Exploration of Underwater Scene using Unity game framework and Apple ARKit framework in Augmented Reality with mobile devices

Michal Šindelář, Fotis Liarokapis

***Abstract*— This paper describes an implementation of Augmented Reality application focused on an underwater exploration. The application is developed for devices with iOS using ARKit framework in cooperation with Unity game platform.**

I. INTRODUCTION

The aim of this project was to discover possibilities of cooperation of Unity game engine framework with recently announced Apple ARKit framework for creating an Augmented Reality (later AR) mobile application. At first we tried to use existing Unity scenes and improve them to be able to compile to iOS devices.

There was a big explosion of AR applications for mobile devices in the last year and we can expect the expansion to grow even more. The most common AR applications in this area such as Snapchat, Instagram or Facebook Messenger aimed for entertainment with tremendous features creating funny effects using various superimposing features for faces, etc. Except for these entertainment applications, there were introduced also more serious ones, which are supposed to help in more thoughtful areas. For instance, a few months ago Ikea started to promote their own products with a brand new AR application that is able to place the products such as tables, chairs, etc. to rooms and let the customer to evaluate how the object suits to the place.

The biggest potential in AR mobile applications is their applicability. Currently, most of the people have their own smartphones with most used operating systems - Android or iOS. Most of the phones have enough computation power to perform needed computation for AR applications purposes. The second advantage of mobile AR applications is the fact that most people are used to carry their phones with them all the time.

Similar applications for underwater exploration were already implemented for Virtual Reality (later VR) devices, such as the 'European underwater cultural heritage' project for HTC Vive[1]. This project aims for the general public - most probably - without a VR device, to provide similar experience only through a mobile phone.

II. BACKGROUND

A. ARKit

In the autumn of 2017 on the Apple's keynote presentation a new ARKit application was announced - a programming interface that lets developers build AR applications powered by this framework. ARKit takes a device's camera, CPU (central processing unit), GPU (graphic processing unit), and motion sensors to provide functions needed for AR experience.

The most important feature of ARKit is probably a markerless plane detection, which is essential for most of the applications built on top of this framework. Plane detection is performed by scale-invariant feature points (later feature points) detection and extraction. ARKit session reads video frames from camera continuously and performs the feature points analysis in the realtime. Currently, ARKit supports only horizontal planes, but it is almost certain that the vertical support is coming soon, as even the Plane class has an interface for vertical plane handling.

B. Unity ARKit Plugin

Before the ARKit announcement, engineers from Apple collaborated with engineers from Unity and prepared a beta version of Unity ARKit Plugin. The purpose of this plugin is to provide a middleware between Unity and iOS programming languages (Swift or Objective C). This plugin enables using native ARKit functions, such as the already mentioned plane detection, directly in Unity. The most simple (and useful at the same time) usage of this functionality is to superimpose the scene from Unity to plane detected by ARKit and provide AR experience for users.

III. IMPLEMENTATION

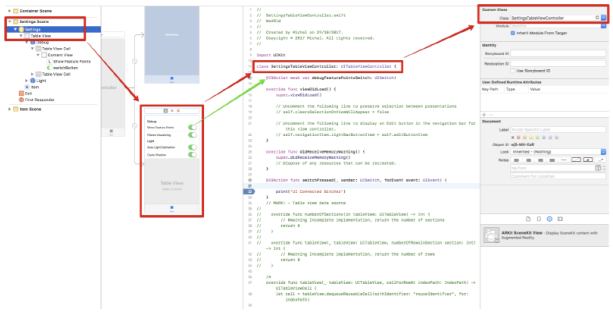
In the beginning we focused on investigating the capabilities of ARKit features themselves. Without any doubts, the main feature needed for the project was the Plane detection. The Plane detection is performed by scale-invariant feature points analysis, as already mentioned. We needed to check how stable the algorithm is, to find out whether this markerless detection approach is suitable for this application, as placing amscene requires stable detection without too big jittering and scene jumping. Additionaly, while announcing the ARKit, Apple promised also the possibility of Light Estimation in the scene, which seemed to be very auspicious as well.

