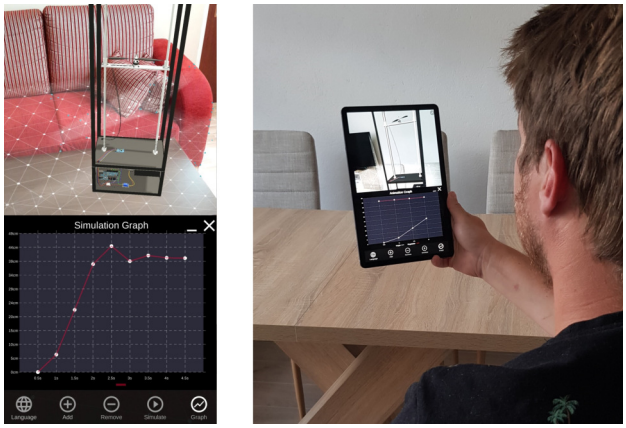

Fig. 6. User point of view

III. Conclusion

The augmented reality still has to overcome some challenges. Over the next few years we are likely to witness a shift in the development of the concept of augmented reality, in terms of software, hardware and a multitude of new applications.

The main functionality of created application is the ability to simulate the behavior of a towercopter based on the selected controller. Currently, the application supports a PID controller, but it can be easily expanded. Before starting the animation, the user is allowed to specify the input parameters, then they are sent to the API simulation module, which returns the complete data based on the simulation of the block diagram. The data are gradually projected into the motion of the towercopter and they are also displayed on a graph. Simulation data can be downloaded or shared.

Among other things, the application should serve as a support tool for teaching subjects dealing with control theory. It aims to help students better understand the behavior of controllers by directly visualizing their behavior. It also provides students with access to a 3D virtual copy of a real towercopter device, allowing a detailed view of how the device is constructed and connected before students come into direct contact with it, which can streamline the teaching process and prevent possible injury or damage to the device.

As a future work, authors would like to extend the application for another devices and use the ARCore Cloud API, that allows to create shared AR applications.

References

[1] D. Nguyen and G. Meixner, "Gamified augmented reality training for an assembly task: A study about user engagement," in *Proceedings of the 2019 Federated Conference on Computer Science and Information Systems, FedCSIS 2019*, pp. 901–904, Institute of Electrical and Electronics Engineers Inc., September 2019. doi: 10.15439/2019F136.

[2] N. S. Rosni, Z. A. Kadir, M. N. M. Mohamed Noor, Z. H. Abdul Rahman, and N. A. Bakar, "Development of mobile markerless augmented reality for cardiovascular system in anatomy and physiology courses in physiotherapy education," in *Proceedings of the 2020 14th International Conference on Ubiquitous Information Management and Communication, IMCOM 2020*, Institute of Electrical and Electronics Engineers Inc., January 2020. doi: 10.1109/IMCOM48794.2020.9001692.

[3] K. Zhang, J. Suo, J. Chen, X. Liu, and L. Gao, "Design and implementation of fire safety education system on campus based on virtual reality technology," in *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017*, pp. 1297–1300, Institute of Electrical and Electronics Engineers Inc., November 2017. doi: 10.15439/2017F376.

[4] M. B. Ibáñez, Á. Di Serio, D. Villarán, and C. Delgado-Kloos, "Impact of visuospatial abilities on perceived enjoyment of students toward an ar-simulation system in a physics course," in *IEEE Global Engineering Education Conference, EDUCON*, vol. April-2019, pp. 995–998, IEEE Computer Society, April 2019. doi: 10.1109/EDUCON.2019.8725185.

[5] M. T. Abhishek, P. S. Aswin, N. C. Akhil, A. Souban, S. K. Muhammedali, and A. Vial, "Virtual Lab Using Markerless Augmented Reality," in *Proceedings of 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering, TALE 2018*, pp. 1150–1153, IEEE, jan 2018. doi: 10.1109/TALE.2018.8615245.

[6] L. Karcol, "Interaktívny WebGL model "towercoptera" [in slovak]," Master's thesis, Slovak University of Technology in Bratislava, 2017.

[7] P. Ťapák and M. Huba, "One Degree of Freedom Copter," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018. doi: 10.1007/978-3-319-74727-9_11.

[8] P. Duffy, "Augmented reality," *Nextechar Solutions*, October 2019.

[9] P. Nowacki and M. Woda, "Capabilities of ARCore and ARKit Platforms for AR/VR Applications," in *Advances in Intelligent Systems and Computing*, vol. 987, (Brunów, Poland), pp. 358–370, Springer Verlag, 2019. doi: 10.1007/978-3-030-19501-4_36.

[10] Google LLC, "Google arcore supported devices." https://developers.google.com/ar/discover/supported-devices/. 16.06.2020.

[11] Google Developers, "Session — ARCore — Google Developers," 2020.

[12] Evan Boldt, "Blender arduino model." https://robotic-controls.com/learn/arduino/blender-arduino-model.

[13] Tom Loots, "Touch gestures." https://dribbble.com/shots/1383148-Touch-Gestures-freebie, 2020.

[14] P. Milán, "Komunikácia medzi 3D enginom a simulačným prostredím [in slovak]," 2019.