Final Report

Augmented Reality Guidance with Vuforia SDK

Mark Hanych, Aleksei Sakhnov

Introduction

As the title suggests, our project focuses on Augmented Reality guidance, specifically on implementation of AR guides using Unity (versatile 3D development platform) and Vuforia, an augmented reality software-development kit.

Augmented reality itself is an interactive experience of a real-world environment where real objects are enhanced by computer-generated information.
In our case, it opens up an opportunity to use AR guides - interactive tutorials, which are used in a real-world location and provide useful information via e.g showing informative hints over the surrounding environment.

Such guides can be viewed on a vast majority of digital devices, most common ones being tablets and smartphones. Unity enables a possibility to create a cross-platform solution for a broad list of digital platforms, however the scope of our project is narrowed exclusively to Android-powered devices.

Project Goal

Our project is, in fact, a follow-up to another project that took place a few years ago. It had the same purpose - creating a cross-platform AR guidance application using the capabilities provided by Vuforia and Unity.
The main idea behind such AR functionality is to set up an image target, which can be then recognized and used as a reference point in order to correctly place the computer-provided information.

Using such targets all over the real-world location is highly impractical. But among Vuforia’s features there is a method called extended tracking. It enables an option to navigate through location without permanently keeping the reference point in camera view - to do so, it utilizes a bunch of various device sensors like gyroscope or accelerometer, as well as calculates “points of high interest” from camera data, for example room corners.

Unfortunately, at that time tracking quality offered by Vuforia tools wasn't good enough, yielded coordinates were inaccurate which resulted in generated objects often being badly misplaced and the idea was discarded.
However, after some time has passed, Vuforia developers made a claim which stated that their extended tracking technology has improved significantly.
As a result, the main goal of our project is to create a functioning AR-guiding application and test if claimed improvements of extended tracking could once again enable the idea of cross-platform, target-less AR interactions.

Pre-Available Functionality

Since RUB already has a working AR guide application based on non cross-platform tools, we were able to use an associated server with a REST API that is able to provide full guide details, including coordinates of every object that should be generated (further referred as "hint"), in a form of JSON response.

General Structure

The project can be divided into three distinctive parts: the API, the UI and Vuforia. API part is about server connection, handling its responses and parsing into suitable format, UI part means everything about application GUI, including the underlying logic, while Vuforia part is of course about implementing the AR scene, image target, tracking and hints.
We decided to divide our responsibilities, so that we could fully focus on our tasks and complement each other instead of interfering, thus Mark (me) worked with API and UI, while Aleksei was implementing all the Vuforia elements. We've also divided our report in the same fashion, because even though we both comprehend every bit of the project, our understanding of our own parts is of course much deeper.

Scene Structure

In order to avoid the complexity of transferring too much data back and forth between different Unity scenes, we've limited our project to three scenes that are responsible for the login screen, the main menu and the Vuforia AR environment, and are named LoginScene, MenuScene and VuforiaScene accordingly.
Of course, less scenes means more management of overlapping UI elements in our relatively complex menu scene, but it is solved via hiding and showing UI objects upon need, while also reusing them for different purposes.

API: Login

After testing a couple of different options, we've chosen the RestSharp C# library for communication with our server, since it's open-source, easy to use and quite powerful - for example, it has built-in JSON/XML serialization functionality, as well as a variety of authorization options, one of which we would definitely need for data exchange, since API is quite limited if accessed without prior authentication.
Server uses the JWT (Json Web Token) authorization standart, which means that every non-public API request should be made with an attached "token" header.

The token itself can be acquired through another authorization process, which utilizes a login/password combination. In the login scene, we use RestSharp's SimpleAuthenticator method to pass user-entered login and password to the API and get a Token object in return. The built-in serialization functionality provided by RestSharp wasn't flexible enough for our needs, which is why we used the much more powerful Newtonsoft's Json.NET library instead.

After initial parsing, we create a new instance of the Token class, which contains attributes like the type of the token, time until it expires and the actual token. Since the token is used in different scenes, it is saved as a valuable in a static CrossScene class. It is also serialized and cached on disk for an option of logging in without having to type all the login data every time when a user wants

to use the app. If authorization has been successful, we switch to the main menu scene.