For these investigation purposes we developed a simple application with basic ARKit features settings. The only functionality of this application was to superimpose a globe into the scene and try

- plane detection,
- light estimation,

[1]F. Bruno et al., "Development and integration of digital technologies addressed to raise awareness and access to European underwater cultural heritage. An overview of the H2020 i-MARECULTURE project," OCEANS 2017 - Aberdeen, Aberdeen, 2017, pp. 1-10.

- visualising the feature points.

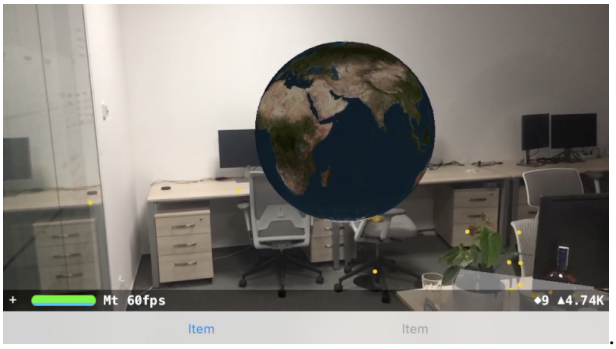


Upon trying the application in different conditions, it was confirmed that the plane detection is stable enough for placing the objects or the Unity scene in the future. The revealed problems were more or less expected. Mainly

- poor lightning,
- lack of texture,
- fast movement.

We also investigated the way how the camera on an Apple device is calibrated. The calibration seems to be done by computing the movement of feature points in the two sequential video frames with accelerometer support. The interesting fact is that ARKit actually uses all the common detected feature points in the two frames, not only one pair, what is common in other algorithms.

The last essential thing we wanted to get through was the auto light estimation. From the testings scenarios we found out that the estimation works only on the intensity level estimation, not for the directional one. Based on that the estimated shadows do not work as properly as they could in case of full directional estimation. According to Apple's updates of iOS it seems that the mentioned directional estimation will follow.



After initial research about the framework itself we moved to existing Unity application that tried [2] to provide digital accessibility of underwater sites and maritime archaelogy.

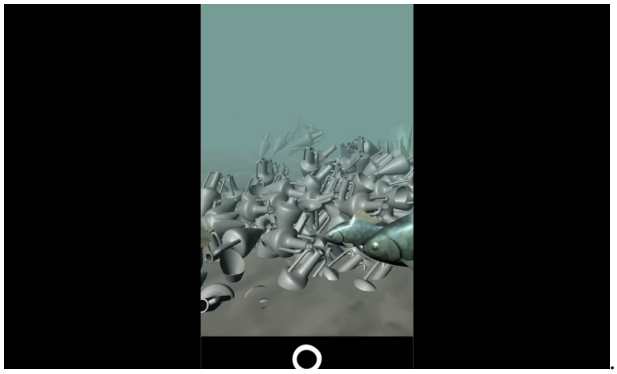
The very first idea was to discover the possibility of only adjusting the existing scenes a bit and making them compilable using ARKit Unity plugin to iOS devices. We spent quite a long time trying in debugging internal middleware errors. There are many things that are impossible for the middleware to compile. There are only limited possibilities in color space, the application after compilation try has serious memory issues that end with crash, and many others. This approach ended as a bad decision and we decided to develop an application rather from skratch.

We decided to develop an application which should combine both VR and AR concept using Portal like application. This kind of application superimposes portal into real world and lets the user to come into the portal and see the virtual reality scene. When looking from the virtual scene to the portal, the user is able to see the real world and vice versa.

We created an underwater scene in Unity with basic environment settings trying to imitate the underwater surrounding. The core effects are implementing of fog that should act as a water opacity and lightning of sun raying through the water to the ground.

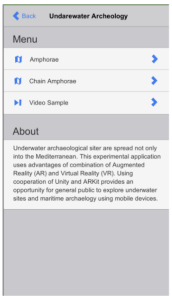
[2]F. Bruno et al., "Development and integration of digital technologies addressed to raise awareness and access to European underwater cultural heritage. An overview of the H2020 i-MARECULTURE project," OCEANS 2017 - Aberdeen, Aberdeen, 2017, pp. 1-10.

