

Virtual Reality for Space Science (PF2-072)

D2: Final Report

Jessica Castle, Rob Penn, Jack Tyler,
Dr. Vanessa Wanick, and Dr. Alexander Wittig

University of Southampton

April 5th, 2018

Table of Contents

Table of Contents	2
1 Project Summary	4
Highlights	4
Impact	5
Milestones & Payments	5
Progress RAG Analysis	6
2 State of the Art	7
Scientific Data Visualization	7
User Interface Design	9
Pattern 1 - cursor navigation (2D)	10
Pattern 2 - hand navigation (3D)	10
Pattern 3 - cursor and hand navigation	11
Pattern 4 - HUD (head-up display)	11
Pattern 5 - Menu navigation	11
Pattern 6 - Field of View (FoV)	11
Pattern 7 - Curved 3D UI	12
Pattern 8 - Sound	12
3 Device selection & trade-off	17
Device Classes	17
VR devices	18
AR devices	18
MR devices	18
Mobile devices	19
Device Selection	19
VR device: Oculus Rift	19
AR Device: Magic Leap 1	19
MR device: Lenovo Explorer	20
Mobile device: Oculus Go	20
Development Environment	20
OpenGL, OpenVR, DirectX	21
Unreal4	21
Unity 3D	22
Selected Development Stack	22
4 Space Science Use Cases	24

Scenario 1: Orbit visualization for Mission Design & Analysis (“Space AR”)	25
Scenario 2: Earth Observation Data (“African Drought Monitor”)	29
Spin-out use cases	31
Bubble Formation in Complex Fluids	31
Computational Fluid Dynamics (CFD)	33
Use Case Selection	34
5 Prototype Development and Implementation	35
Prototype Definition	35
User Interface Design and Development	35
Orbit Interchange File Format	37
Future extensions	40
Example Data Sets	40
The Solar System	41
Low Earth Satellite Orbits	42
6 Prototype Evaluation and Testing	44
Evaluation of the Prototype	44
Testing	46
What worked	46
What changed	46
What can be improved	47
Platform specific results	47
5 Conclusions & Future Work	49
Impact of the work	49
The way forward	50
References	52
Appendix	58
A1 - Design style guide	58
A2 - Device Trade-off Tables	64
A3 - Codes	69

1 Project Summary

The project finished on schedule and was carried out successfully according to the project plan.

In the course of the project the team carried out the following activities:

1. Literature Review to determine the state of the art
2. Device Selection and Trade-off
3. Space Science use case analysis and selection
4. Prototype development and implementation
5. Prototype evaluation and testing

Highlights

Successful development of the prototype orbit visualization tool, which is capable of displaying custom orbital data read from a JSON file on a variety of VR and AR devices including the Oculus Rift and Microsoft HoloLens.

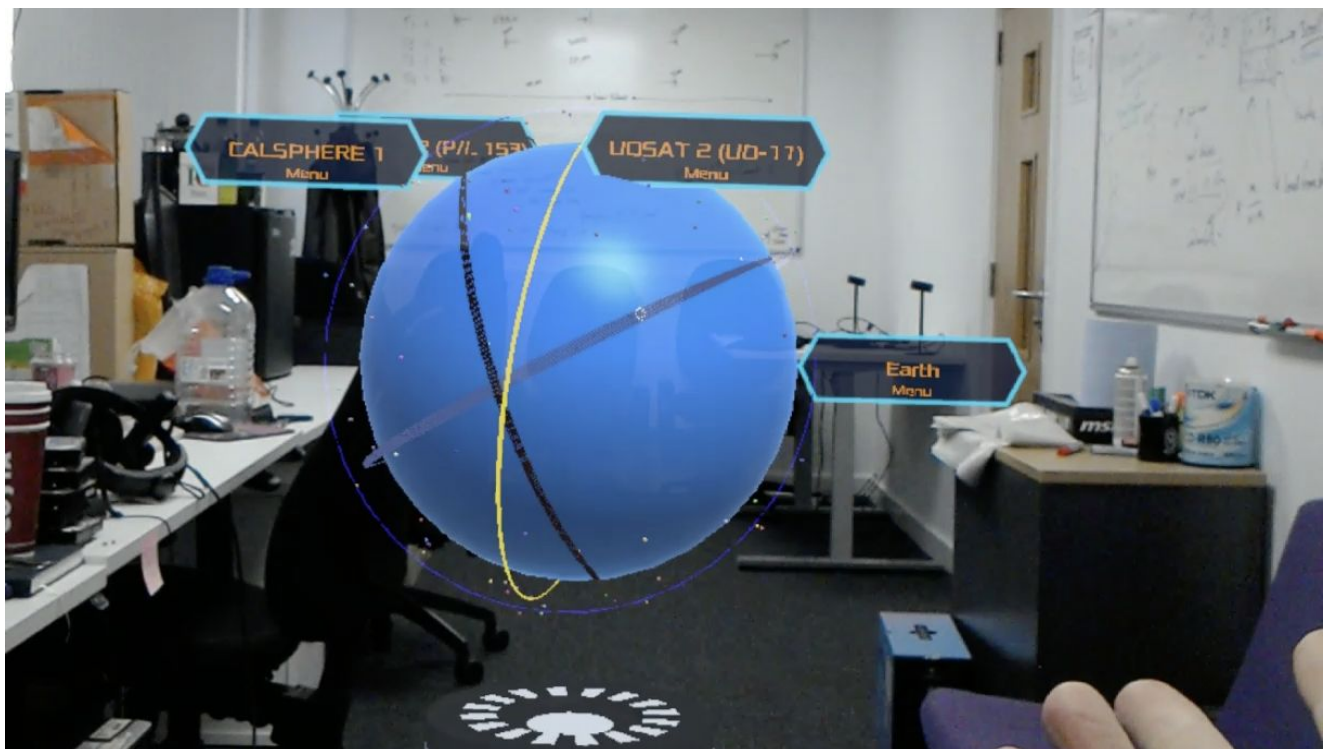


Figure 1: HoloLens visualization of current Low Earth orbiting spacecraft.

Impact

The project is generating impact through the following presentation and publications:

- 4th international workshop on key topics in orbit propagation applied to space situational awareness (KePassa 2019), Logroño, Spain. April 24 - 26 2019.
<https://kepassa.unirioja.es/>
Accepted oral presentation by Dr. Alexander Wittig.
- Workshop at Colegio Pedro II (Rio de Janeiro, Brazil) on techniques and development of holograms and VR/AR technology in education (workshop given by Dr. Vanissa Wanick to tutors and designers). April 1 - 2 2019.
- Astronautics Seminar at the University of Southampton. April 3 2019.
Oral presentation by Dr. Alexander Wittig and Rob Penn.
- Dr. Vanissa Wanick and Jess Castle are working on a full paper to the XIII International Conference on Graphics Engineering for Arts & Design that will be held in Rio de Janeiro in September 2019

Milestones & Payments

This report constitutes the final deliverable for milestone M3. Once accepted by UKSA the last invoice will be issued and sent to UKSA by University of Southampton finance.

Table 1: Payment plan

Milestone	Project Month (end of)	Payment	Description
M1	1	25 % (£ 18,614.20)	Selection of hardware and ordering of equipment
M2	2	55 % (£ 40,951.24)	Intermediate progress report (including prototype definition)
M3	6	20 % (£ 14,891.36)	Final report end of project

Completed & Paid	Completed	Late	Ongoing
------------------	-----------	------	---------

Progress RAG Analysis

Table 2: RAG Analysis

WBS	Title	Progress	Status (Red/Amber/Green)				Comments Mitigation/Correction Plan
			Overall	Technical	Schedule	Resource	
WP1	State of the art	<i>100%</i>					
WP2	Trade-off and Specifications	<i>100%</i>					
WP3	Implementation	<i>100%</i>					
WP4	Testing & Analysis	<i>100%</i>					
Overall Project		<i>100%</i>					The project has been completed successfully

2 State of the Art

This section introduces the current state of the art derived from an extensive literature review performed as part of WP 1. The findings from this section directly informed the design of the proposed use cases later in this report.

Scientific Data Visualization

While off-the-shelf visualisation softwares, such as Virtualities, do exist, scientific visualisations often require more complex analysis pipelines than is made available by commercial software. Thus, scientific visualisations are often made ‘from scratch’, and hard-coded for one particular dataset.

Much work has been done on the use of AR and VR in scientific visualisation, derived largely from efforts driven by the need for flight simulation and training methods, such as (Furness, T. A., 1988)’s early work on military flight simulators.

Indeed, early VR systems were used as spatial ability training tools (Dünser, Steinbügl, Kaufmann et. al., 2006; Shufelt, J. W. Jr., 2006), and were found to be a valid and effective method of training in the medical sector (Verdaasdonk et. al., 2008), reducing errors during certain surgeries by up to 66% (Ahlberg, Enochsson, Lars, 2007). Use of VR and AR in the education sector is ongoing to this day (Andrienko, Keim, MacEachren, & Wrobel, 2011; Gavish, et al., 2015).

VR has found modern use in scientific applications where the data is high-dimensional, or too abstract to understand efficiently using traditional computer visualisations (Elsayed et. al., 2013). For example, (Helbig et. al., 2014) used VR to visualise atmospheric cloud data, to make more intuitive the ‘correlations between variables like wind vectors, humidity and cloud coverage’, to ‘detect inconsistencies in the data’, and to compare data from multiple models. Indeed, interviews conducted as part of the study rated the VR visualisation as “highly supportive for analysing large heterogeneous datasets”, and (Helbig et. al., 2014) cite that VR allows their 2D, 3D and 4D data to be visualised more readily, and that users are able to move through the data to allow ‘flexible, task-specific analysis.’

(De Ridder, Jung, Huang et. al., 2015) used VR to better understand complex fMRI visualisation, as there are a ‘number of challenges’ that prevent fMRI visualisation from being fully utilised in traditional 2D visualisations, including ‘understanding which structures are consistent and different between individuals and across the population’. Using interactive analytics, the authors suggest that VR/AR can potentially overcome these challenges by ‘allowing for a reduction in visual clutter’, and by allowing users to navigate the data in a ‘natural’ manner that reduces cognitive load.

In general, the scientific sector has noted VR to provide the user with a deeper understanding of data, and to increase the amount of useful information that can be extracted from a dataset (Bach, et al., 2016; Keim, Thomas, 2008).

AR, while a more recent development, has also quickly found applications in a variety of scientific areas. Owing to the reduced immersion inherent to AR devices, a trend has emerged of using AR for uses where it is important to retain context about the user's surroundings (Azuma, 1997).

AR has been found particularly useful in GIS applications, where datasets are often complex, and require some form of context about the user's surroundings. (Zollmann et. al., 2012) performed in-situ visualisation of Civil Engineering data on a planned building site, allowing for the live viewing of planned piping routines and building layouts. When compared to traditional visualisations that often rely on super-imposing 2D or 3D image renders of the building site and piping layout onto sample images, the authors found that AR could simplify such tasks by presenting an integrated view of the geospatial models in 3D at the building site itself, and 'provide immediate feedback to the user and intuitive ways for data capturing, data correction and surveying'.

While issues arose in the extensive pre-preparation of the visualisation - particularly with correctly defining the locations of the features, which have to be known *a priori* - a later publication by the same leading author, (Zollmann et. al., 2017), demonstrated situated visualisation. Situated visualisation uses more recent devices, equipped with GPS receivers, to overlay geotagged data over physical features based on location and viewpoint alone, and can interact with a live geodata server for real-time data interaction and synchronisation.

Despite the lower level of immersion, detailed visualisations have still been performed using AR, such as (Kim et. al., 2018)'s study of aerodynamics simulation data. By using a method similar to the situated visualisation above, an AR headset with wireless connectivity was interfaced with a high-performance compute node running the standard Computational Fluid Dynamics (CFD) software OpenFOAM. The user may then select the location and context of the flow source to be visualised - in this case the flow from an air purification system - and view the resulting flow in real-time from the CFD solver via a Unity display engine (Figure 1). It was noted that the use of environment-based tracking and visualisation here allows the user to 'review CFD results accurately and in immersive fashion, even in a large and complex space', and particularly allows the user to immediately view the context of the airflow in the 'real-world', rather than the out-of-context raw data typically visualised from traditional solvers.

(Sun, Han & Ma, 2018) found that many users prefer to interact with virtual objects in 'figure-' and 'table-spaces'; that is, the where the visualisation takes the size of e.g. a mobile telephone, or where



Figure 2: Kim et. al., 2018’s simulation of real-time aerodynamic flow simulation from an air purifier. The results were generated using a high-performance compute node, and downloaded and visualised in real-time using the Unity game engine. The authors praised the visualisation for allowing them the review CFD results accurately.

the visualisation is the size that would fit comfortably on a desk. Many studies of VR and AR have also criticised the lack of ability to change datasets mid-visualisation (Ghadirian & Bishop, 2008).

Our research indicates that there exists a significant gap in the literature in the use of AR to augment a desk-based setting, rather than outdoor physical surroundings. The ability of a user to continue to interact with their computer with traditional input methods whilst updating the data being visualised may significantly improve productivity. Seamless access to established user- or domain-specific software (such as Matlab or Python scripts), while at the same time maintaining a virtual view of their data, can help overcome the problem of static data visualization and the limited interactions afforded by current VR/AR controllers.

