Progress Report: ARCHAE
Reconstructing Historical Sites with Augmented Reality

Rowan Mather, u2100495

November 2023

1 Introduction

Despite the burst of augmented reality tools emerging in all walks of life, it remains something of a novel concept to facilitate a user to place their own creations in their virtual world. In particular making such an application generally accessible to a technologically untrained audience is a focus of this project.

This report details the progress up until the Christmas period of ARCHAE: the Augmented Reality Constructed Historical Architecture Environment. This application will allow visitors to wander round sites with a mobile device and look how it used to be using augmented reality.

2 Research

Advantages of using mobile augmented reality in education are well-documented. The exploration aspect can help motivate students to learn, and such visualisations are a secondary communication method for those that struggle with the written word [1]. Furthermore, it grants access to tools and places which are otherwise restricted - be it equipment [2], or in this case that which no longer exists.

At a higher level, there are many use-cases within design and planning. Being able to visualise the proposed landscape eases collaborative approaches [3], as seen in technology such as Morpholio (see Figure 1). It can also mitigate construction errors.

The **Existing Work** section of the specification details several of these existing systems with similar features used as inspiration. It also lists their shortcomings when applied to this project's use-case. Most notably, the uniqueness of this task stems from its generality. For example, there exists an application for viewing The Colosseum in augmented reality (see Figure 2), but there is little work when it comes to world-wide and customisable features. This project in effect combines many of the most useful aspects of all these systems into a single immersive application.

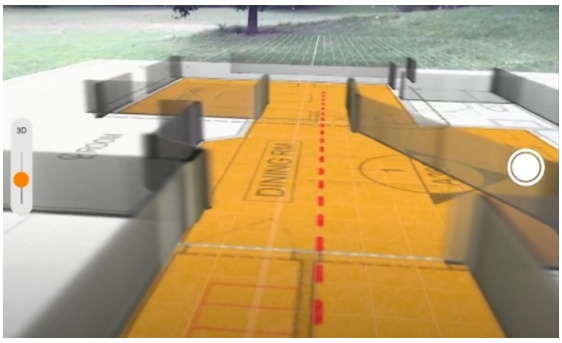

Figure 1: Morpholio Trace Sketch Walk [4]

Figure 2: RomeMVR - Time Window [5]

3 Progress Summary

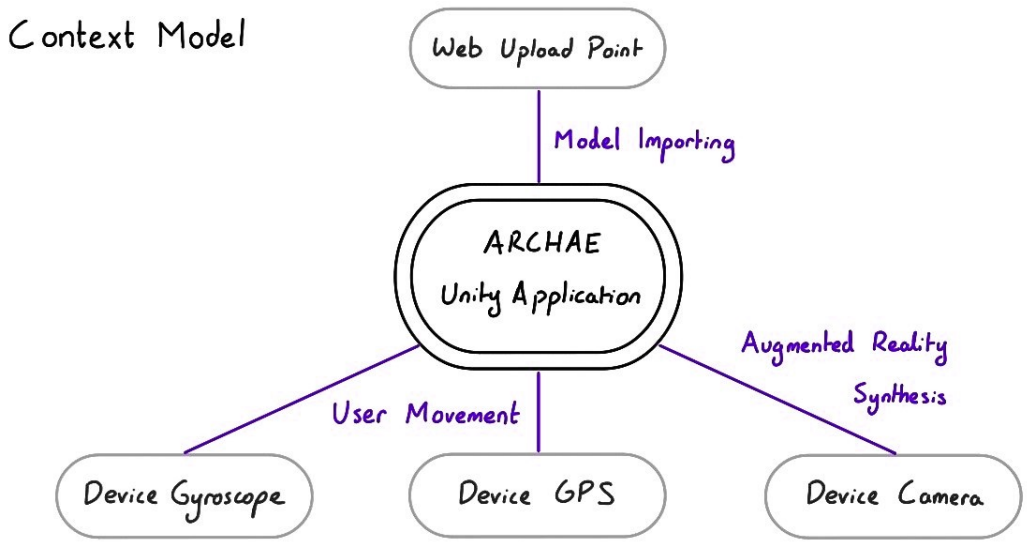

Figure 3: Modified context model of all involved systems.

