UCL COMP0016 Team 12 - Lilly Neubauer, Dillon Lim, Oliver Vickers

Bi-Weekly Report: Period ending in 7th February 2020

Overview: What we've done

After having completed our last submission, we had two main tasks to complete. Firstly, a new version of the AR package we are using, Unity AR Foundation, was released during the time that we've been working on the project. Lilly had realised that in order to implement our floor trigger image recognition functionality, we would need to use a Tracked Image Manager that was not included in our current AR Foundation version (2.0) so we would need to migrate to ARF 3.0. This migration was tricky because she also had to update the version of Unity we were using to 2019.3, which contains ARF3. When the project was imported, it would no longer compile because it said it was missing some required assemblies. This turned out to be a known bug with VSCode and not with Unity, and was fixed by downgrading VSCode to 1.1.3 and re-installing all the packages from package manager.

She completed this migration with some of the existing functionality and started learning how to use tracked image manager. There were some problems with working out what type of images can be used as recognition images - they need to have a certain number of features to provide accurate tracking info, so, for example, the new IBM Watson logo, which is quite simple, is not complex enough. For this reason we chose to use the older and more complex logo as our trigger image.

On the backend side, we knew that a big challenge for this term was to implement the database functionality of the backend. We had already implemented the Webhook that Watson could call out to, but instead of hard coding values we wanted this to connect with an SQL database hosted in the Azure cloud. We chose to use the SQL database because a) we have existing knowledge of SQL queries and b) we could host it in the Azure cloud which we have a student subscription to. Additionally using SQL functionality makes our backend pretty adaptable because there are many SQL database solutions to choose from. Oliver went about creating a new Azure SQL database instance and adding a "Staff Members" Table to this. He started writing basic queries to fetch data from the table to the webhook, so that we were able to demonstrate asking the Avatar "What's Maria's email?" And were able to see the query being passed through the webhook and returning so that the avatar read out the email provided in the database. This was a good step forward that showed that our pipeline worked all the way through.

Lilly also researched solutions that we could use to send a text to a staff member from the Webhook. We decided to use Twilio which is well documented and has a free tier. She created an account and looked into the code for adding this functionality to the webhook.

Tasks completed:

- Migration to AR Foundation 3.0
- Testing of Tracked Image Manager in ARF 3.0
- Creating a new Azure SQL Database
- Creating a new StaffMember table
- Writing basic queries from Webhook to SQL database
- Writing Watson Dialog Node to trigger webhook to fetch from database
- Testing our full pipeline from Android → Watson → Webhook → Database and back

- Researching SMS messaging solutions (Twilio)

Are we on track?

We received feedback from our first submission which was generally positive, apart from the website which needs some major improvement. We felt encouraged that our TA thought we are generally making good progress with the project, so overall we feel the project is on track. There are still some issues with the new version of unity and ARF3 working with the Watson functionality, and we need to keep working on the website.

Problems to be resolved → Steps we intend to take

- Do we need to hard code every separate type of database query we want to make, if we want to add new tables for different types of information? → Experiment with different query types in the webhook; and with automating the writing of new types of queries.
- Sizing with tracked image manager. Sometimes objects placed on the tracked image appear too large and to small → Try experimenting with different scale values for the tracked image
- How to make the avatar rotate to face the camera? → Work out how to get the relative transform position of the camera and how to use Unity LookAt(); method in an AR context.

Plan for the next two weeks

Week one goals:

- Dillon to build UI elements, focusing on backing away messages and menu buttons
- Oliver to look into adding additional tables, starting with a "date" table to store events
- Lilly to start implementing additional AR functionality
    - getting the avatar to rotate to face the user
    - Getting text to show next to the avatar and also rotate to face the user

Week two goals:

- Add additional tables
- Incorporate UI functionality into main project