User Interface Design

This section summarises the current 3D UI patterns and research guidelines that can be used in the context of the project. Interface Design is about intermediating the interaction between humans and computers, usually through representations and commands (e.g. menus, windows, icons, pointers, buttons, etc.) (Dix et al., 2004). In Virtual Reality (VR) environments the UI is inspired by natural interfaces that would eventually mimic some aspects of the real world. Research has been conducted in order to build the bridge between the real and the digital through interfaces, creating the Tangible UI (TUI) and radical atoms (Ishii and Ulmer, 1997; Ishii 2008; Ishii et al., 2012). Mentioned in (Wanick, Xavier and Ekmekcioglu, 2018), opportunities for UI Design in VR environments may also include Brain-Computer Interfaces (BCI), which is the interaction of elements using brain data (Pike et al., 2016). Yet, for the context of this project, it is possible that the current research in the area might not be completely

accessible for data visualisation purposes and for this reason we focus on the understanding of current research in 3D UI. A 3D User Interface (UI) is an UI that involves interactions in a 3D environment (LaViola et al., 2017).

Bowman (n.d.) provided an extremely relevant overview of current research in 3D User Interfaces (3DUIs) and applications. According to his review, usual GUI is composed of Windows, Icons, Menus and Pointers (WIMP). We have seen a shift from GUI towards a natural user interface (NUI), which tends to replicate “real” or ”natural” interactions through metaphors. So far, as mentioned by a talk given to Microsoft Research in 2016¹, Wolfgang Stuerzlinger cites that current 3D UI should be aiming towards the use of 6DoF and that main UIs today are still limited to fewer degrees of freedom. For this project, it is important to mention at least 2 aspects: the interaction **utilising a cursor** (which is mainly 2D) and a **representation of a hand** (which replicates the hand in 3D). These two forms of interaction might have an impact on the project’s functionality, usability and manipulation of objects. For instance, if the users can see their hands, it might have more precision, particularly in 6DoF is possible. To mention, *Unity* has also published in their page² a summary of types of UI in VR environments: non-diegetic, spatial and diegetic. Also as found in the public repository (GitHub), on this link³, there are plenty of examples on interactions with constellations, which may be useful for the purpose of Scenario 1 (Orbit). These were also summarised in this section. The patterns and possibilities are described below.

Pattern 1 - cursor navigation (2D)

Description: today, most of our UIs limit us to “cursor” style, similar to website navigation. In VR, this can be used as a point (laser point), similar to a 2D cursor. Thus, using controllers, users might be able to point towards a button and interact with it (this could also work with gaze pointer from the headset). This can be achieved through the use of controllers or orientation-tracking sensors.

Pattern 2 - hand navigation (3D)

Description: hand navigation is a way to see your own hands within the VR environment. This may lead to a certain “precision” or perhaps a sense of control, as users can “see” their own hands and that would replicate the “real” world. By using any type of interaction that contains users’ hands (even in VR and AR), it might be easier to convey interactions that contain gestures since it would be more “natural”.

¹ <https://www.youtube.com/watch?v=BAk9B2UNhdM>

² <https://unity3d.com/learn/tutorials/topics/virtual-reality/user-interfaces-vr>

³ <https://gist.github.com/omgmog/d2371c861b2232ac2c445ee34af8edb6>

Gesture tracking plugins can be found in both Unity and Unreal engines. This youtube video⁴ shows how this interface works in Unreal. Orientation-tracking sensors work on this pattern.

Pattern 3 - cursor and hand navigation

Description: another way is mixing both the cursor metaphor and hand navigation with controller visualisation. That could enhance the precision of the points (cursor or point), using possibly a mix with gaze pointer and the visibility of controllers in the VR environment. For AR, the hands will be already part of the interaction. Thus, the mix of a cursor with hand gestures might be more appropriate for the purpose of this project, mixing feedback from the cursor interaction and gestures. For that, we can use controllers with a mix of sensors for orientation-tracking movements (e.g. track head movement).

Pattern 4 - HUD (head-up display)

Description: usually a non-diegetic strategy (i.e. not part of the virtual world), this is similar to a 2D menu and usually displayed “transparent” in video games e.g. for health visualisation. The problem of HUD is the distance to the user’s eye and how many menus to navigate. This may follow the user’s head display or not, depending on the desired experience.

Pattern 5 - Menu navigation

Description: it is possible to utilise floating menu navigations that can be accessed through the controllers. The menu navigation can be floating in front of the user or it can follow their PoV. However, it is a matter of usability to understand the best distance. Google (2017) has developed a metric (distance-independent millimetre, dmm) in order to make sure that users would be able to interact with the interface in an appropriate manner.

Pattern 6 - Field of View (FoV)

Description: Head Mounted Displays (HMDs) provide a 360-degree interaction in the field of view (FoV). This could be used in other to get a better sense of 6 DoF in virtual environments. It is basically the movement conveyed by the headset. It is important to consider this within the virtual experience since it is one of the aspects that differentiate the current experience (e.g. Desktop, 2D) from the proposed one (the visualisation in VR/AR).

⁴ <https://www.youtube.com/watch?v=0D1Iy3Nprik>

Pattern 7 - Curved 3D UI

Description: 3D spatial user interface can be curved. There is a pattern developed in 2001⁵ that shows the use of a sphere as the main UI in 3D spaces. The curved UI is a current use in many 3D environments since it fits the angles and the DoF available for the user PoV. However, this is limited by the small FoV of AR devices.

Pattern 8 - Sound

Description: Sound can be used as a way to give feedback to the user. (Fictum, 2017) states that 50% of the VR experience is composed by sounds and that these sounds should come from headphones, in order to make the user feel more immersed in the environment.

Besides all the patterns here mentioned and collated, there are more guidelines to consider. (LaViola et al., 2017) mention that the main strategies to convey a good user experience (UX) in 3D environments is to use real-world metaphors to guide user interaction, together with physics-based constraints for precision and hybrid interactions (such as 3D and 2D, cursors and gestures, etc.). Thus, the UI should allow users to have a sense of the real-world, being seamless and intuitive. At the same time, being 3D means that it is crucial to understand the influence of visual perception, 3D depth cues, spatial sound perception, presence (Slater et al., 1994) and the interaction of humans with space, including spatial abilities and sense of surroundings (LaViola et al. 2017). Therefore, the 3D UI should be adapted to the 3D space. For instance, by curving the UI, users could have a wider overview of the menu in 180-degrees (Awadhi et al., 2018).

In a recent study, (Bach et al., 2018) have analysed the effectiveness of different tasks in different environments with regards to visualisation of scientific data using HoloLens, Tablet and Desktop. By doing so, (Bach et al., 2018) suggest that the most effective scenario is bridging the visualisation. That is, some tasks might be better performed in specific scenarios. For instance, it might be more useful for the user to input data by using a keyboard instead of using gestures. Also, it is possible that users in general might feel more familiar with the Desktop interactions and therefore, it might take them some time to learn new interactions. The study was very significant in order to inform our project since researchers have also found that a long interaction using HoloLens (40 min) was not an issue for the users (they were not feeling tired). Also mentioned in (Bach et al., 2018)'s work was that HoloLens can provide

⁵ <https://patents.google.com/patent/US7013435B2/en>

embodiment, immersion and engagement, in which users could “feel” the data. One aspect not studied in this research was the understanding of data. There was no evaluation about cognition or decision-making based on data analysis. However, for the purpose of the present project, the research showed that tasks using HoloLens for scientific data visualisation could be undertaken comfortably. In fact, it is the nature of the task that will guide the design of the UI. As (Fictum, 2017) also mention, we need to distinguish between a pure entertainment experience (e.g. a VR game) and the visualisation tasks for scientific data, which is not “entertaining” by nature. Also, (Bach et al., 2018) mentioned that hybrid interactions might have a more effective use for visualisations and in fact, this is in line with the concept of *Blended Interaction*, introduced by (Jetter et al., 2014). Blended interaction is about merging analogies from the real world in the digital world, until they become a pattern (like the floppy disk as a saving button).

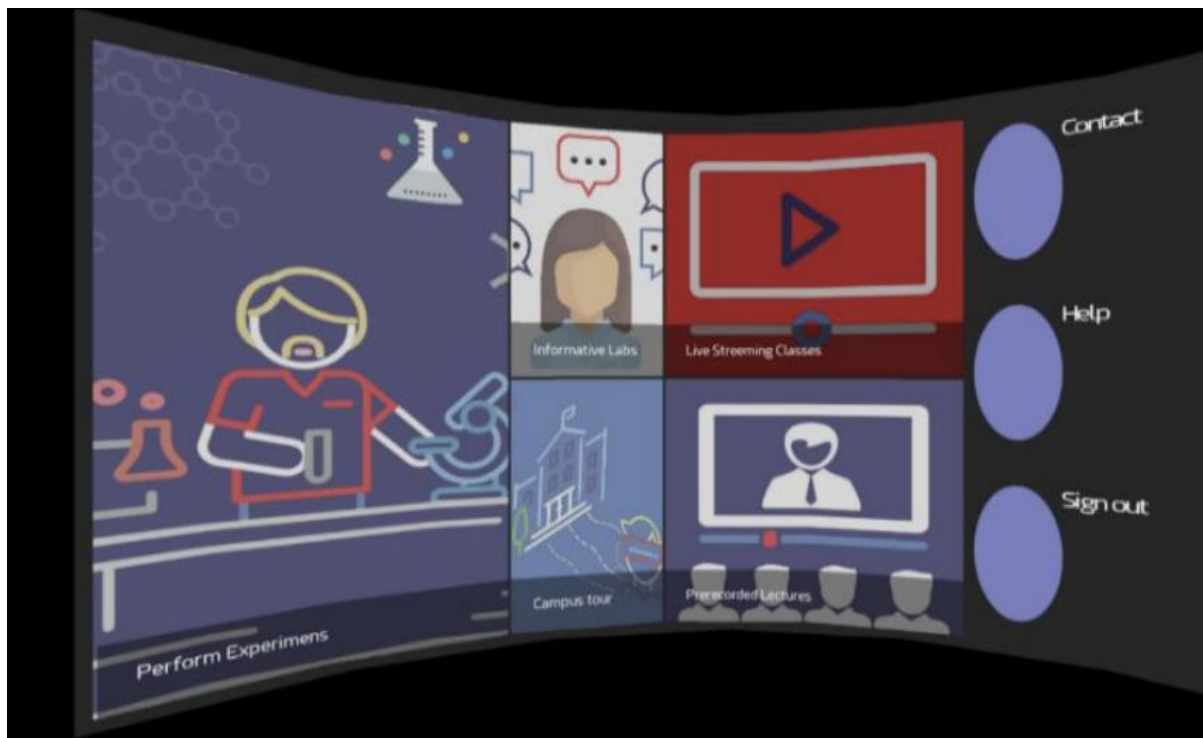


Figure 3: The Curved UI menu and possible interactions (Awadhi et al, 2018)

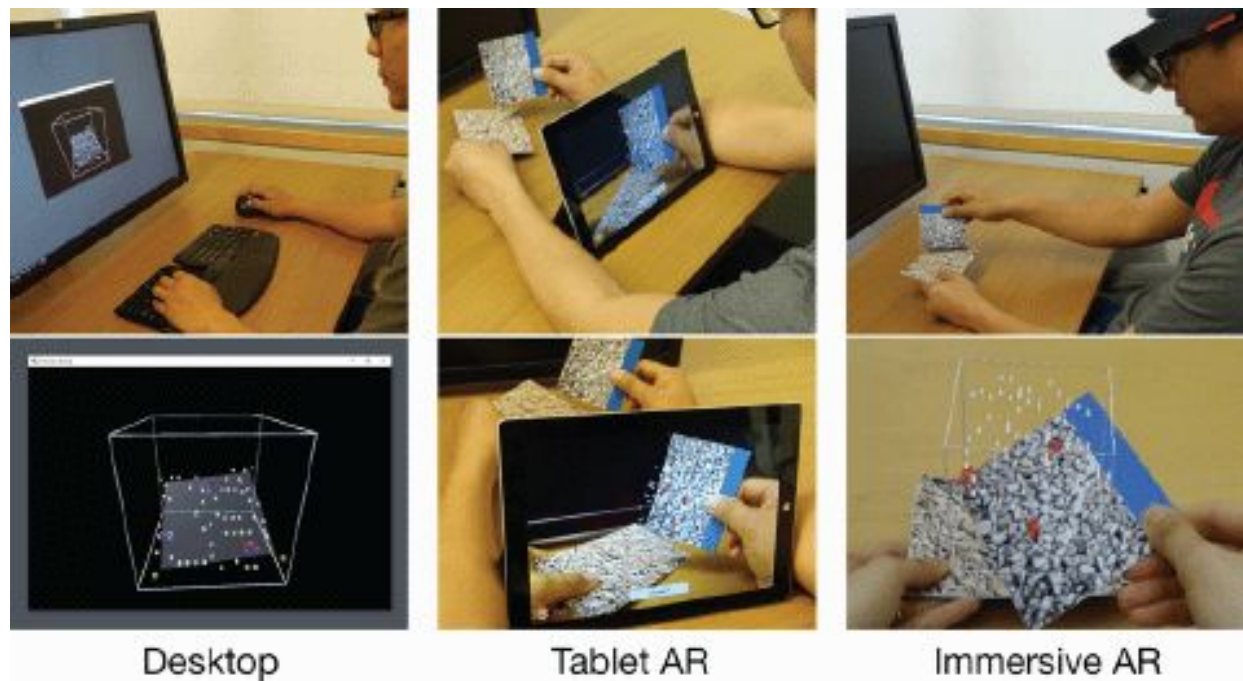


Figure 4: The user set-up (top) and the interface from the perspective of the user (bottom) from (Bach et al., 2018).