Work can be divided quite neatly into three main categories:

- Model Importing: Uploading 3D models to the server, selecting them to download to the device and rendering them as objects in the scene.
- User Movement: Controlling the user's in-scene position/rotation and rendering objects relative to this, as well as collecting their real-world position/rotation.
- Augmented Reality Synthesis: Combining the scene visuals with the real-world camera input and adding additional information such as small tags.

These will be referred to in this summary and the **Next Steps**. The modified context model in Figure 3 outlines how they involve each system.

3.1 Model Importing

This area of work has been the focus for the initial phase of the project. In order to allow for general purpose use, the decision was made to connect the application to a web-server. Creators are enabled to upload their site models and specify their location and appearance. For prototyping purposes, the upload location is currently a GitHub repository (available here [6]). However, code is written generically so it is trivial to change the web URL to a purpose-built upload point.

Figure 4: Object hierarchy using a site prefab.

Whilst Unity is built for displaying 3D models and moving a camera around them, it is highly non-standard to import these models in real-time. Selecting and modifying the correct tool to achieve this was the first significant challenge of this project. Whilst testing importing from the local machine directly, it became apparent that this would bypass Unity's normal model pre-processing.

To simplify things, an intermediary asset working on .obj files (one of the most common object file types) was selected [7]. In order to perform more complex operations on the location and appearance of the model, the output of this asset was linked as a child to a custom prefab (template), as shown in Figrure 4.

Importing and positioning models is now trivially possible by uploading the files to the server and running the application. A small complete example is shown in Figure 5. The house is a test model created specifically for this purpose, and the monkey head is one of the default objects available in Blender (the 3D modelling software).

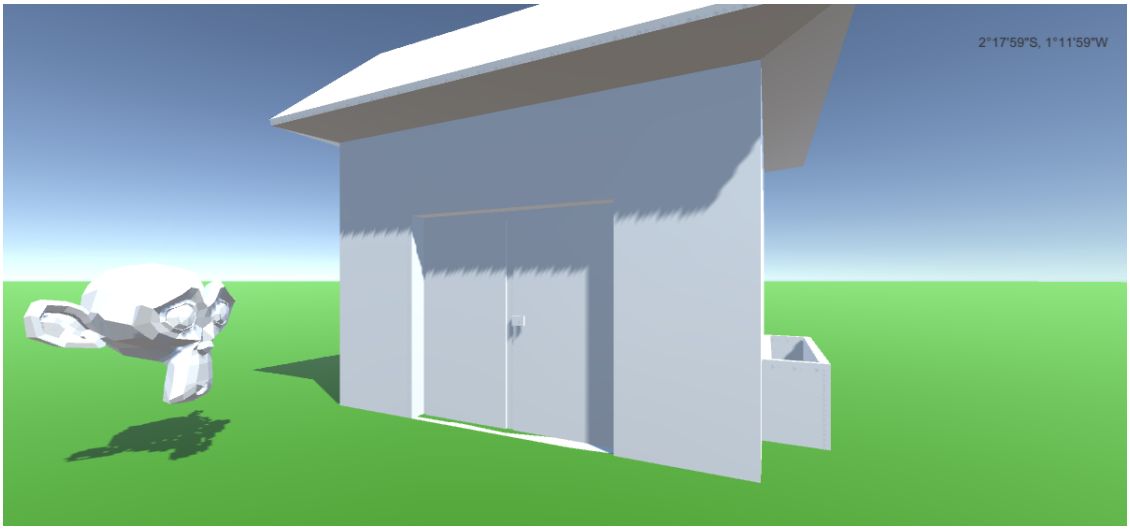

Figure 5: Current example scene in the Unity editor.