API: Main Menu

MainMenu scene features the highest number of API calls, which are structurally scattered across two scripts, MenuScript & InfoScript. As soon as the scene loads, “category” endpoint is accessed via the GetCategories method and an array of all available categories is retrieved and placed in a UI Dropdown. Then, a connection to the “browse” endpoint is made, retrieving available tutorials which are stored as a list of the Tutorial class instances for later use.

Returned tutorial objects aren’t complete, they are missing all the “steps” that should be followed by the users. We can get a full tutorial object via the GetTutorialById function, it is used to download chosen tutorials. Along the tutorial object itself, every hint attachment needs to be downloaded separately. It is done in the DownloadTutorial method by looping through each step, accessing the “tutorial” API endpoint and downloading the corresponding attachment if it is present.

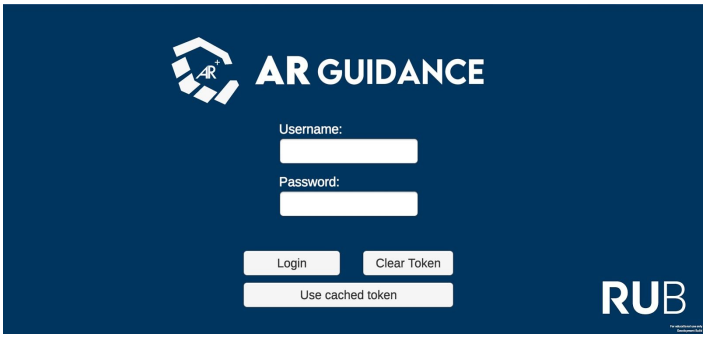
Regarding the VuforiaScene - it works exclusively with downloaded tutorials and thus doesn’t need to implement any API calls.

UI & Scene Hierarchy: LoginScene

Let’s move to the UI part. In case of our LoginScene everything is quite straightforward: besides the decorative elements like title or logo, it consists of two user inputs for username and password submitting, as well as a set of buttons allowing users to login via submitted credentials or via a token saved after previous log-in, there is also an option to clear the saved token.

It is worth noting, that every scene also contains a ScriptHolder Gameobject with each scene-controlling script attached. Static UI elements that are used in scripts are usually referenced via [SerializeField] / public declaration of corresponding variable, which is then assigned in the editor, since this is both easiest and

fastest (compared to Gameobject.Find or other similar Unity functions) way to do so.



UI & Scene Hierarchy: MenuScene

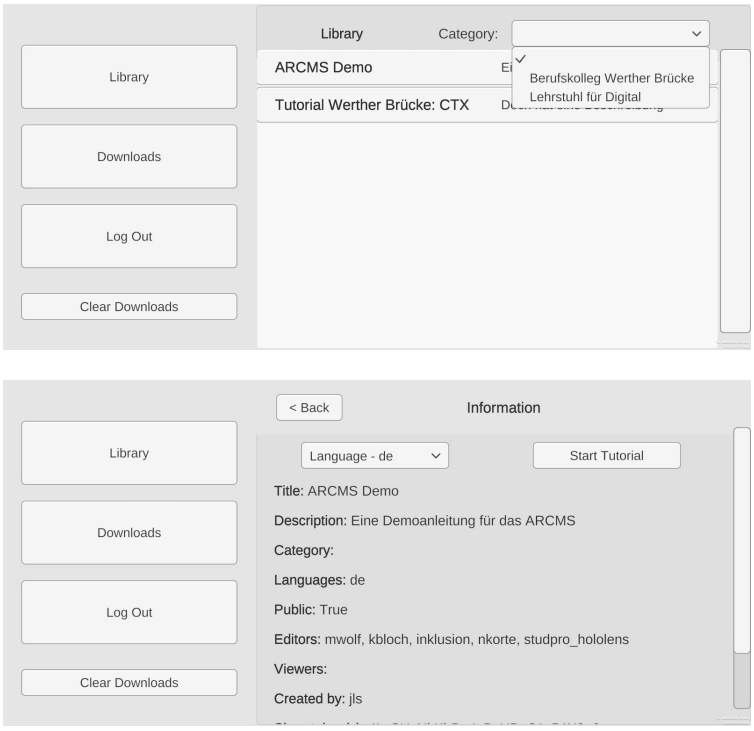
The structure of our MenuScene is a bit more complex. Its key objects are two scroll view elements named LibraryScroll and InfoScroll. LibraryScroll is designed to showcase a list of dynamically generated list items, which represent either downloaded tutorials or the ones available for download.
At first, we've tried to place list items via offsetting their height, which was a quite tedious and hardly customizable way to do the task, but then we've discovered Unity components called VerticalLayoutGroup and ContentSizeFitter, which are able to do the same job immeasurably easier and are much more reliable.
List items are implemented as button prefabs, each prefab contains a "ItemDetails" script which stores an attached Tutorial instance, handles the OnClick event and calls ShowInfo method, thus switching to the InfoScroll section.
InfoScroll element is used to showcase dynamically generated labels with general information about the selected tutorial.