It is expected that 3D environments would have at least 3 aspects to consider for data visualisation: stereoscopy (3D perception), degrees-of-freedom (possible interactions in “space”) and proximity (e.g. embodiment or “touching” the data) (Bach et al., 2018). This is crucial for this project since we expect to include those 3 areas in each of our possible scenarios. Another aspect to consider in 3D UIs is the quality of experience (QoE). If considering UX elements and the QoE, (LaViola et al., 2017) also state that there are 3 key areas to consider: presence, cybersickness and interaction fidelity. In fact, usability is still a key challenge in 3D UIs in immersive environments. As mentioned in Bowman’s review, there is still no standard 3D UI guideline and most of the usability guidelines follow heuristics applied in usual graphic user interface (GUI). Thus, research in the area is still maturing. However, it is worth mentioning that according to Bowman’s review, it is crucial to consider the balance between the DoF and the task. Another aspect to consider is the technique of “travel” or point and teleportation (Bozgeyikli et al., 2016). This technique allows users to move around the VR environment and it might be useful for intuitive interactions. However it may be difficult due to cybersickness. One way to solve this could be by introducing the user to the experience in “small” steps, so the user gets gradually immersed in the environment (Bromley, 2017). Other challenges for 3D UIs mentioned by Wolfgang Stuerzlinger from

Microsoft Research are precision and latency. Stuerzlinger mentions that in the real world we use our hands to pick up a simple object and we look at our hands to do that. The same should happen in VR. The other aspect is latency; if there is a delay on the feedback, the interaction might not be intuitive and that could break the immersion. Thus, as much as the UI could be mainly visual, it also includes a combination of aspects such as sound, haptics, the limitations from the device and the user task.

It is worth mentioning that with regards of “good” UI in VR/AR environments, there are many ways to measure the QoE. (Fictum, 2017) suggests that a UX test should include 7 simple steps:

1. the determination of the nature of the experience (what is the purpose?)
2. the simulator sickness test (a questionnaire, developed by (Kennedy et al., 1993))
3. the experience (navigation and controls)
4. cognitive map (ability to understand concepts during the experience, usually measured with time)
5. advanced tracking (possibility to use biometric data such as heart rate, temperature)
6. qualitative questions
7. emotional feedback
8. whether the user achieved the ultimate goal of the application (and if the user would use it again).

Other metrics of QoE worth mentioning are perceived presence, perceived usability and emotions (Hupont et al., 2015). For instance, there is a proportional relationship between perceived immersion and perceived usability.

To conclude, our review shows that just like the field of visualisation in immersive environments, the area of 3D UI is still under development. In order to promote the best experience in the context of this project, we will consider the points suggested in previous research (best practices), such as the use of metaphors in 3D UIs, the hybrid interactions (utilising gestures with cursor when applicable), simplicity, use of sound and haptics (when applicable) and most importantly ensure that users can have the minimum interaction with precision. That is, there is no latency on feedback to the user and users can perform a single task without any interruptions. In order to make sure this is in place, we will also conduct a UX study once we have the first prototypes done and collect user feedback (using Fictum’s steps) and other metrics of immersion/presence, such as a version of Game Experience Questionnaire (GEQ) (IJsselsteijn, et al., 2008) applied to our prototype and the MEC Spatial Presence Questionnaire (SPQ) (Vorderer et al., 2004). Yet, although our main goal is to ensure that the user can perform a single task, we expect to develop for the future a more inclusive experience, considering accessibility issues.

An additional concern in the development of any modern UI is inclusivity and accessibility to all users. Since the field is still maturing, there is little research of VR/AR UIs for people with visual or hearing impairments. For instance, Magic Leap is the only device today that allows users to order prescription glasses with it. The other goggles do not offer this function, although for some of them it is possible to wear glasses on together with the headset; but that might cause discomfort.

3 Device selection & trade-off

In order to perform a trade-off analysis for device selection, the following two key requirements have been identified from the project description:

- Visualisation capability for potentially large and complex scientific datasets
- User immersion and interface to allow complex user interactions

Visualisation capabilities are directly linked to the hardware of each device. Specifically, the available mass memory on the device directly limits how much data can be made available to the user to load and explore interactively. The available graphics memory on the device, along with computational power of the graphics processing unit, restrict the amount of data that can be visualized at once.

The device's main and graphics processing units together play a key role in ensuring a positive and fluent user experience. In virtual reality, achieving a high frame rate for the visualization is the most important factor. Low framerates lead to stuttering, tearing, and artifacts which in turn can lead to motion sickness and visual discomfort of the user.

User immersion and user interface, on the other hand, determine how much the user can interact with and change the data being visualized. Scientific analysis typically extends far beyond relatively simple "point-and-click" activities. In order to explore data, scientists need to be able to perform extensive activities which can include entering precise data, or building complex sequences of data processing steps. Performing these tasks with a controller providing nothing more than a virtual laser pointer would be tedious and too restrictive.

In other instances, users may want to forgo the traditional numerical data input, and instead avail themselves of the unique possibilities to directly "touch" the data visualization and modify it directly with their hands. Literature has found that such "hands on" interaction with scientific data can bring benefits in both speed and accuracy, depending on the task at hand (Bach, et al., 2016).

Device Classes

All devices considered offer varying degrees of immersive experience, defined through a combination of the headset display type and user control interaction system. However, there are large differences between the different systems. To allow for a better comparison, and to identify the relative strengths and weaknesses of each system, we classified all considered devices into groups based on their technical commonalities.

The four distinct device classes we identified are:

- VR (virtual reality) devices
- AR (augmented reality) devices
- MR (mixed reality) devices
- Mobile VR (virtual reality) devices

VR devices

Virtual Reality headsets completely immerse users into a virtual environment, replacing their normal environment by a virtual one. They consist of two independent screens mounted in front of each eye and use an infrared laser tracking system to determine the rotation and position of both the users' headset and controllers in the virtual environment. VR headsets in our definition are tethered devices connected to a stationary PC via a cable allowing continuous, uninterrupted use as well as greater computational and rendering potential limited only by the connected PC system. Interaction is driven by 360° position and rotation tracked controllers with multiple button inputs and internal gyroscopes.

AR devices

Augmented reality devices display visual 3D data ("holograms") on transparent displays within the headset which overlay the virtual data into real world space and onto real world objects. They use fixed multi-directional spatial tracking cameras to orient the data in alignment with the users' surroundings and head motion. Augmented reality devices are driven by battery powered, mobile chipsets to provide freedom of movement at the cost of lower rendering capabilities. They do not require a separate computer or cables to operate. Interaction is available in the form of a combination of hand gesture recognition, voice recognition and simple "untracked" controller input system, i.e. a simple clicker button that replaces certain gestures.

MR devices

Windows MR devices offer a combination of full VR immersion and augmented reality projection. These devices employ headset mounted high contrast detection cameras to track controllers and static objects within its optical field of view, combined with a front facing "reality capture" camera allowing the user to switch between a fully immersed VR environment and a mixed reality virtual environment, which combines reality capture and virtual reality together. MR headsets, like VR headsets, are tethered to a PC and are afforded the same rendering capabilities as VR headsets. Their tracking facilities, however, are limited by the viewing angle of the inside out tracking cameras. Interactions are

driven by dual multi-button input controllers, position and rotation tracked within the viewing angle of the inside out tracking system and internal gyroscopes.

Mobile devices

Mobile virtual reality headsets are typically powered by mobile phones connected into a headset fitted with VR orthographic lenses. The use of a mobile phone as the display device greatly reduces their cost. An exception is the Oculus Go, which already contains a mobile device as a fully sealed unit. Whilst they offer a VR perspective, they can only offer rotational tracking of the headset, using the internal gyroscopes of the fitted mobile phone to provide relative motion tracking with no absolute positioning. Most mobile VR systems offer little in terms of interaction other than head movement detection. The Oculus Go, however, does have an external controller with gyroscopic movement sensors and button inputs, similar to a virtual laser pointer.

Device Selection

Using the above mentioned criteria, we surveyed the current market and evaluated available devices within their specific subcategories. The resulting device comparison evaluation matrices can be found in Appendix A-2. We then selected one representative device from each class for further evaluation.

VR device: Oculus Rift

Offers precision position and rotational headset and controller tracking, whose performance is limited only by the PC it is attached to, with built in headphones and a display quality that matches the HTC Vive. But at a reduced cost, the Oculus Rift was the more affordable virtual reality device choice.

We also note that we have a university owned HTC Vive available for development and testing purposes. This affords us the ability to test the software on multiple VR hardware devices without incurring extra cost to the project.

AR Device: Magic Leap 1

In the technical comparison between the Magic Leap 1 and HoloLens, the Magic Leap 1 is the more suitable choice for our use case. This is not unexpected, as the Magic Leap 1 is a brand new second generation AR device, while the original Microsoft HoloLens has been on the market for several years now. With a wider field of view, lighter headset, greater system memory and dedicated storage, controller and gesture interfaces, the Magic Leap is right now the superior device. In particular, the advertised eye tracking feature of the Magic Leap 1 could vastly improve the user experience compared to the HoloLens.

Pricewise, the advertised price of the Magic Leap 1 is comparable to that of the HoloLens. Unfortunately, availability of the Magic Leap 1 is limited, as it is currently only available in the continental US and there appears to be a significant waiting period.

Due to the limited availability of the Magic Leap and a loan HoloLens unit available for development and testing purposes from the University of Southampton, as well as the announcement of the HoloLens 2 in February 2019, we decided to delay the purchase of an AR device until the HoloLens 2 is available for purchase in the UK. We are currently on a waiting list with Microsoft.

MR device: Lenovo Explorer

Windows mixed reality headsets offer very similar performance to VR headsets due to being tethered to a PC, but come with a more limited optical tracking system. Across the available range of headsets, there is very little specification difference. Thus comfort, aesthetics and price are the deciding factor. As the lightest headset, with the widest field of view and at a competitive price, the Lenovo explorer was chosen in this class.

Mobile device: Oculus Go

As the least powerful range of devices, the Oculus Go offers mobile ready VR capabilities, without the multitude of potential complications, or fingerprints, of an external mobile phone. Pricewise, the Oculus Go is competitive when compared to purchasing a dedicated high-end mobile phone for the other possible solutions. With an additional controller for more precise user interaction, it is the preferred development and testing device within the mobile range.

Development Environment

There are a variety of environments available for developing content for the various VR/AR devices selected above. They differ mainly in the level at which the development takes place, ranging from vendor-specific, hardware-dependent SDKs allowing full access to the basic features of each device to all-in-one, ready-to-use solutions that require very little programming at all.

In the following, we compare three different VR/AR development environments at different levels of abstraction. The representative environments we selected for this comparison are:

1. OpenGL, OpenVR/Device SDKs, DirectX
2. The Unreal4 Engine
3. Unity 3D

OpenGL, OpenVR, DirectX

This development environment is the closest to the real hardware of each device. It consists of pure C/C++ code that is accessed by a very low-level API. When used correctly, it provides high performance and can yield relatively portable code.

However, the coding required is quite extensive. There are no pre-built graphics capabilities built in, and even simple graphic primitives (such as spheres or lines) have to be coded manually. The rendering in this setup is done by either an OpenGL or DirectX driver that uses user-supplied fragment and vertex shaders to translate a given geometry into an image. It is the responsibility of the developer to render the same image twice from slightly offset position, once for each eye. The resulting images are then displayed in the headset using either a device specific SDK (e.g. the Oculus SDK or the HTC Vive SDK).

The attitude (position and direction) of the user, as well as any tracked controllers, can be accessed directly using functionality provided by the SDK. It is entirely up to the developer to use this information to render the scene from a correct angle.

While OpenGL and DirectX are well established industry standards that hide the details of individual 3D graphics processors from the developer, the development of a similar industry-standard library for accessing VR devices is still in its infancy. The OpenVR library, developed by game publisher Steam, is an attempt to provide some level of abstraction to developers so that the same code can support multiple different devices. However, in our tests we were unable to set up and compile the OpenVR sample applications on a Linux system for use with an HTC Vive headset.

As a conclusion, we consider this development environment not suitable for our specific purpose of developing a quick proof-of-concept demonstrator application. We note, however, that this relatively lightweight development stack would be very suitable for extending existing visualization applications that already provide a full OpenGL or DirectX rendering for screen output.