3.2 User Movement

The location of the user corresponds directly to a real-world GCS/GPS co-ordinate (Geographic coordinate system and its implementation, the Global Positioning System). For this there is a custom location type. At present, this can be configured manually with temporary buttons. Although it appears that the user through in the virtual world, the models move relative to the camera, each calculating their physical distance difference and scaling that to a Unity virtual co-ordinate. Models are also only displayed within a certain render distance in order not overload the number of vertices in the scene at once. One can see their location in the corner of the screen displayed in degrees, minutes and seconds (converted from the decimal degree form).

In order to fully look around the scene, a keyboard input is used to rotate the camera. One can move freely through the editor using the combination of these functions.

The next step was to gather live user location and rotation data and map this into the scene. Unity has a number of built-in tools to assist with external input from the device such as location (GPS [8]) and rotation (gyroscope [9]).

However, some of these tools are dated, which caused compatibility issues when testing on a mobile device. The gyroscope was registering as present and functional, but was returning only the identity rotation (0s). To verify it was a Unity issue, checks were performed first on the native device - this confirmed the functionality of the gyroscope.

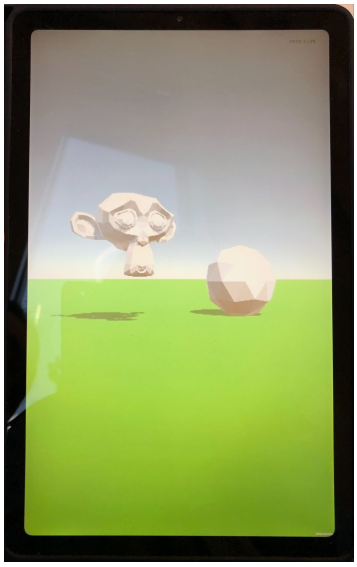

Figure 6: Testing the app locally.

A more recent input manager was installed and tested, as directed by this bug report [10]. The recommended 'attitude sensor' was not operational but a different gyroscope input stream was registering data. Although the new system should cover a great range of devices, both the new and legacy system will be used to ensure the application has the widest compatibility possible.

3.3 Augmented Reality Synthesis

The final area of work for this project will be working with the camera input to merge the virtual world with the real one. As of yet, little progress has been made towards this beyond identification of potential tools [11].

4 Next Steps

4.1 Importing Models

Since this has been the focus of term 1, only extension features remain to be implemented: texturing/colouring and additional metadata. The aim is to have a minimum viable product before completing this work, but improving the basic appearance of models will be subsequently of high priority.

4.2 User Movement

Mirroring the real GPS co-ordinates and camera orientation in the simulation requires completion. It will then be tested on a variety of locations and model sizes.

In addition, as shown in the design in Figure 7, the manual movement should be controllable on screen, rather than using custom Unity utility functions. Switching in and out of a live-exploration mode should enable or disable the angle scroll and arrow keys.

To improve the performance of the system it would be beneficial to consider alternative approaches to exclusively moving the models around the camera, as well as further rendering shortcuts. For example, the camera could move around a certain fixed area itself, before all the models are moved and re-rendered. In addition, rendering only the exterior of buildings where appropriate, or a lower vertex-count at a certain distance may speed up the response times. However, this would all be extension work and performance issues have been negligible with test data thus far.

4.3 Augmented Reality Synthesis

Work on this will follow completion of the live-movement. The virtual model display will be combined with the real-world camera input, so as to layer the historic structure over the present day one. The exact sizes of models will need to be considered carefully, along with the possible option of manual re-scaling.

It may be necessary to correct for GPS inaccuracy with some form of manual reconfiguration, but this remains to be seen.

As further extension, integration of the tagging and timeline metadata will be added, so that curators can label information on their display and demonstrate how the site has changed over different intervals.