On the left side of MenuScene UI there is a set of four buttons. Buttons named Library and Downloads will refill LibraryScroll with corresponding tutorial list items, LogOut button returns to the login screen, while the ClearDownloads button allows to erase all the tutorials that are stored on disk as well as their hint attachments.

Each Scroll also contains a top-based panel with a “back” button that allows to e.g. switch from detailed tutorial information back to the tutorial list, as well as a label providing the current section name (Library/Downloads/Information) and other section-related elements like category dropdown or download button.

Speaking of the category dropdown, it is filled with server-provided categories and its “OnValueChanged” event is used to make a new API call and populate LibraryScroll with tutorials that belong to the chosen category.

Also, InfoScroll contains another dropdown filled with languages that are available for considered tutorial and a Start button, which saves both the preferred language and an instance of the specified tutorial in the CrossScene script (this approach allows us to avoid usage of Unity’s DontDestroyOnLoad method and possible complications) and switches to VuforiaScene.

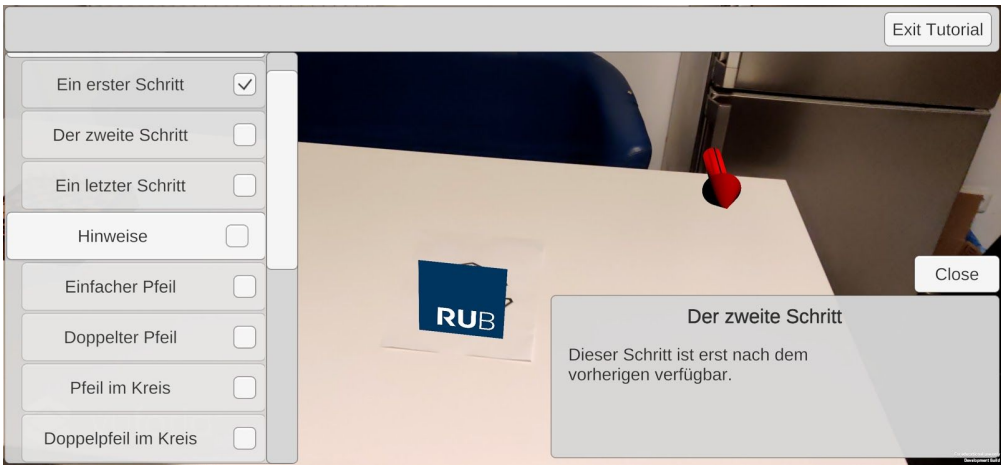


Main Menu

UI & Scene Hierarchy: VuforiaScene

This scene uses UI to provide an option to navigate through the list of tutorial steps, as well as display extended information about each step. Its key elements once again are two ScrollViews named SideBar and InfoPanel.
As the scene starts, SideBarScroll is filled with steps, extracted from the tutorial object that was previously saved to the CrossScene script.
They are divided into parent-steps and children-steps which can be distinguished by their left-margin as well as color. Just like the MainMenu list items, each sidebar item contains a SideItemDetails script which contains corresponding variables and handles clicks.