Unreal4

The Unreal4 engine provides a mid-level API for developing 3D games and visualizations. Originally developed for gaming, in particular first person 3D games, it has grown to include powerful functionality for designing efficient 3D visualizations. It requires code to be developed either in C++ or using a “blueprint” system based on pre-fabricated code blocks that requires no coding at all.

The main concern with this engine is device vendor support. While there are plug-ins that allow access to VR devices, they are developed by third parties and usually not supported by the hardware

manufacturers. This poses a real risk as there is no guarantee any of these interfaces will be supported and work as expected.

Unity 3D

Unity 3D is a high-level development environment that abstracts large portions of the details of how to display and interact with objects in a 3D environment. Scenes in Unity can be created visually in a scene editor, and interactions or movement is scripted with simple C# or Java code.

Due to its high level of abstraction, Unity 3D allows cross-platform development for a variety of target platforms, including Windows, Linux, and MacOS, as well as various mobile and AR/VR platforms. Only small adjustments are needed to move an application from one platform to another, mostly to compensate for differences in the capabilities of each platform.

The key advantage of Unity 3D is industry support. Today, Unity 3D has been accepted as the de-facto standard for VR development. Practically all VR/AR device manufacturers provide some level of direct support for the Unity3D environment. This is reflected in the talent pool available for Unity development: there are plenty of developers capable of using and developing for Unity. This includes in-house expertise at the University of Southampton. In addition, the internet provides vast resources on the use of Unity and how to solve problems developers might be facing. Lastly, Unity provides a wide ecosystem of additional features that can be added to a project without having to code them through the Unity Asset Store.

All these conveniences come at some cost in the performance of the resulting programs. A custom-made, low-level visualization code written specifically for a single type of visualization will always outperform a general purpose tool such as Unity. It is unclear at the moment how big the impact of this performance degradation really is. But Unity is used in the commercial development of many graphics heavy computer games, so it is possible to use it in technologically demanding environments.

In conclusion, we decided to move ahead with Unity for the rapid prototyping and development of our proof-of-concept VR application.

Selected Development Stack

One particular shortcoming of Unity 3D alone is the integration of different types of controllers to implement user interaction. This is addressed by the opensource VRTK plugin for Unity, which provides a modular VR interaction toolkit directly within Unity. Furthermore, it allows to easily switch between different manufacturer SDKs for easy cross-device development. It also enhances the debugging facilities of Unity to extend to VR devices.

Unfortunately, as this is a third party toolkit, and this entire field is still young and developments are happening quickly, there are some incompatibilities to take into account when combining different versions of the various toolkits to build our final development toolchain. We therefore give the exact versions of each of the following tools to ensure that our working setup can be reproduced exactly. We advise to adhere to the given versions as any variation (especially to previous versions) may cause incompatibilities.

Our development stack is currently composed of:

- Unity 3D 2018.2.20f1 (Unity, used for VR development)
- Unity 3D 2018.3.7f1 (Unity, used specifically for HoloLens development)
- VRTK v3.3 (via github)
- SteamVR 1.2.3 (Steam)
- Oculus Unity integration v1.29 (Oculus)
- Windows Mixed Reality Toolkit v2 beta 2 (Microsoft)

A working setup of these tools has been assembled and is available in a shared git repository hosted within the University of Southampton. A copy of the entire project including all files required to build is included in the deliverables as outlined in Appendix A3.

4 Space Science Use Cases

In the following, we describe several possible use cases of AR and VR visualizations specifically applied to the scientific visualization of space related data. Features of these use cases are described and basic user interaction with the tool is developed. Following this exposition, one of these scenarios is selected for detailed study and prototype implementation in the following section.

For all of the use case scenarios developed below, several key considerations emerged as a common sticking point:

1. when implemented in a VR setting, the inability to interact with any existing software requires a full reimplementaion of any existing domain specific tools. This is particularly troublesome in the scientific field, where very specialized, niche tools have been developed for a long time with a plethora of different UI systems. In an AR environment, instead, it is easily possible to just keep using any existing software, and just augment its features with additional overlays in the real world.
2. This also alleviates another problem with VR systems: typing on a keyboard nowadays is a natural way of inputting complex information into computers. In VR, interactions are limited to a small number of gestures combined with some buttons and joysticks on a controller. In many cases, these interactions do not allow the precision input often required for scientific work.
3. As identified in our literature survey, many scientific visualization tools are hand crafted, specialized programs to visualize a very particular data set in a singular way. Their applicability to other problems is limited and a transfer of their functionality is difficult and time consuming. We aim to develop a more general tool that is capable of visualizing at least an entire class of data in the field.

Based on these general considerations, we identified the following two use case scenarios. We also include some additional potential use cases based on discussions with colleagues during the development of the project to highlight potential uses outside of the space field.

Scenario 1: Orbit visualization for Mission Design & Analysis (“Space AR”)

The visualization of the orbital motion of spacecraft in (three dimensional) space is one of the most natural applications of advanced visualization techniques. Many tools exist to produce interactive plots of orbits and trajectories that can be rotated and zoomed by the user. One common example is the NASA General Mission Analysis Tool (GMAT)⁶. The resulting plots are quite useful for single trajectories, but once the data density to be visualized increases, simple 2D projections of the 3D data become hard to interpret. Figure 5 shows one such plot from GMAT with 6 different trajectories around Earth. While the shape of the orbits is clear from this figure, their relative position, possible conjunctions, and their “closeness” is not easily discerned from this figure.

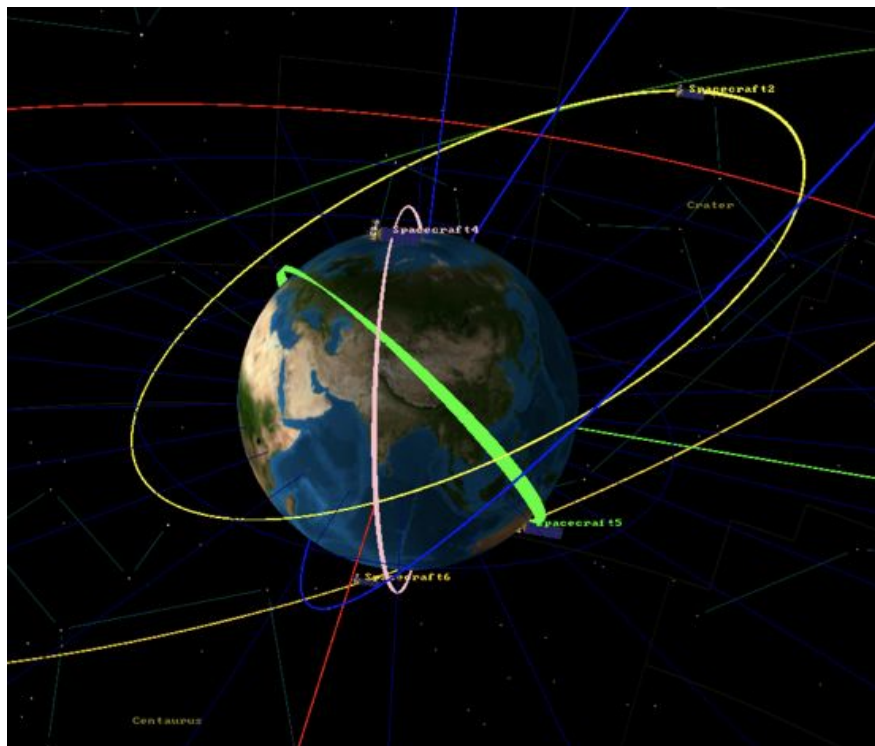


Figure 5: Several Earth bound orbits plotted using GMAT

With the addition of a stereoscopic display adding real depth perception, as AR and VR devices do, there is a possibility to improve this kind of conjunction analysis and to make the spatial relation between orbits more obvious to the user.

⁶ <https://software.nasa.gov/software/GSC-18094-1>

Further problems with orbital visualizations come in when the density of objects to display increases. A typical problem in space visualization is the vast sizes involved. Compared to the size of a satellite, the distances in space are so large that a single satellite shown to scale would simply not be visible. The solution is, of course, to increase the size of the objects shown to make them more visible. But that introduces further visual distortion in addition to the inaccuracies introduced by the 2D projection of a 3D space.

This problem becomes very clear when looking at the space environment around Earth. There are currently about 800 active satellites in low Earth orbit, with another 400 large inactive objects such as rocket bodies. Additionally, some 10,000 smaller fragments are found in orbit. Visualizing even just the large bodies is a challenging task. Figure 6 shows a simple Python plot of the orbits of 1269 low Earth objects from the NORAD TLE database generated using the Matplotlib library. It is obvious that this very simplistic visualization does not convey any information content.

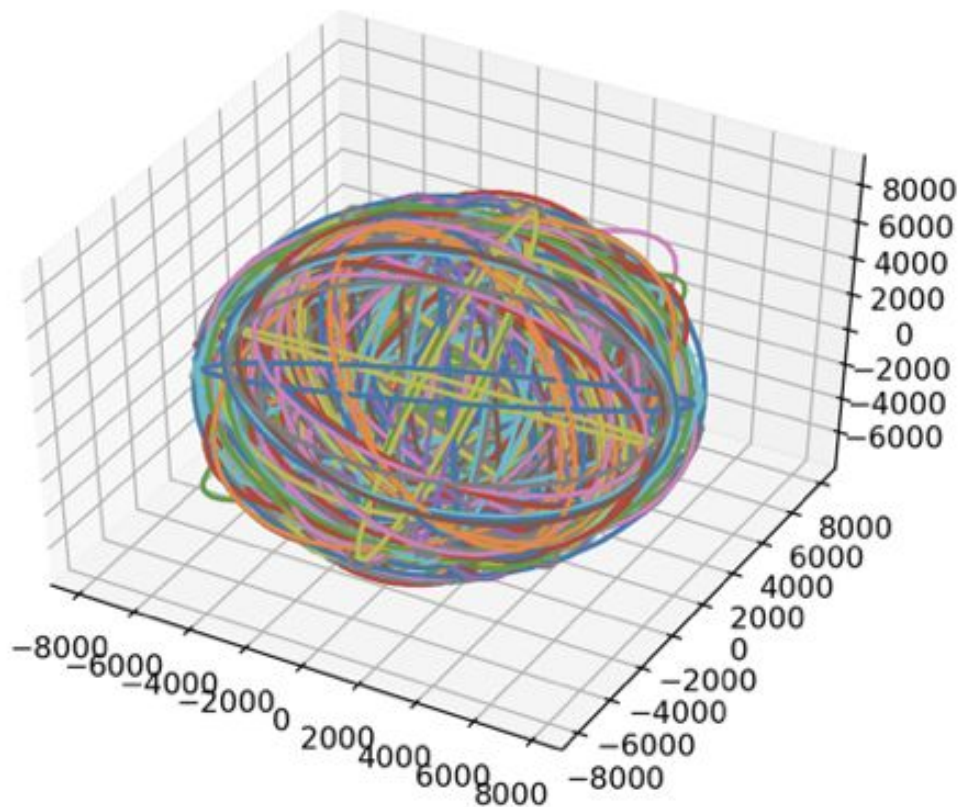


Figure 6: 1269 low Earth object orbits plotted using a simple Matplotlib 3D plot from Python

More appealing visualizations of this data are available, for example through the Stuff in Space website⁷ or artistic visualizations commissioned by space agencies (Figures 7 and 8).