5 Design

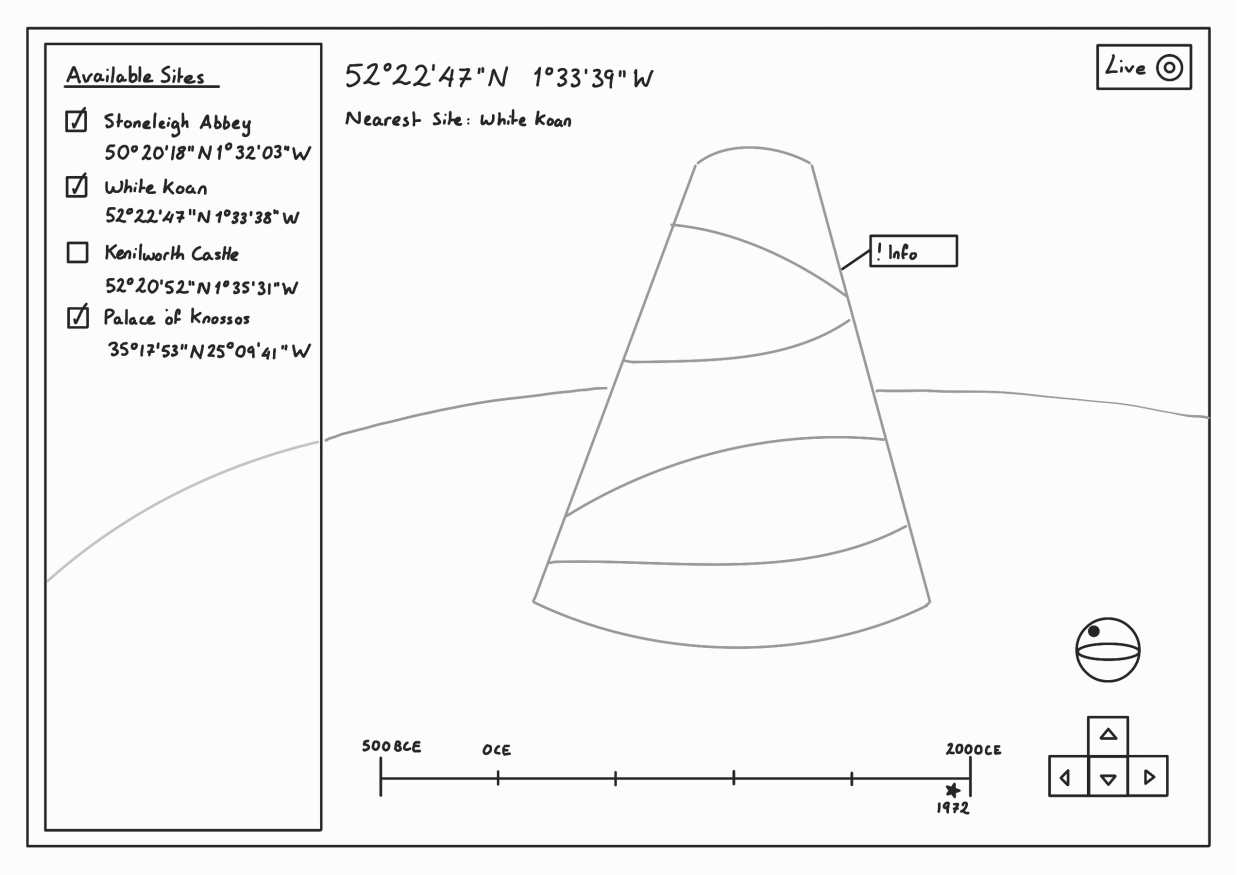

Figure 7: Mock-up of the Application User Interface

To illustrate the ideas outlined in the objectives, Figure 7 shows a mock-up of the main UI. Some modifications may be made, and the menu screen is yet to be designed, but the overall functionalities in reading order are:

- Selecting available sites.
- Viewing the location.
- Enabling/disabling live mode.
- Viewing tag information.
- Manually rotating the camera.
- Adjusting the timeline.
- Manually moving the camera.