After a step has been clicked, InfoPanel Scroll is filled with step-related information like its title and description, attachment (loaded from disk and converted to a sprite), various links or symbols (loaded as sprites from Unity's Resources) and then shown to the user. In the top-right corner of the UI there is also an ExitTutorial button, which returns the app to MenuScene.



Vuforia Scene

Implementing Vuforia

VuforiaScene is the last scene in our project and the first scene where we actually implement Vuforia itself. There are two main components here: AR-Camera and Imade Target. The first serves as a representation of an actual camera on an Android device. And the second one is the keystone of Vuforia's tracking. Image Target represents an image that Vuforia Engine can detect and track. The Engine detects the image by comparing extracted natural features from the camera image against a known target resource database. Once the Image Target is detected, Vuforia Engine will track the image and create a coordinate system with origin in the middle of it. To actually augment our reality with a "hint" we should create a proper object and link it with Image Target as its child.

We have decided to implement six different hint prefabs. One of which was used to create 2D-Pictures and five others represented one 3D-Arrow each. As soon as one of the steps is selected we check if our hint has "Arrow" or "ISO7010" type. It determines if a hint will be a 2D or a 3D object.

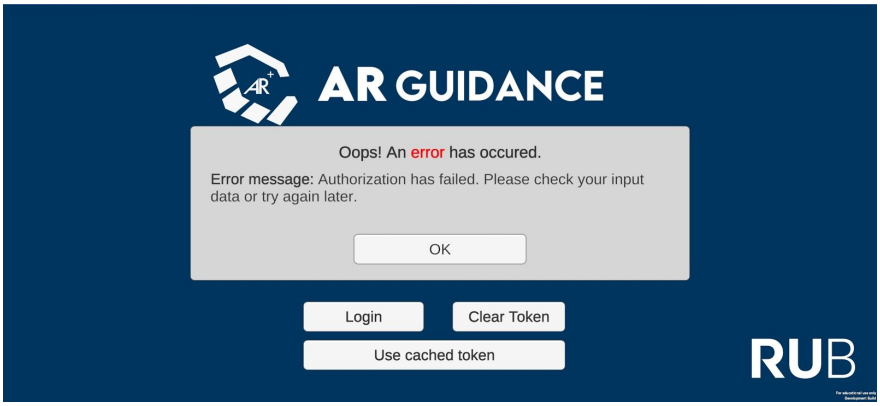
If it is a 2D-object - a proper picture from our gallery is assigned to the sprite-prefab via use of a Sprite Renderer. If it is an arrow - the color of the material assigned to all 3D-prefabs is being changed and the exact prefab is chosen through a series of "Name" property checks.

After our hint-object is created it is linked to the Image Target as a child and can now be seen through the device's camera. However we are not done yet, as our "hint" is now displayed in the very middle of the Target. So our 4x4 transform matrix we received from the server should be interpreted via "Matrix4x4" class

which is in turn translated through “TransformExtensions” function into 3x3 transform Matrix and assigned to the hint. Only after that our arrow or sign is positioned relatively in place and can be used as a real hint.

Basic Error Handling

Most of our functions are wrapped in separate try-catch clauses, this allows us to show UI-powered error messages either with the related exception message or with our own, user-comprehensible explanation.