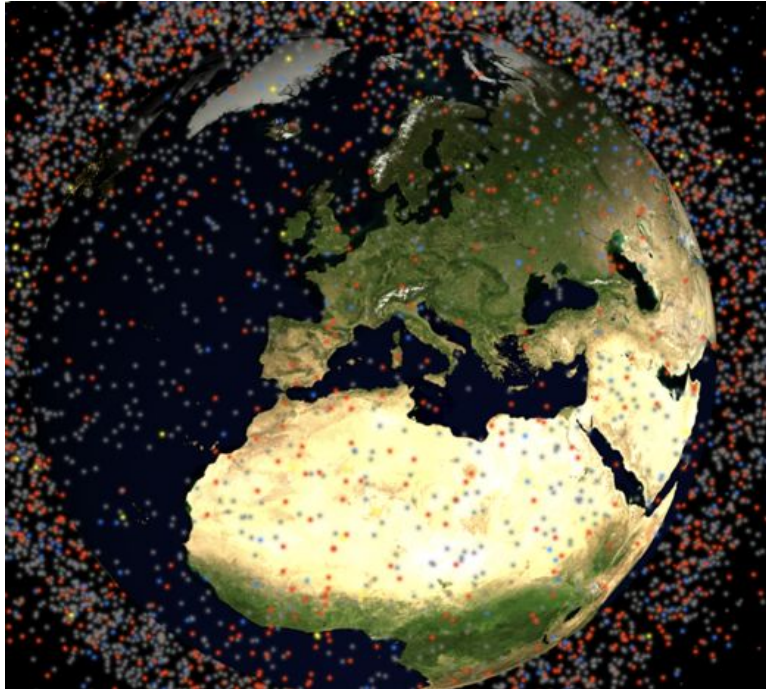


Figure 7: Screenshot of the Stuff in Space website visualizing objects in orbit around Earth in real time

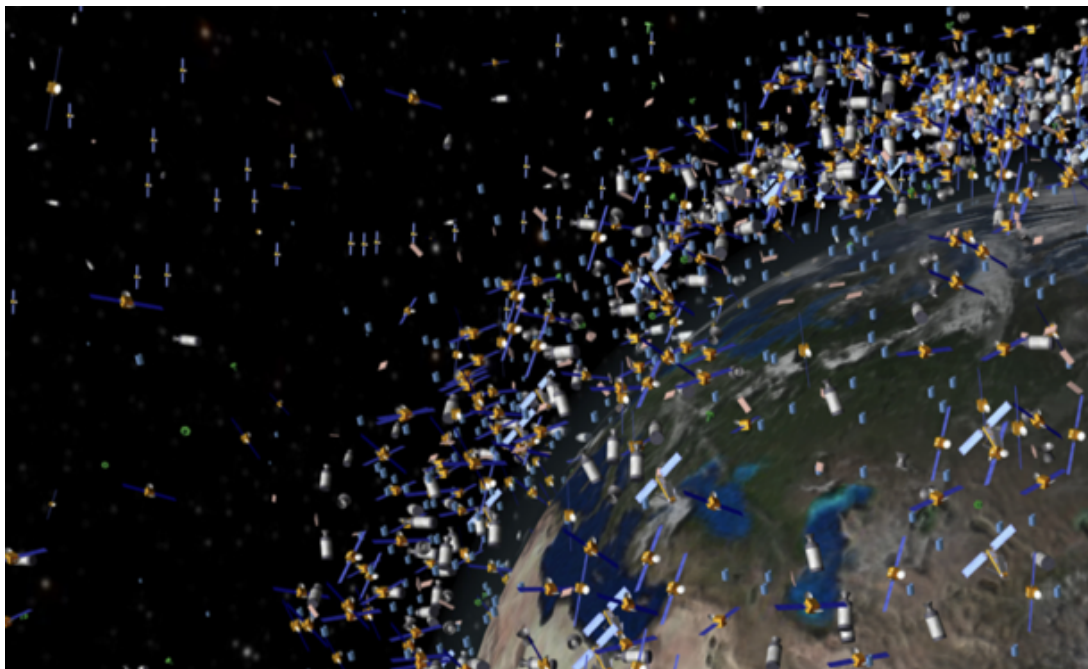


Figure 8: NASA Artist's visualization of objects orbiting Earth based on real data (not to scale)

⁷ <https://stuffin.space/>

The difficulty with these types of visualizations is that while they look very appealing, they distort the scientific data underlying the visualization. Apart from the size of the objects being vastly exaggerated (objects as depicted in Figures 7 and 8 would be on the order of 50-100 km in size), the loss of depth perception in the figures also greatly diminishes the ability to judge distances into the figure, and hence the density of objects. This is particularly visible in the region just above the Earth surface, where orbits bend around to disappear behind the Earth. In both visualization it looks like there is a big accretion of objects while in reality the density of objects around Earth is rotationally quite uniformly distributed.

This is where the the potential of orbit visualization using AR and VR devices comes in. The proposed use case scenario is depicted in Figure 9, where a user working on her usual domain specific software to generate the relevant orbital data can then view a more correct, stereoscopic projection of that data using an AR or VR device. Again here the potential power of AR to seamlessly integrate into existing workflow is highlighted. Instead of having to put on specialized VR goggles to see the visualization, with AR the user could just keep the AR device on her face at all times, merely looking to the side of her screen to see the visualization updated in real time as the data is generated and modified through her existing codes.



Figure 9: Sketch of the user interaction with the Space AR tool

Scenario 2: Earth Observation Data (“African Drought Monitor”)

This scenario is focused on the effective visualization of scientific data collected and derived from space based Earth observation. In particular, we focus on the application of the African Drought Monitoring project, which is an international collaboration between world leading universities, including the University of Southampton GeoData group, and supported by the UN.

The challenge in this field is to distill useable and actionable *information* out of the vast quantities of *data* produced by modern Earth observations missions and through data fusion with other ground-based data sources. This process is greatly helped by good visualizations of the varying geo-tagged indicators. Data is collected over a grid of points on the ground and traditionally a single scalar value is then visualized on a 2D map using a color bar as shown in Figure 10.

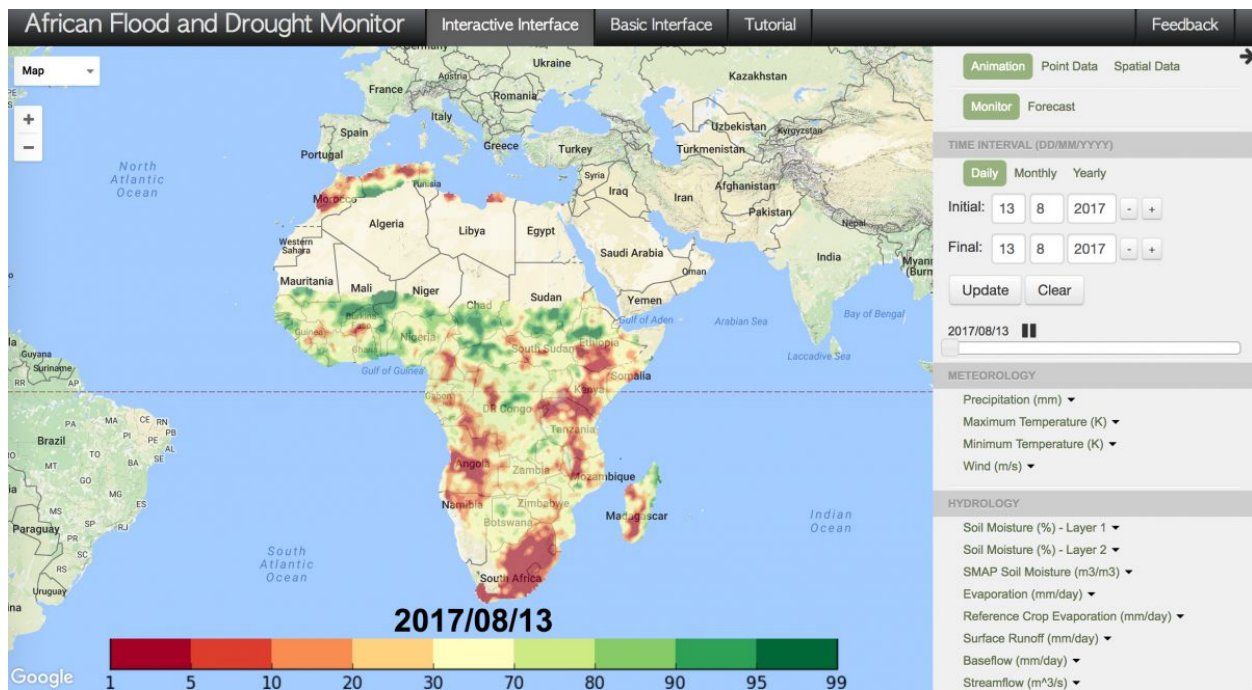


Figure 10: Current 2D user interface of the African Flood and Drought Monitor website.

The addition of a third dimension in VR/AR could allow more than one of these indicators to be displayed and analyzed at the same time. This would be particularly useful for the analysis of cause and effect, as well as in determining how certain synthetic indicators computed by synthesizing observation data from different sources relate to actual observations.

The challenges in this use case include the development of an entirely new user interface to replace the 2D mouse based interface with a more intuitive, VR based interface. While not impossible, this task is greatly complicated by the fact that such VR user interfaces are still quite new and often very time intensive to implement, much more so than traditional 2D interfaces. The lack of precise pointing with a controller when compared to a computer mouse means interfaces must be bigger and hence take up more space. Figure 11 illustrates a simple interface mockup to select two data layers to be displayed on a map.

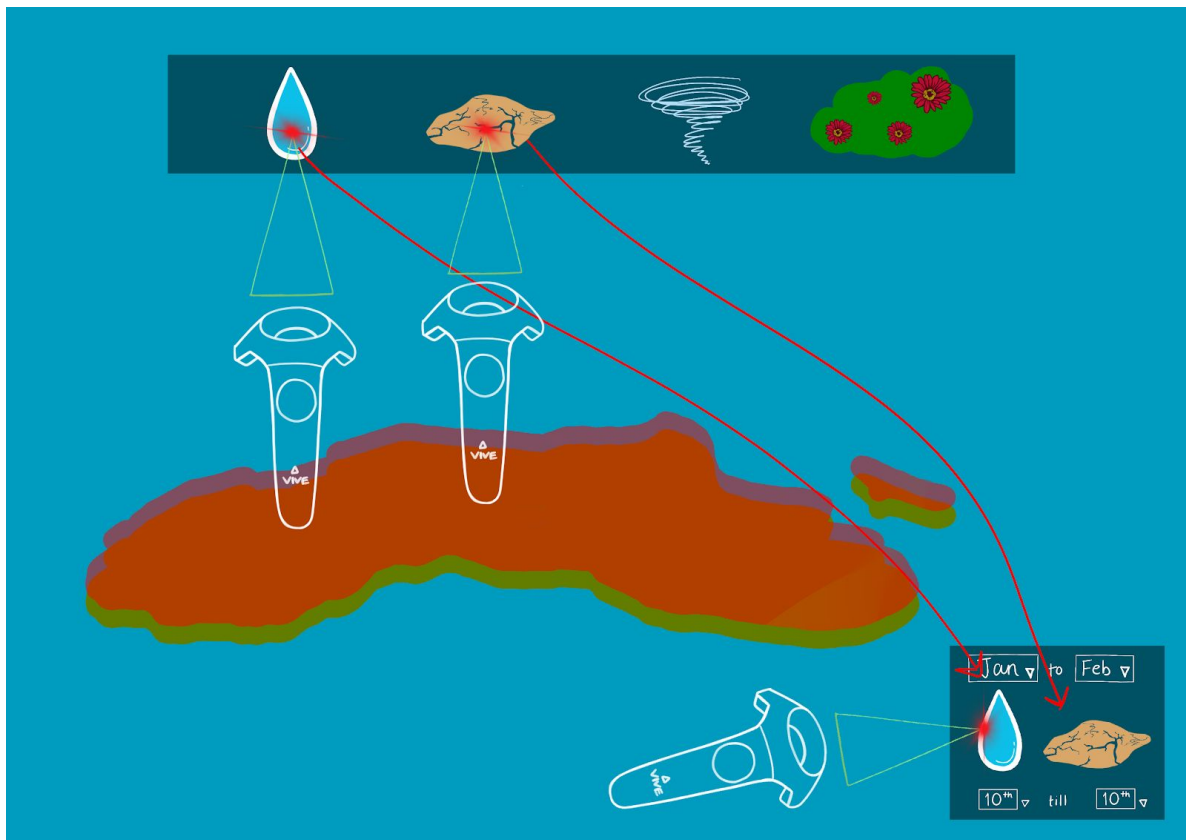


Figure 11: User interface for selection of data layers to manipulate in subsequent operations

The other challenge that can possibly be addressed by VR/AR technology is the display of more than one data set at the same time. Also this field is so far not yet well researched, but we identified several possible visualization techniques that could be applied:

- Symbolic visualization: Inclusion of symbols on the map to indicate additional sparse data on top of color coded dense map data

- Map stacking: use of the full third dimension to represent several layers of 2D color coded map data (see Figure 12)
- Elevation mapping: Combining the color coding from one data source with the ability to give the map a 3D surface profile based on the second data set (see Figure 12)

For all of these methods mentioned above their suitability for the visualization task would need to be studied after implementation of a prototype.



Figure 12: Different ways to display more than one data source.

Left: map stacking. Right: Elevation mapping.

Spin-out use cases

In addition to the two space science use cases identified above, we also had informal discussions with colleagues in other fields. In particular in aeronautics and fluid dynamics there appears to be large untapped potential for VR/AR visualization. Both fields generate large quantities of inherently three dimensional data that is very hard to visualize.

Bubble Formation in Complex Fluids

One such use-case, contributed by Dr. Ivo Peters, concerns the visualization of bubble formation in complex fluids recorded with the help of CT scans. Dr. Peters and his colleagues are interested in the way complex fluids are displaced by pressurized gas. This is for example important for natural phenomena such as volcanic eruptions, where gas bubbles are produced inside a complex fluid (magma). The high gas pressure can cause solidification of the complex fluid, with subsequent fracturing, as opposed to the formation of smooth spherical bubbles that are formed in normal fluids.

These bubbles are recorded using x-rays (CT scans). The result of the CT scan is an STL file containing the geometry of non-spherical bubbles made by injecting air into a strongly shear thickening suspension. These STL files can then be easily visualized in VR/AR environments for further analysis.

Because similar techniques are also used in medical imaging (De Ridder et. al., 2015), AR/VR technology has already been applied to such data. However, also here it seems that there is not yet a single industry standard tool that has emerged to allow this visualization. Using an already available tool for visualizing medical data developed for an earlier project, we were able to project a holographic image of the scanned bubble shape using a Microsoft HoloLens. Figure 13 below gives an impression of the visualization, a video of the complete interaction is available on youtube at

<https://www.youtube.com/watch?v=K7ELMfOF4ZU>.