6 Project Management

6.1 Supervision

Supervision of this project is undertaken by Dr Claire Rocks. The weekly meetings are ideal for keeping on track and receiving expert guidance.

6.2 Documentation

Alongside the history of the code being placed in a GitHub repository, an outline of every work session or meeting is written up in a google. A convenient aide-mémoire, this document has been written by referring to it. Since there are distinct sections in the construction of this project, it would not be difficult to lose track of progress were there not the notes.

Overall progress is documented in the gantt chart timeline (see Figure 9). As well as tracking the dates of tasks, the percentage completion is given.

6.3 Methodology

As detailed in section 5 of the specification, the methodology is most akin to the incremental standard method.

1. The first increment can be considered complete. This is exactly the 'must-have' objectives (detailed below).
2. The second increment is mostly complete. This is the live data section.
3. The sub-specification for the third increment of extension features will be drawn up fully upon completion of the second, although the content is mostly covered by the **Next Steps** section.

6.4 Objective Review

The complete list of objectives (detailed originally in Specification section 4: **Requirements Analysis and Objectives**) remain unchanged. For ease of reference, they are copied below.

Whilst none of the 'could-haves' are in place yet, this is by design. Development is progressing in layers of priority so completing anything beyond a 'must-have' or 'should-have' in term 1 would be unexpected and possibly unwise.

Work on the 'should-haves' has begun, specifically those in objective subset I and III. Although there are no further subdivisions of priority than the three tiers, these are crucial to the augmented reality aspect so should be implemented first.

Good progress has been made towards completion of the fundamental aspects of the project - indeed all 'must-haves' have already been implemented. Whilst there is much to be done to facilitate the entire intended use-case and improve the user experience, the application is ready to be built and used in its skeleton form at any time. Testing has begun on a native mobile device to ensure portability from the computer editor. The project is therefore in an excellent position to proceed to term 2.

Objective Set:

I. **User may upload a model to a remote server.**
   1. (M) Models will be in the wavefront (.obj) form.
   2. (C) Models can be uploaded in additional common formats e.g. .fbx, .dae, .blend
   3. (M) User may upload a texture and other metadata to pair with the model.
   4. (S) A subset of the server models can be externally selected to be available in app.
   5. (C) Users cannot upload without submitting a review request.

II. **User may download models from the server into their local app.**
   1. (M) Models can be imported from the server in their pure object form.
   2. (S) They can be coloured/textured according to an accompanying texture file (.mtl).
   3. (S) Users can select a subset of the server models to import locally.

4. (S) Models will render with their textures and metadata.
5. (M) Models will render with reasonable latency, generally a maximum of 2 seconds per model.

III. **User will be able to set their location and move around the models.**

1. (M) User has a manually configurable geo-location (latitude & longitude).
2. (M) User has a manually configurable orientation.
3. (M) Models can be placed in a geo-location relative to the user.
4. (S) Models will render only when within a certain distance to the user.
5. (S) User can change the render distance.
6. (C) User can change the scaling of the whole scene.
7. (S) User location can be set to their actual mobile device's geo-location.
8. (S) User can move freely while the app tracks their geo-location.
9. (S) Compass direction can be set to that of device.
10. (S) Tilt can be set to the spirit level of device.
11. (S) User can switch between manual and device modes.
12. (S) Add camera input as background for the scene.
13. (C) Have a slider for model transparency.

IV. **Additional information about the sites will be accessible.**

1. (C) Models can be tagged with information in a specific relative location to the model.
2. (C) Tags can be clicked to view further information/photos.
3. (C) Tags can be shown/hidden in scene.
4. (C) A date (range) will be in the corner corresponding to the age of the site you are viewing.
5. (C) A timeline scroll will be at the bottom of the screen.
6. (C) Models can be attached to a specific time period and will only be visible when this is selected.