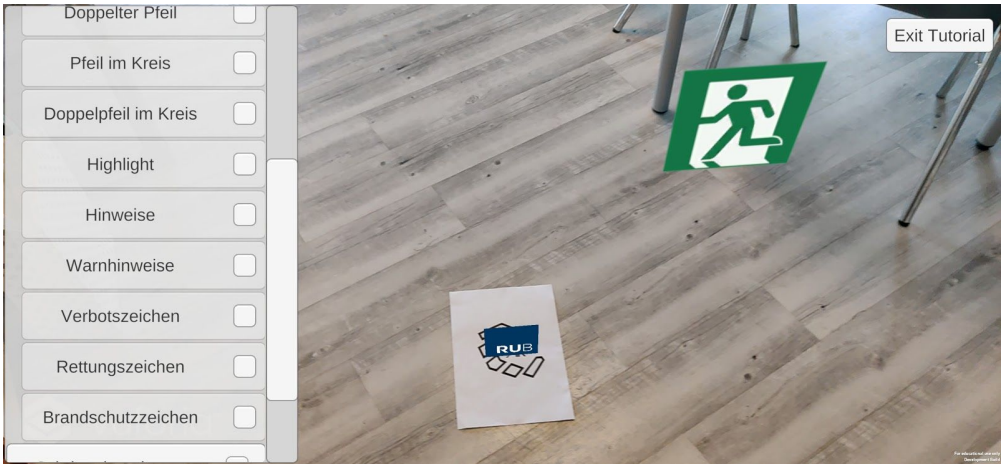


Example of an error message

Testing and Conclusions

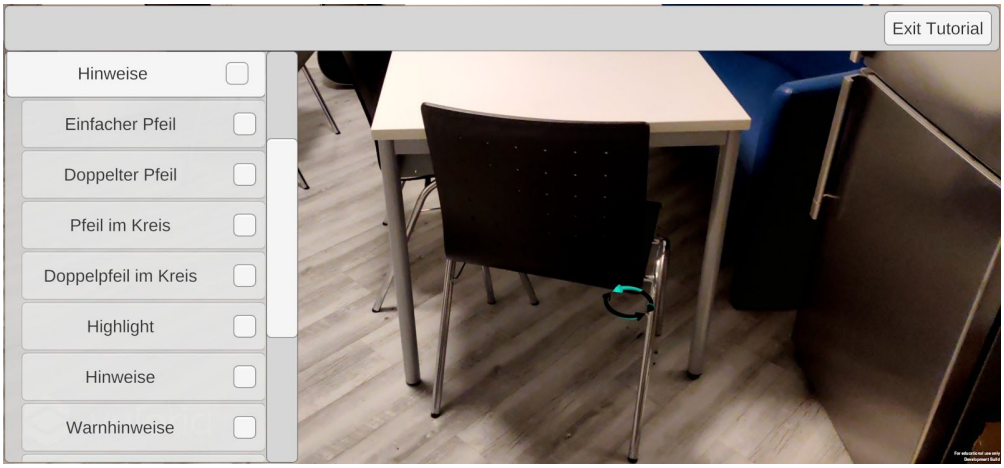
Because of the project's main goal, testing is probably one of its most valuable parts. Unfortunately, due to the ongoing pandemic we were unable to test our application at university facilities or with tablets that could be obtained for testing at the university, which is why all the tests have been done on our own Android-powered smartphones, but fortunately they have still yielded pretty definitive results. In this section, we are gathering and evaluating all the observations we were able to make.

First of all, let's address the standard, non-extended tracking, which is used by Vuforia as long as the target marker is visible. Standard tracking works great on all types of surrounding surfaces, even if it is almost indistinguishable from the marker page itself (e.g a bright white wall or table). But some tracking problems start to arise as soon as the device is moved, especially if it is rotating around the marker at the same time. However, if this process is done slowly, tracking can still keep up with the changes and successfully rearrange the coordinate system. It is also worth noting, that standard tracking is highly dependent on the surrounding light, if it is not intensive enough or there is a slight shadow on the marker - Vuforia might lose the target.