Figure 13: Holographic image of a CT scan of a bubble using the Microsoft HoloLens.

Computational Fluid Dynamics (CFD)

In computational fluid dynamics, it is common to generate large data sets with pressure distribution and flow directions over complicated grids. These are the results of complex simulations of flows around obstacles, such as the air-flow around an airplane wing or a race car chassis. Powerful domain specific visualization and data analysis tools, such as paraview⁸, exist in the field that allow for appealing and easy representation of the data.

There have been some early attempts at connecting VR environments with these plotting tools (Hnwell et al., 2015). However, in the case of paraview, these attempts were made quite early on in the development of VR environments using several projectors inside a closed room, and have since stalled.

We briefly investigated the possibility of interfacing existing visualization codes such as paraview with new VR/AR technology. Conceptually, this should not be too difficult. These codes can already render the requested view from any view point. All that is required would be to render the same image twice with a small displacement of the view point, once for each eye of the observer. The resulting images could then be fed back into the VR system.

The difficulty arises from the continuous interfacing, and from the very high performance requirements of VR/AR systems. In order to work without causing discomfort to the user, the images have to be permanently redrawn at a very high frame rate (ideally 90 FPS), tracking the position and motion of the user. Additionally some kind of minimalistic user interface would have to be provided inside the VR visualization to engage with the representation even on a very simple level (e.g. rotate, zoom).

This mode of operation is sufficiently different from the typical program structure of a classical on-screen visualization. Here the image is only drawn once and then displayed statically. The user may have the option to engage with the representation on-screen using a simple mouse interface, rotating or zooming the model. As a response to each of these interactions, the image is updated to represent the new view. For this operation to feel fluent to a human user, a very low frame rate of 10 FPS or even less is sufficient. Furthermore, as soon as the user stops interacting with the model no further redrawing is required.

Another complication here stems from the need to interface with the VR/AR hardware on a low level. Directly rendering each scene for each eye into the VR/AR headset currently requires the use of

⁸ Paraview: <https://www.paraview.org/>

device specific APIs, making the resulting code highly device dependent. Code written for an Oculus device would most likely not work with an HTC Vive without modifications. There are attempts to unify the low-level API of VR/AR devices, such as the SteamVR framework (see [Development Environments: OpenGL, OpenVR, DirectX](#)). These work in a similar way as has been done with 3D graphics rendering (OpenGL, DirectX) and GPU computing (OpenCL). However, as this field is still in its infancy these unified APIs are not yet universally accepted and their use remains rather complicated.

We believe it should be possible to interface industry standard visualisation tools, especially open-source tools like paraview, with VR/AR devices. The result would be a powerful, domain specific visualization tool with a large user base and familiar to most professional users in a field. However, the development of such a tool is beyond the scope of this study.

Use Case Selection

After careful deliberation in the team, the use case selected for implementation in the prototype is the orbit visualization tool for mission analysis and design (use case 1). The factors leading to this decision were:

- Existing expertise and interest of the team in orbital dynamics
- Clear use case with a conceptually straightforward implementation of features
- Work suitable for the given time frame of the implementation phase, with a focus on AR/VR development
- Interest in the community in the resulting tool with wide practical applications in the field
- Potential difficulty interfacing with the data sources for scenario 2 which are unrelated to the use of AR/VR and hence out of the scope of this project.
- Potential difficulty in re-implementing large parts of an existing user interface in VR for scenario 2.

In the following section we further specify the detailed requirements for the prototype for this use case, and describe its implementation.

5 Prototype Development and Implementation

Prototype Definition

Based on the previous selection of the use case to be implemented, the following requirements were set for the prototype in close consultation with the wider astronautics community within the University of Southampton.

1. Interface to dynamically load orbit definitions
 - a. All data to be loaded from a JSON file/data stream
 - b. No hard coded dependence on specific scenario being visualized
2. Capability to visualize orbits
 - a. Capable of displaying at least 1000 different orbits at same time
 - b. Capable of handling orbits containing at least 150,000 points
 - c. Automatic sizing based on orbit scale
 - d. Display of full orbit (line), current position in orbit (point), or both
3. User interface
 - a. Allow user to select current simulation time
 - b. Allow automatic playback in real time / sped up time
 - c. Allow interaction with individual objects to display additional data
 - d. Allow manual changes to orbit display style (line, point) individually or globally
 - e. Allow user to load different data files
4. Device Compatibility
 - a. Microsoft HoloLens
 - b. Oculus Rift
 - c. Windows MR
5. Example Data Sets
 - a. Solar System
 - b. Low Earth satellite orbits

User Interface Design and Development

We have looked at how to implement the UI based on the scenario selected. We have separated the UI into two categories: gestures and visuals. By working on the visuals, we've developed a style guide for the UI, with a specific colour scheme and visual information (sliders and menu). For that, we have

taken into consideration best practices in HUD development in games and VR/AR environments that could be implemented in the current project. HUDs are key design elements in games, particularly in first-person shooting (FPS) games. There are at least two ways to display information in the game environment: diegetic and non-diegetic (Peacocke et al., 2015). (Zammito, 2008), expert in UX Design in games, also shows examples of non-diegetic and diegetic UIs in games and how that is influenced by the game genre. For example, a space game could show the HUD on the astronaut's helmet (and that could be considered diegetic). Thus, the diegetic HUD is part of the game fiction, whereas the non-diegetic format is not attached to the game world. For example, non-diegetic interfaces appear on the top of the screen showing health bars, score, location and other information for the user to interact with the scenario. On the other side, a diegetic UI has elements in the environment that display data, as for example, a wall, a TV and so on. In their website and online documentation, Unity 3D recommended that non-diegetic UI doesn't work in VR⁹. Depending on the interaction, diegetic UI can make the user more immersed in the experience (Peacocke et al., 2015), therefore, having a non-diegetic UI should a good choice for the Orbit visualisation scenario. On the other hand, it might be difficult for users to find the relevant information within a full diegetic UI environment, especially because users would be having a high cognitive load. Therefore, the best UI choice is between non-diegetic and diegetic, which is called spatial UI (and also mentioned in the Unity 3D online documentation).

By using the Spatial UI, users can move around the scene and the UI stays in a fixed position. This is useful for the purpose of our scenario since it is expected that the use case of the orbit visualisation should be in a work environment when scientists are willing to visualise data from different perspectives (this means that the cognitive overload is high, so attention is required).

Another requirement was the slider for time navigation in the orbit visualisation. For that, we have been through different iterations of form and colours that could be applied in the different aspects of the UI. With this, we developed a style guide for the whole project. A style guide is similar to a branding guideline. (Quesenbery, 2001) mentions that style guides are communication tools that teams utilise in order to keep consistency. This is crucial during the development stages of products/services. Style guides should contain colour variations, size restrictions, image guidelines, typography and other aspects of the implementation of the design work. For this project, we have developed sliders and the elements of the whole WIMP (window, icons, menus and pointers); all elements followed the same pattern and identity included in the style guide and all elements had possible versions. We expect to use the other variations in the future, providing personalisation of colour schemes, for example and perhaps variations on the text

⁹ <https://unity3d.com/learn/tutorials/topics/virtual-reality/user-interfaces-vr>

size in order to be more inclusive and accessible (for people who might have visual deficiencies such as colour blindness). Appendix A-1 contains a detailed justification for the style guide.

Orbit Interchange File Format

In order to exchange orbital data between a specialized user application generating the orbital data and our visualization tool, an exchange file format based on a JSON encoding of the orbital data is employed. The easy to read and write nature of JSON, along with its ubiquity and the availability of robust JSON generators and parsers in any modern programming language, make it a natural choice for this task. Furthermore, by passing the data in an extensible, object oriented fashion the file format can easily accommodate future developments and additional data features to be passed to the visualization tool.

For the prototype, all orbital data is passed in a single JSON object, called a scene object, describing the scene to be visualized. The scene object contains some metadata providing information about the scene in general in an info object, as well as an array of orbit objects to be visualized. The scene object currently has the structure shown in Table 3, while the info and orbit object structure is shown in Tables 4 and 5 respectively.

The info and scene object are self explanatory. Note that it is currently not possible to mix different coordinate systems and units within the same scene, nor is it possible to provide more than one scene per data file.

Table 3: JSON scene object description

property name	type	description
info	info object	contains general information about the scene. Information from this object applies to all orbits in this scene.
orbits	array of orbit objects	contains a list of orbit objects contained within the scene.

Table 4: JSON info object description

property name	type	description
coordinates	string	Type of coordinate system used in describing the orbits. Currently only “cartesian” is supported.
units	string	Distance units of measurement used in the orbital data. Currently “km” and “au” are supported representing kilometers and astronomical units, respectively. These units apply to all dimensional distances in the orbit objects (including e.g. radius)

Table 5: JSON orbital data object description

property name	type	description
name	string	A human readable string identifying the object. This is used to display e.g. tooltips.
display	string	Initial display style of the orbit. This currently supports the three styles “line”, “point” and “line,point” corresponding to showing the entire orbit as a line, the current location of the spacecraft as a point (sphere), or both at the same time. This property can also be changed globally or for individual orbits through the UI.
radius	number	The radius of the sphere to be displayed if the point display style is set. The program will enforce a minimum size to ensure objects remain visible.

color	string	Hex encoded color (same format as in HTML or CSS) indicating the color of the object and the orbit.
eph	array of coordinates	The ephemeris of the orbit. Each element of this array is itself an array containing the 6 coordinates describing a point. The interpretation of these 6 values depends on the coordinate system specified in the info object. For Cartesian coordinates, the 6 entries specify the x, y, z components of the position followed by the v_x , v_y , v_z components of the velocity.
time	array of numbers	This array contains the times in Julian Days (JD) corresponding to each entry in the eph array in strictly increasing order. Both eph and time of the same orbit must be of the same length, but they can differ in length between different orbits. The minimum length is 2 specifying an initial time and a final time for which the object ephemeris is valid.

The orbit objects contain the actual information to be visualized. This is mostly stored within the eph and time properties of the objects. The eph property contains an array of points in a 6 dimensional space. This is intentionally kept quite general to facilitate different coordinate systems such as various orbital element formulations to be used instead of Cartesian coordinates. If Cartesian coordinates are used, the entries in the eph array are interpreted as 6-tuples containing the x, y, z components of the position followed by the v_x , v_y , v_z components of the velocity. In the current prototype, the velocity is not used, but its presence allows for it to be visualized in future versions of the code e.g. as arrows attached to the objects.

Time is currently always given in Julian Days (JD) and is provided in the time property. Each entry in time corresponds to the same entry in the eph array of positions, and must be provided in strictly

increasing order. The prototype interpolates positions and velocities linearly between time entries (after converting to Cartesian coordinates, if necessary). This way visually smooth motion of objects along an orbit is possible even if the ephemeris is not dense.

The minimum number of entries in the ephemeris is two. This is because the tool uses the minimum and maximum times found over all orbits to set the limits of the time slider allowing the user to select the current display time. Ephemeris times for any individual orbit are considered to be continuous, covering the interval from the first to the last entry.

Any orbit for which the current display time falls outside of its specified time interval will automatically be hidden. This allows objects to “come into existence” at specific points during the visualization, e.g. as the result of a spacecraft deployment or a fragmentation event.

Future extensions

The structure shown here is clearly extensible and several possible extensions are already evident from the structure. Some extensions to this format we have already identified include:

- orbital data in different units and formats (such as Keplerian elements or similar);
- additional display styles (such as showing only a short bit of orbit around the current location, or hiding orbits completely);
- different, possibly more convenient, time formats such as MJD2000;
- additional information on each object allowing for more sophisticated styling (e.g. different shapes, 3D models, textures);
- additional information on the attitude of the satellite at each time point to allow correctly oriented display of objects;
- the info object currently only holds a bare minimum of information, but could clearly be extended to provide additional descriptions of the scene in human and machine readable form.

Example Data Sets

To illustrate and test the requirements set for the prototype, we devised the following sample applications showcasing typical cases of orbit visualization.

1. The Solar System
2. Low Earth satellite orbits

Each of these visualizations is designed to test different aspects of the prototype as described below. Both are generated by Python code completely separated from the Unity 3D project.

The Solar System

This visual representation of the Solar System includes orbits of all planets including Pluto. Figure 14 shows a simple 2D plot of the resulting orbits.

Two versions of the data are provided: one with data over a period of 10 years, and one covering 250 years. The orbits are generated using the secular equations for the orbital elements of the planets in (Standish and Williams, 2006), with all units in AU. The radii given in the data files are the physically correct mean radii for each planet.

The number of data points along each orbit is chosen such that each full period contains about 180 data points, i.e. data points are about 2 degrees apart.

This data set possesses several properties that illustrate and test different aspects of the prototype:

- Mercury has a very short period of only 88 days. Thus in the full 250 year data set, Mercury completes about 1037 full orbits containing almost 190,000 data points.
- The inner Solar System (comprising of Mercury, Venus, Earth, and Mars) is tiny compared to the outer planets in the solar system.
- The size of the planets (and the Sun) is negligible compared to the size of the Solar System.