V. **The app will be intuitive and guided.**

1. (C) Include a menu with an app tutorial.
2. (C) Most information can be hidden so the user is not overloaded.
3. (S) The default view will be a simple drop-down list of available models and device mode.

6.5 Timeline Review

For the most part, work has been completed in line with the initial plan.

There was a slight slow start in the setting up of the Unity environment since most of this work was completed over the Summer in a testing Unity project. Porting progress over into a clean file was appropriate once the basic system was fully decided upon, but this was time-consuming.

The section on real-world data is ahead of schedule, despite the aforementioned challenges. The difficulty of this section has in fact been a motivation to start implementation sooner. Therefore in the worst case going forward, the initially planned deadline will simply be met, and in the best case, work on extension features will also be started early.

One issue which caused delay beginning this real-world work was a hardware problem: there was not enough memory available on the main development device to install the necessary tools to build for mobile devices.

Having spent a day removing and transferring surplus files and applications, there should now be sufficient space for the remainder of the project. However, in the unlikely event that it takes up a further 8GB, work may have to be completed using an external memory device. This would cause further latency in the already slow Unity engine, but make completion possible.

Two timelines are listed below, namely that in week 6, (before the modification of next term's work) and the updated timeline with more detailed tasks for next term. The categories predominantly correspond to the methodological increments, but there is some extra sub-division, and the addition of all non-application development tasks.

Key:

- Grey: scheduled work period
- Dark green: work on schedule
- Pale green: work off schedule

7 Ethics

No ethical consent is required.