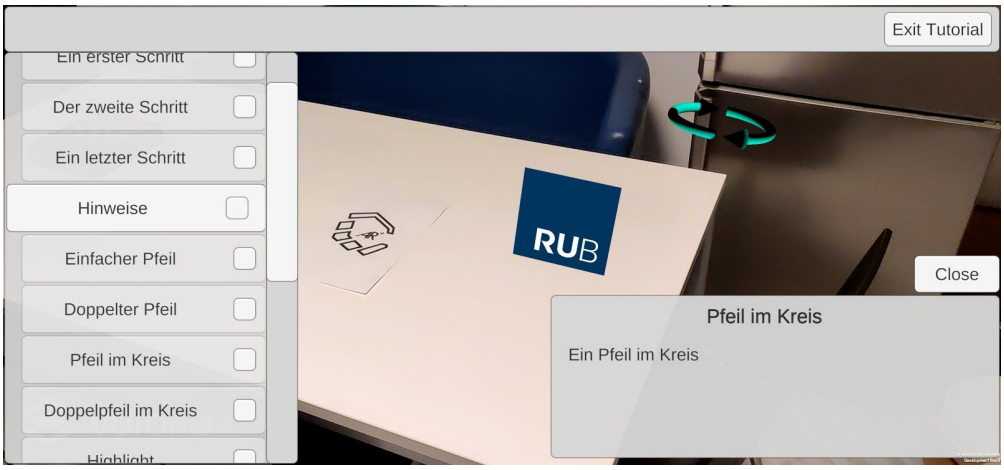
Example of a standard tracking

Now let's move to the most important part, which is extended tracking. This type of tracking occurs as soon as the target is lost from the camera, but Vuforia still receives enough information from device sensors and the geometry of the surrounding environment in order to keep the coordinate system and its objects in the correct place.
As our findings show, extended tracking seems to work pretty well while the device doesn't move too much on the x or z axis - a user can direct the camera away from the marker, look around with it and expect to find the coordinate starting point in the same location when the camera returns to the target.



Extended tracking example, target is not in view

However, as soon as the device starts moving - e.g. the person holding it makes a step to the right, both the starting point and all of the nested objects shift by the same amount, in the same direction and on the same axis (especially noticeable with X/Z axis - moving forward/back and right/left).



Example of imprecise tracking after looking away while moving and then looking back at the target

With all of the above being said, the possibility to navigate the room freely depending solely on extended tracking is quite unlikely, but it still might be possible to e.g. slowly get around an object and it is definitely possible to look away from the target while keeping a good tracking precision.

Reflexion (Aleksei's perspective)

1) Has the task or your understanding of the task changed over time? If yes how?

At first we were going to develop an app for Microsoft Hololens presented to us by our supervisor's team. However we could not get access to it as I and Mark were both stuck in our home countries because of Corona Lockdown. So we had to find another solution and we managed to do so by changing our target-platform to Android. It also took us quite some time and arguing to actually understand what we wanted to achieve with our project and how to do it.

2) How appropriate was the planning of the course of the project, especially the dates? How often did the planning have to be adjusted? Why? What was your role in it?

From the very beginning we had lots of problems with planning. Our initial plan was quite good, but we never managed to work up to it. First, before we could even start, we had to change our project goals. Then, one of our group members decided to move to another project. And finally we had lots of delays or timetable changes because of different Force Majeures. Mark has offered to take the lead in organizing our work and I decided to follow his lead and support him if and where it was necessary.

3) Which methods have you used to support the joint planning and how have they proven themselves?

We have decided to regularly meet in Zoom to report what has been already done and discuss our next steps. It has helped us to track the development process, but we both lacked developing experience to properly evaluate how much work we still have to do and how long it will take. Also on the first steps of development we have used lots of diagrams to help us create the structure of our app. This method proved very helpful, but was a bit hard to use because of our social distancing. Brainstorming was the last tool we used. It was very efficient in

dealing with bugs and testing as we could quickly analyze any problem and come with a proper solution.

4) How did the coordination during the project work? What problems did you have? How were they approached? What was your role in it?

There were three major problems in our coordination: information exchange, version control and no pre-established coding style. All of them were nearly nonexistent in the beginning, but as the development continued we have encountered complications. Especially in those parts where we had to interact with each other's code. None of us had a complete understanding of what the other person already did. Moreover, we actually were working on two different versions of the project and had to spend time on connecting them instead of just updating one. And the last problem was the difference in our coding styles, which made it difficult to understand each other's work. However we managed to overcome those difficulties as we were always ready and willing to explain everything and answer any possible question.

5) To what extent were you sufficiently competent to collaborate? What skills did you have to acquire in the course of the project? What did you learn from the study project?

There were two major stages in our project: development part and coding. The first one we did completely together as we had to build our app a proper structure and it was unwise to do it separately. This part was free or mostly free from problems and complications as we were very enthusiastic to collaborate and teem up against this challenge. Second stage, however, was much more individual as we worked separately on our parts of the task. It was here that we encountered our coordination difficulties as none of us had enough experience to foresee and predict them. Dealing with those complications has pretty much shown the importance of proper organization and a competent team leader. I personally have learned to leave enough time in planning for unexpected delays such as sudden illness.