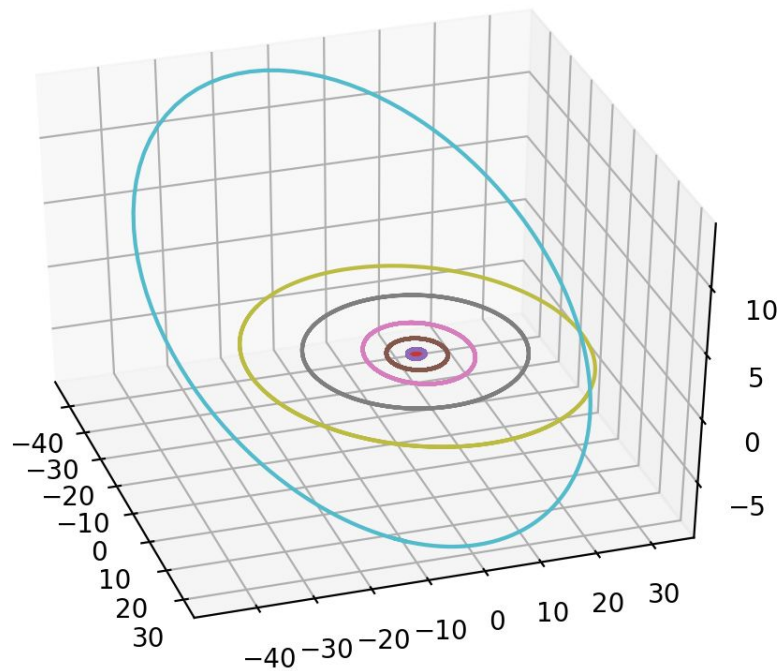


Figure 14: Simple 3D visualization of the orbital data for all planets using the Python PyPlot plotting library.

Low Earth Satellite Orbits

This is a visual representation of the orbits of all currently active Low Earth satellites (defined as having a mean motion greater than 11.25 and an eccentricity of less than 0.25) as found in the space-track.org¹⁰ LEO dataset of February 1 2019. This data set is based on the NORAD TLE catalogue and contains all objects having received updates within the 30 days prior to the query date. The resulting data set is then cross-referenced with the SATCAT data set of the same date from celestrak.org¹¹ to identify any objects with a classification other than decayed or unknown.

The resulting TLEs are then propagated in 3 minute steps using the SGP4 propagator implemented in the Python sgp4 package¹². Figure 15 shows the resulting orbits in a traditional 3D plot.

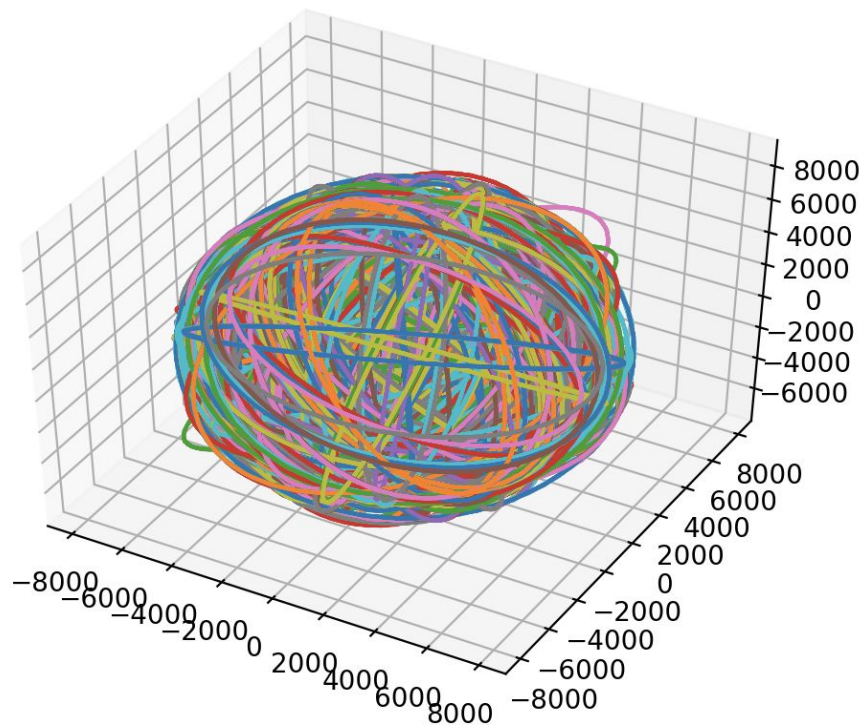


Figure 15: Simple 3D visualization of the orbital data for the full LEO orbit data set (1269 orbits) using the Python PyPlot plotting library.

¹⁰ https://www.space-track.org/basicspacedata/query/class/tle_latest/ORDINAL/1/EPOCH/%3Enow-30/MEAN_MOTION/%3E11.25/ECCENTRICITY/%3C0.25/OBJECT_TYPE/payload/orderby/NORAD_CAT_ID/for/mat/tle

¹¹ <https://www.celestrak.com/satcat/search.php>

¹² <https://pypi.org/project/sgp4/>

The data set is provided in several variants for testing purposes:

1. 12 different orbits over 24 hours
2. 102 different orbits over 24 hours
3. 1269 different orbits over 4.8 hours (all orbits meeting the given criteria)

Each data set also includes the Earth as an additional object fixed at the center of the visualization. The size of the Earth is physically correct, while the radius of each satellite is intentionally exaggerated to 50 km so they are easily visible. This can be thought of as a “critical conjunction warning” radius rather than the physical size of the satellite.

This data set possesses several properties that illustrate and test different aspects of the prototype:

- The largest data set contains over 1200 different objects and corresponding orbits.
- A large number of satellites clustering closely around a central body

6 Prototype Evaluation and Testing

Evaluation of the Prototype

Evaluating the final prototype against the prototype specification in Section [Prototype Definition](#), we find that the developed prototype essentially meets all specifications as set out.

In some cases, allowances had to be made due to time constraints for partial implementation of requirements which could be improved in a further development activity of the project. These are indicated in orange in Table 6.

Table 6: Prototype evaluation

Requirement	Note / Implementation
1.a. All data to be loaded from a JSON file/data stream	A basic JSON object structure has been defined and a parser for it has been implemented within Unity.
1.b. No hard coded dependence on specific scenario being visualized	All information required is passed via JSON. Currently, some scaling constants are hard coded depending on the units identified in the JSON file.
2.a. Capable of displaying at least 1000 different orbits at same time	This requirement is tested by scenario “LEO orbits - 1269 orbits - 0.1 days”. Displaying only the current position is possible with this scenario on all supported devices.
2.b. Capable of handling orbits containing at least 150,000 points	This requirement is tested by scenario “solar system 1800 - 2050”. The orbit of Mercury in this scenario contains 189,575 data points. On the HoloLens, showing lines for this orbit does introduce lag due to limited computing power on the device.
2.c. Automatic sizing based on orbit scale	This requirement is tested by scenarios “LEO orbits” and “Solar System” which are of different scale (km and au).
2.d. Display of full orbit (line), current position in orbit (point), or both	The display style can be controlled either from the JSON data or using the user interface.
3.a. Allow user to select current simulation time	A global time slider is included in the main menu. The limits are automatically extracted from the provided JSON data.

3.b. Allow automatic playback in real time	An option for automatic playback in real time is provided in the main menu. Due to limitations of some devices, playback is limited to 4 updates per second, leading to potential “skipping” of objects along orbits.
3.c. Allow automatic playback in accelerated time	An option for automatic playback in accelerated time is provided in the main menu. Acceleration steps can be selected on an exponential scale. Due to limitations of some devices, playback is limited to 4 updates per second, leading to potential “skipping” of objects along orbits.
3.d. Allow interaction with individual objects to display additional data	Individual objects can be selected using a device specific method (e.g. pointing, looking at them). Selection of an object opens an additional per-object menu.
3.e. Allow manual changes to orbit display style (line, point) individually or globally	An option for changing display style globally for all objects is provided in the main menu. An option for changing display style individually per object is provided in the per-object menu.
3.f. Allow user to load different data files	An option for loading different data files is provided in the main menu. Data files can either be loaded from static sample files included in the tool, from web addresses, or from local files (not in HoloLens).
4.a. Microsoft HoloLens	A special version of the tool for the Microsoft HoloLens 1 is provided.
4.b. Oculus Rift	A version of the tool compatible with the Oculus Rift is provided.
4.c. Windows Mixed Reality	A version of the tool compatible with the Oculus Rift is provided.
5.a. Solar System Data File	Two versions of a data file including the entire solar system to scale are provided with 10 years of data and 250 years of data, respectively.
5.b. Low-Earth satellite orbits Data File	A file is provided containing the orbits of all currently active low Earth orbiting satellites as of February 1, 2019 listed in the space-track LEO dataset ¹³ .

¹³ <https://www.space-track.org/>

Testing

Testing of the prototype, both from a technical side as well as testing of the user interface was mostly carried out through internal testing in the team with additional feedback collected from a presentation of the work to local experts in the Astronautics group at the University of Southampton. Due to the early state of the prototype development, it was decided to forgo formal interviews or user questionnaires in favor of informal discussions.

In the following, we list some key outcomes of the testing phase:

What worked

- The style guide ensured a consistent look and feel of the prototype. The contrast of the colour scheme worked very well.
- All display performance related targets were achieved, in particular with respect to the number of orbital objects and the length of the orbits. This is a particular achievement given the often relatively low computational and graphics power available on AR and VR devices.
- The JSON interface to external code proved very robust and is capable of transmitting data of a wide variety of scenarios to the prototype for display.
- The selection of Unity 3D as the development platform allowed us to reuse the same code to generate and test prototypes on a variety of platforms.

What changed

- Although we have developed different versions of the slider as a standalone object, the slider was integrated to the UI inside a window that could be controlled by the users.
- During the development of the project we had to take design decisions such as adding a main menu object attached to a pedestal that would be the centre of the visualisation. This was necessary to ensure that the user has an initial point of interaction. From that, each orbit had an attached window/tag that could then be opened and adapted according to the user's preferences.

Although not initially planned as such, the rapid development of the prototype over effectively just two months required a fast way to expose the quickly changing options and settings in the program to the user. While not ideal from a UI design perspective, the selected method does provide quick access to those settings.

What can be improved

- Attitude of objects should be included. The JSON data format is already specifically designed to be flexible to allow easy inclusion of such additional data. Once included, this change will allow for physically correct 3D models and textures to be applied to objects to e.g. display the surface of the Earth correctly aligned in space or replace the spheres by correctly aligned satellite models.
- For the future, the UI could be implemented in different versions that could be personalised by users (particularly if users are colour blind). Also, the variation of colours could be very helpful for the different applications MR, VR and AR, since in the MR and AR the background is not controlled by the software (in VR is the “space” texture, which will be always dark).
- Clicks and interactions with the UI should be also reviewed and simplified. Currently some simple, common tasks require many interactions to achieve.

Platform specific results

During the platform specific testing, we identified some challenges associated with a particular platform. In particular it is worth mentioning that the Oculus Go platform that was tested did not perform very well both due to limited processing power and inherent limitations in the quality of the hardware visualization. Furthermore, for the Microsoft HoloLens we only had access to the developer preview version of the HoloLens 1. It is believed that the new HoloLens 2 will remedy several of the shortcomings identified in this report.

Our findings are summarized in Table 7 in the form of a RAG analysis.

Table 7: Platform specific test results

	physical setup	computational power	user comfort	user interaction	visual quality
HTC Vive	requires mounting	PC		large controller	
Oculus Rift	requires positioning	PC		large controller	
Oculus Go	none	very limited	slightly heavy	small controller	poor resolution
Microsoft Mixed VR	none	PC		large controller	
Microsoft HoloLens 1	none	limited	heavy	Hand gestures (latest SDK)	small field of view

5 Conclusions & Future Work

The project was completed successfully. A thorough literature review has been carried out, identifying uses for VR and AR in scientific data visualizations. The current landscape of AR and VR devices has been analyzed and a representative set of devices has been selected for further study. Two possible scenarios for the application of AR and VR technology to space science have been analyzed. Following this analysis, the orbit visualization for mission design and analysis scenario has been selected for implementation. Subsequently, a prototype has been developed and tested to show the applicability of the technology to this field.

The prototype was developed with particular focus on the usability as a general tool in the field. Instead of implementing a specific visualization for a particular set of orbital data, the code is kept general, allowing a user to feed in orbital data from any domain specific source. This versatility and flexibility is successfully demonstrated by means of two very different example data sets: one depicting the solar system, and one showing the motion of Low-Earth satellites around the Earth.

The input data JSON format proposed in this work is by design easily extensible to allow the easy inclusion of additional features in future iterations of the software. It is believed that such flexible yet easy to use orbit visualization tools will be required for the use of AR and VR devices in scientific applications.

Both AR and VR devices were found to provide greatly improved depth perception for users when compared with traditional 2D projections of 3D orbital data. However, it is in the nature of VR devices to insulate the user from the surroundings, which greatly interrupts the well established workflow of researchers. This limits their application as a tool in mission analysis and design as all interactions with existing orbit generation code would have to take place inside the VR environment. AR devices, on the other hand, show great promise despite their early state of technological development. With these it may be able to seamlessly bridge the interactive 3D holographic display and the traditional 2D screen interface to established domain specific software used for mission analysis and design.