Figure 8: Week 6 Gantt Chart

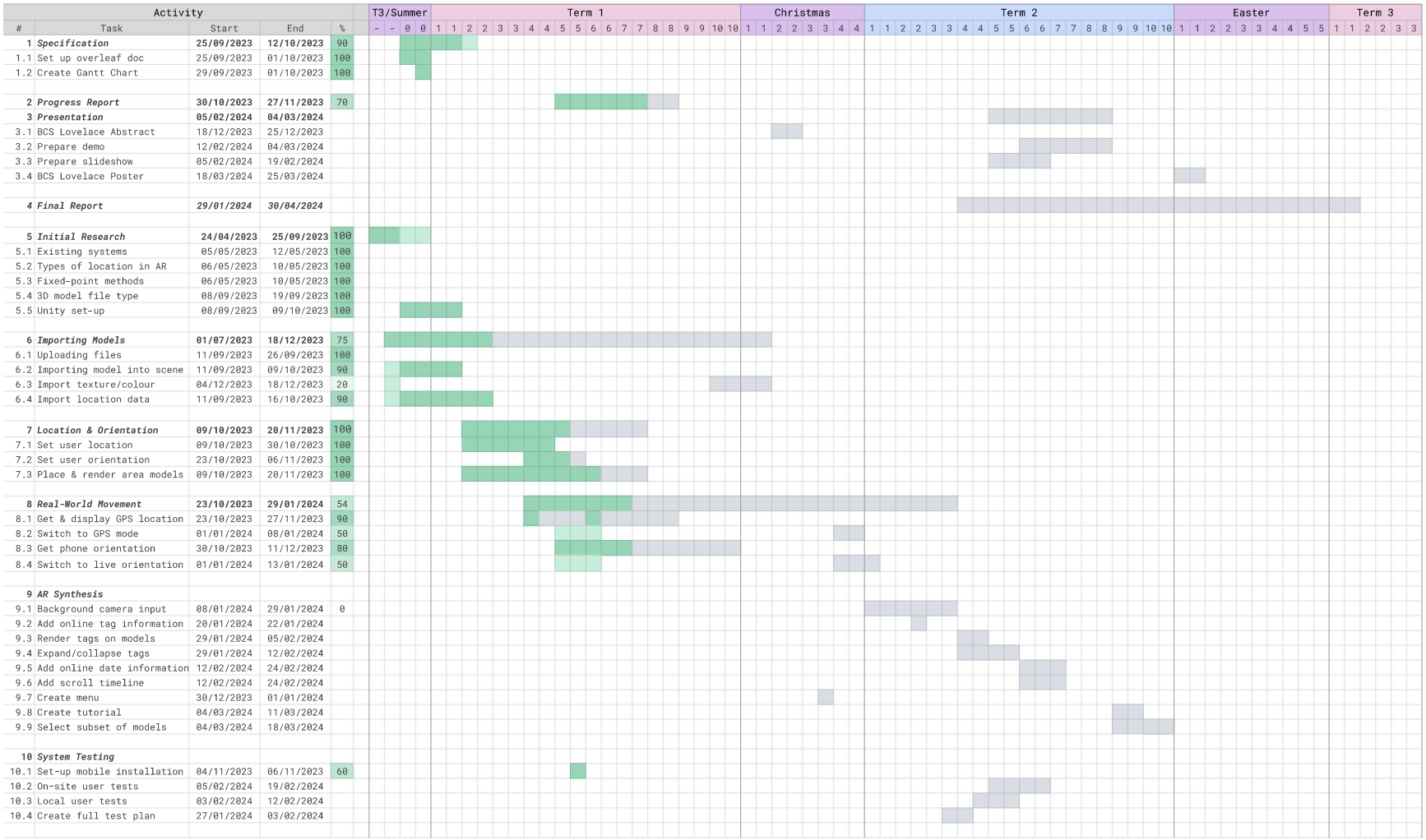

Figure 9: Updated Term 2 Gantt Chart

References

[1] M. Billinghurst and A. Duenser, "Augmented reality in the classroom," *Computer*, vol. 45, no. 7, p. 62, 2012.

[2] M. Bower, C. Howe, N. McCredie, A. Robinson, and D. Grover, "Augmented reality in education – cases, places and potentials," *Educational Media International*, vol. 51, no. 1, pp. 1–15, 2014.

[3] J. L. S. P. Carlos Carbonell Carrera and J. de la Torre Cantero, "Teaching with ar as a tool for relief visualization: usability and motivation study," *International Research in Geographical and Environmental Education*, vol. 27, no. 1, p. 72, 2018.

[4] Morpholio, "Morpholio trace." `https://morpholioapps.com/trace/`, 2022. Last accessed on 07.10.2023.

[5] Altair4 Multimedia, "Rome mvr time window." `https://altair4multimedia.it/lavori/rome-mvr-time-window/`, 2011. Last accessed on 07.10.2023.

[6] Rowan Mather, "Archae repository." `https://github.com/Rowan-Mather/csproject2023/tree/sites`, 2023. Last accessed on 25.11.2023.

[7] Dummiesman, "Runtime obj importer." `https://assetstore.unity.com/packages/tools/modeling/runtime-obj-importer-49547`, 2019. Last accessed on 25.11.2023.

[8] Unity, "Locationservice." `https://docs.unity3d.com/ScriptReference/LocationService.html`, 2022. Last accessed on 25.11.2023.

[9] Unity, "Gyroscope." `https://docs.unity3d.com/ScriptReference/Gyroscope.html`, 2022. Last accessed on 25.11.2023.

[10] Unity, "[android] gyroscope functionality doesn't respond in the player when using a tablet." `https://issuetracker.unity3d.com/issues/android-gyroscope-functionality-doesnt-respond-in-the-player-when-using-a-tablet`, 2023. Last accessed on 25.11.2023.

[11] Unity, "Class arinputmanager." `https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/api/UnityEngine.XR.ARFoundation.ARInputManager.html`, 2022. Last accessed on 25.11.2023.