6) What have you done to work as efficiently and reliably as possible in the project?

I tried to build my participation in the project on two keystones. First of all, I was always ready to answer Mark's questions and explain anything he needed regarding my part of the code. I was and still am sure that any problem can be overcome if we are willing to promptly and fully help each. Secondly, I strived to always stay calm and never let our anxiety or nervousness influence our work and decisions.

7) What would you do differently if you had to work on the task again with the experience you had at the end of the project?

I would definitely try to convince my future teammates to pay enough attention to building a proper foundation for our project and establishing necessary rules of interaction and coworking. As it can save much time in the future and it might be crucial not to lose time near the end of the project.

8) What challenges arose with regard to the group dynamics in the team and how did you deal with them?

I have already mentioned most of the problems we've encountered. Normally they were unpleasant as they represented our prior mistakes, but we managed to overcome each and every one of them by staying in touch, supporting each other and never letting difficulties overcome us.

Reflexion (Mark's perspective)

1. Our main task itself has drastically changed soon after the project started. Due to a lockdown in our home countries, me and Alexei were stuck there, unable to work with equipment provided by the university. After consulting our project supervisor, we've agreed to change the focus of our project from Vuforia-powered AR development for Hololens to development for Android, so that we could work from home and test everything on our own devices. It turned out to be a great idea, since a similar lockdown was introduced in Germany and the university facilities were unavailable for a long time.

My understanding of the project goals has of course also changed over time - at first, it was quite confusing to figure out what needs to be done exactly, but after a few online-meetings with our supervisor the task became pretty much clear.

2. Our initial schedule was pretty good, but further into the term our project has stalled significantly, mainly because of our unsuccessful task difficulty evaluation, but also due to my heavy COVID occurrence and family issues. Still, since I had way more enthusiasm for the project (it is quite important for me, because I already have a big delay in my study), I tried to plan things like meetings with our project tutor and development goals.

3. Mostly we've used straight-up discussion, supported by some diagrams and structure graphs, especially at the beginning of the project, since back then it was extremely difficult to grasp the necessary application architecture and workflow.

4. Since there were only two of us, the required coordination was mainly about dividing tasks and making development decisions, both of which we've done on equal terms through discussion. It's worth noting that sometimes it was difficult to coordinate our actions or plan meetings because of the quite unusual night-based lifestyle of my teammate.

5. I don't mind working in a team, but I really like dividing tasks, working on different parts in almost full separation and merging them afterwards, so that's what I've suggested for our project as well. The problem is, in projects like these you can't really fulfill your part and wash your hands, while I was highly unwilling to supervise my colleague's progress in detail, which later backfired since he got ill and was unable to finish his project part entirely by himself. Since I didn't evaluate the complexity of my own tasks correctly, I had to help him while not being done with my part yet, and as a result we've gotten into a huge time trouble near the project deadline. This was my first project of such caliber and from now on I'll definitely show deep interest in detailed progress of my future teammates and try to evaluate it myself in order to avoid similar problems. I will also regularly re-evaluate my goals and ongoing progress as thoughtfully as I

can, while also always planning to finish my tasks not “in time”, but much, much sooner.

6-7. Since we’ve got into time trouble at the end, it is pretty obvious that we weren’t efficient and reliable enough. The main things I would (and certainly will) change are: leaving a huge time margin for all kinds of Force Majeure while planning things, knowing my capabilities and correctly evaluating complexity and difficulty of tasks (and since it isn’t always possible to full extend, at least always being aware of possible chains of various complications and leaving some time margin for that, too), closely supervising the work of my colleagues and offering them to supervise mine.

8. I’ve already mentioned every challenge we’ve faced, some of them we were able to overcome through communication (e.g. different time-lifestyles), others were given to us quite difficult, (e.g time/completion evaluation), but in the end we’ve learned tons of things about ourselves, what do we have to work on and change in our approach to project management, which will greatly help us regardless of what our future projects will be about.