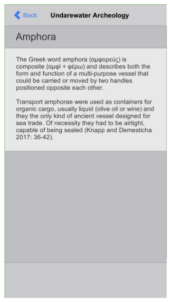
The most interesting objects are definitely the real models of the amphorae. Except for these core objects, there are also animated fish, model of a shipwreck, whale swimming above the user and rocks placed randomly on the floor.

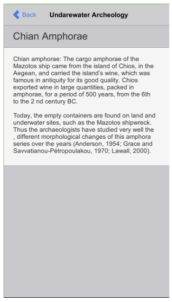


In the application there is implemented simple User Interface (later UI) menu divided into three sections:

- Amphorae,
- Chain Amphorae,
- Video Sample.







In the first two sections the information about respective topics is present. In the Video sample user can see a video of real underwater exploration.

IV. RESULTS / CONCLUSION

We discovered the possibilities of using the ARKit features for iOS devices in the development of serious AR applications. There is no straightforward solution of adjusting existing Unity scenes and applications to be able to compile using ARKit Unity plugin. It seems that a less time consuming solution might be to reimplement and respect the restrictions since the beginning.

ARKit framework seems very promising in the future with its markerless planes detection that could be extended to the vertical plane detection in a near future. Also, there is big potential in cooperation of ARKit with powerful Vision framework, which is based on huge Apple Core Machine Learning framework.

We succeeded in creating a working application for exploring of underwater scene with real models of Amphorae. The control of the application is not completely straightforward, as the user has to hold the mobile phone all the time. However, the potential of AR mobile application in general is really big, especially due to mobile devices portability and usability.

The application is compilable on iOS > 10 and requires processors higher than A9. Application is optimized for resolution of iPhone 8 (1334px x 750px).

Except for the main purpose of this project, we invested time also in exploring the cooperation with Vision framework. For example, using its features it is possible to implement also the standard marker detection, perform gesture detection in very simple way, using pre-trained machine learning models for detecting objects in the scene and uncountable other usages.

V. FUTURE WORK

Using mentioned usages of Vision framework, it would be possible to implement controlling of the application using hand gestures, what could be the next step for better user experience of the application.

In a future, an excavation possibility should be added to increase the interactivity of the application. This effect requires proper sand simulation, which in recent projects seemed to be quite hard for computation and needs to be optimized for mobile devices with limited computational power.

REFERENCES

[1] "ARKit." ARKit, Apple Inc., 2018, developer.apple.com/arkit/.
[2] "Vision." Vision, Apple Inc., 2018, developer.apple.com/vision/.
[3] Carvajal, Ignacio Nieto. "Augmented Reality with ARKit: Detecting Planes." Digital Leaves, Digital Leaves, 23 Oct. 2017, digitalleaves.com/blog/2017/10/augmented-reality-with-arkit-detecting-planes/.
[4] "Unity ARKit Plugin." Unity-Technologies / Unity-ARKit-Plugin, bitbucket.org/Unity-Technologies/unity-arkit-plugin/.

[5] Bruno, F., Lagudi, A., Ritacco, G., Agrafiotis, P., Skarlatos, D., Čejka, J., Kouřil, P., Liarokapis, F., Philpin-Briscoe, O., Poullis, C., Mudur, S., Simon, B. Development and integration of digital technologies addressed to raise awareness and access to European underwater cultural heritage. An overview of the H2020 i-MARECULTURE project, OCEANS 2017, IEEE Computer Society, Aberdeen, Scotland, 19-22 June 2017. To Appear.

[6] F. Bruno et al., "Development and integration of digital technologies addressed to raise awareness and access to European underwater cultural heritage. An overview of the H2020 i-MARECULTURE project," OCEANS 2017 - Aberdeen, Aberdeen, 2017, pp. 1-10.

[7] Paladino, T. Apple AR: Developer Opens Portal to Underwater World with ARKit. 11 Jan. 2018, https://mobile-ar.reality.news/news/apple-ar-developer-opens-portal-underwater-world-with-arkit-0179442/