Impact of the work

As part of the project, several presentations on various aspects of the work have been prepared or carried out. Further presentations of the work are currently being considered. Confirmed presentations include the following meetings:

- 4th international workshop on key topics in orbit propagation applied to space situational awareness (KePassa 2019), Logroño, Spain. April 24 - 26 2019.
<https://kepassa.unirioja.es/>
Accepted oral presentation by Dr. Alexander Wittig.
- Workshop at Colegio Pedro II (Rio de Janeiro, Brazil) on techniques and development of holograms and VR/AR technology in education (workshop given by Dr. Vanessa Wanick to tutors and designers). April 1 - 2 2019.
- Astronautics Seminar at the University of Southampton. April 3 2019.
Oral presentation by Dr. Alexander Wittig and Rob Penn.
- Dr. Vanessa Wanick and Jess Castle are working on a full paper to the XIII International Conference on Graphics Engineering for Arts & Design that will be held in Rio de Janeiro in September 2019

Further peer-reviewed publication of the outcome of this project is currently being considered. Possibly this will require some additional work to be carried out before being ready for publication.

Furthermore, the prototype will be used for outreach activities at the University of Southampton as part of Open Day displays and possibly on be put on display during the University of Southampton Design Show 2019.

The way forward

As part of the development of an early stage prototype to demonstrate the feasibility of using AR and VR for the visualization of mission design and analysis, we identified several points worthy of further study. We believe that addressing these issues will greatly improve the usability and applicability of the tool in the field. During development, we already took great care to allow for such extensions to be added to the prototype in an easy manner.

- **Attitude & orientation:** the attitude of an object can be included in the extensible JSON data format. This will allow the rendering of textures and 3D models instead of spheres for different objects.
- **Textures and spacecraft models:** once attitude of objects is known to the visualization tool, it is straightforward to include textures (e.g. the surface of Earth) or even entire 3D spacecraft models in the display. This will vastly improve the appeal of the visualization while also extending the scientific application to the visualization of attitude dynamics e.g. during rendezvous and docking or debris removal missions.

- **Collision demo:** the current prototype code already allows for objects to appear/disappear depending on the current simulation time. This opens the door to simulate events with non-constant numbers of objects such as cubesat deployment or collision events. Due to time constraints, we were currently not able to include such a demonstration in the project but it is possible to generate the required orbital data to visualize this without changes to the prototype in keeping with the paradigm of flexibility adopted in its development.
- **Applications:** during a presentation of the work at the Astronautics seminar at the University of Southampton on April 3, colleagues have identified several possible applications for the Space AR visualization tool, ranging from space debris mitigation methods to scientifically correct visualization of the vast distances in the solar system.

We plan to carry on work on this project in the context of student projects both in the Astronautics group and at the Winchester School of Arts. Our students are highly motivated and resourceful and we expect their enthusiasm and their contributions to be a driving force in the further development of the prototype.

At the same time, there is a limit to how much can be achieved with a constantly changing cast of students working on the project. For this reason we are continuously looking for further funding opportunities to carry on our development of the prototype to fully leverage the power of modern VR and AR devices and turn the prototype into a viable product.

References

- Ahlberg, G. et al. (2007). Proficiency-based virtual reality training significantly reduces the error rate for residents during their first 10 laparoscopic cholecystectomies. *The American Journal of Surgery*, 193(6), 797–804.
<https://doi.org/10.1016/J.AMJSURG.2006.06.050>
- Awadhi, S. et al. (2018). Interactive Virtual Reality Educational Application. *Advances in Science, Technology and Engineering Systems Journal* Vol. 3, No. 4, 72-82 (2018).
- Andrienko, G., Andrienko, N., Keim, D., MacEachren, A. M., & Wrobel, S. (2011). Challenging problems of geospatial visual analytics. *Journal of Visual Languages & Computing*, 22(4), 251–256.
<https://doi.org/10.1016/j.jvlc.2011.04.001>
- Azuma, R. T. (1997). A Survey of Augmented Reality, 4(August), 355–385.
<https://doi.org/10.1016/j.dss.2003.08.004>
- Bach, B., Dachsel, R., Carpendale, S., Dwyer, T., Collins, C., & Lee, B. (2016). Immersive Analytics. *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces - ISS '16*, 529–533.
<https://doi.org/10.1145/2992154.2996365>
- Bowman, D. (2019). 3D User Interfaces. [online] The Interaction Design Foundation. Available at: <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/3d-user-interfaces> [Accessed 2 Apr. 2019].
- Bozgeyikli, E., Raij, A., Katkooi, S. and Dubey, R., 2016, October. Point & teleport locomotion technique for virtual reality. In Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play (pp. 205-216). ACM.
- Bromley S. (2017). Measuring Simulator Sickness | Steve Bromley - User Research.
<http://www.stevbromley.com/blog/2017/05/26/measuring-and-minimising-sickness-in-vr-games/>.
Published 2017. (accessed on 29-11- 2017).

- De Ridder, M., Jung, Y., Huang, R., Kim, J., & Feng, D. D. (2015). Exploration of Virtual and Augmented Reality for Visual Analytics and 3D Volume Rendering of Functional Magnetic Resonance Imaging (fMRI) Data. *2015 Big Data Visual Analytics, BDVA 2015*.
<https://doi.org/10.1109/BDVA.2015.7314293>
- Dix, A. et al. (2004). Human-Computer Interaction. *Human-Computer Interaction, Third(January)*, p.834.
- Dünser, A., Steinbügl, K., Kaufmann, H., & Glück, J. (2006). Virtual and Augmented Reality as Spatial Ability Training Tools. In *CHINZ '06 Proceedings of the 7th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: design centered HCI* (pp. 125–132).
<https://doi.org/10.1021/np0204186>
- Elsayed, N. A. M., Sandor, C., & Laga, H. (2013). Visual analytics in Augmented Reality. *2013 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013*, (October), 4–7.
<https://doi.org/10.1109/ISMAR.2013.6671817>
- Fictum, C. (2017). VR UX. [S.l.]: CreateSpace Independent Publishing Platform.
- Furness, T. A. (1988). Harnessing virtual space. *Society for Information Display Digest*, 16, 4-7.
- Gavish, N., Gutiérrez, T., Webel, S., Rodríguez, J., Peveri, M., Bockholt, U., & Tecchia, F. (2015). Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks. *Interactive Learning Environments*, 23(6), 778–798.
<https://doi.org/10.1080/10494820.2013.815221>
- Ghadirian, P., & Bishop, I. D. (2008). Integration of augmented reality and GIS: A new approach to realistic landscape visualisation. *Landscape and Urban Planning*, 86(3–4), 226–232.
<https://doi.org/10.1016/j.landurbplan.2008.03.004>
- Google (2017). Designing Screen Interfaces for VR (Google I/O '17). Youtube.
<https://www.youtube.com/watch?v=ES9jArHRFHQ>. Published 2017. (accessed on 29-11- 2017).
- Hanwell, M. D., Martin, K. M., Chaudhary, A., Avila, L. S. (2015). The Visualization Toolkit (VTK): Rewriting the rendering code for modern graphics cards. *SoftwareX*, 1, 9-12.
<https://doi.org/10.1016/j.softx.2015.04.001>

Haynes, P., Hehl-Lange, S., & Lange, E. (2018). Mobile Augmented Reality for Flood Visualisation. *Environmental Modelling and Software*, 109, 380–389.

<https://doi.org/10.1016/j.envsoft.2018.05.012>

Helbig, C., Bauer, H.-S., Rink, K., Wulfmeyer, V., Frank, M., & Kolditz, O. (2014). Concept and workflow for 3D visualization of atmospheric data in a virtual reality environment for analytical approaches. *Environmental Earth Sciences*, 72(10), 3767–3780.

<https://doi.org/10.1007/s12665-014-3136-6>

Hupont, I., Gracia, J., Sanagustín, L. and Gracia, M.A., 2015, May. How do new visual immersive systems influence gaming QoE? A use case of serious gaming with Oculus Rift. In 2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX) (pp. 1-6). IEEE.

IJsselsteijn, W., Van Den Hoogen, W., Klimmt, C., De Kort, Y., Lindley, C., Mathiak, K., Poels, K., Ravaja, N., Turpeinen, M. and Vorderer, P., 2008, August. Measuring the experience of digital game enjoyment. In Proceedings of measuring behavior (pp. 88-89). Wageningen, Netherlands: Noldus Information Technology.

Ishii, H. & Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits, and atoms. Proceedings of the 8th international conference on Intelligent user interfaces, (March), pp.3–3.

Ishii, H. (2008). Tangible bits: beyond pixels. Proceedings of the 2nd international conference on Tangible and Embedded Interaction (TEI '08), pp.xv–xxv.

Ishii, H. et al. (2012). Radical Atoms : Beyond Tangible Bits , Toward Transformable Materials. *Interactions*, XIX(February), pp.38–51. Available at: <http://dl.acm.org/citation.cfm?id=2065337>.

Jetter, H.C., Reiterer, H. & Geyer, F., 2014. Blended Interaction: Understanding natural human-computer interaction in post-WIMP interactive spaces. *Personal and Ubiquitous Computing*, 18(5), pp.1139–1158.

Kageyama, A., Tamura, Y., & Sato, T. (2000). Visualization of Vector Field by Virtual Reality. *Progress of Theoretical Physics Supplement*, 138, 665–673.

<https://doi.org/10.1143/PTPS.138.665>

- Keim, D., & Thomas, J. (2008). Scope and Challenges of Visual Analytics. *IEEE Visualization Conference 2007*, 4404(4404), 1–58.
https://doi.org/10.1007/978-3-540-71080-6_6
- Kim, M., Yi, S., Jung, D., Park, S., & Seo, D. (2018). Augmented-reality visualization of aerodynamics simulation in sustainable cloud computing. *Sustainability (Switzerland)*, 10(5).
<https://doi.org/10.3390/su10051362>
- LaViola, J.J., Kruijff, E., McMahan, R.P., Bowman, D. and Poupyrev, I.P., 2017. 3D user interfaces: theory and practice. Addison-Wesley Professional.
- Peacocke, M., Teather, R.J., Carette, J. and MacKenzie, I.S., 2015, October. Evaluating the effectiveness of HUDs and diegetic ammo displays in first-person shooter games. In *2015 IEEE Games Entertainment Media Conference (GEM)* (pp. 1-8). IEEE.
- Pike, M., Wilson, M.L., Benford, S. and Ramchurn, R., 2016, May. # Scanners: A BCI Enhanced Cinematic Experience. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 293-296). ACM.
- Quesenbery, W., 2001. Building a better style guide. In *Proceedings of UPA*.
- Rossmann, J., Bücken, A., Hoppen, M., & Priggemeyer, M. (2016). Integrating Virtual Reality, Motion Simulation and a 4D GIS. *Research in Urbanism Series*, 4(1), 25–42.
<https://doi.org/10.7480/rius.4.854>
- Shufelt, J. W. J. (2006). A vision for future virtual training. *Nato: Rto-Mp-Hfm-136*, (October 2002), 4–15.
<http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA473302>
- Slater, M., Usoh, M. and Steed, A., 1994. Depth of presence in virtual environments. *Presence: Teleoperators & Virtual Environments*, 3(2), pp.130-144.
- Smith, N. G., Knabb, K., DeFanti, C., Weber, P., Schulze, J., Prudhomme, A., DeFanti, T. A. (2013). ArtifactVis2: Managing real-time archaeological data in immersive 3D environments. In *2013 Digital Heritage International Congress (DigitalHeritage)* (pp. 363–370). IEEE.
<https://doi.org/10.1109/DigitalHeritage.2013.6743761>

Standish, E. M., Williams, J. G. (2006).

Orbital Ephemerides of the Sun and Moon and Planets. In *Explanatory Supplement to the Astronomical Almanac*, University Science Books

Verdaasdonk, E. G. G., Dankelman, J., Lange, J. F., & Stassen, L. P. S. (2008). Transfer validity of laparoscopic knot-tying training on a VR simulator to a realistic environment: A randomized controlled trial. *Surgical Endoscopy*, 22(7), 1636–1642.

<https://doi.org/10.1007/s00464-007-9672-3>

Vorderer, P., Wirth, W., Gouveia, F.R., Biocca, F., Saari, T., Jäncke, F., Böcking, S., Schramm, H., Gysbers, A., Hartmann, T. and Klimmt, C., 2004. MEC spatial presence questionnaire (MEC-SPQ): Short documentation and instructions for application. Report to the European community, project presence: MEC (IST-2001-37661), 3.

Wanick, V., Xavier, G. and Ekmekcioglu, E. (2018). Virtual Transcendence Experiences: Exploring Technical and Design Challenges in Multi-Sensory Environments. In Proceedings of the 10th International Workshop on Immersive Mixed and Virtual Environment Systems (pp. 7-12). ACM.

Zammito, V., 2008, July. Visualization techniques in video games. In *EVA*.

Zhang, S., Demiralp, C., Keefe, D. F., DaSilva, M., Laidlaw, D. H., Greenberg, B. D., ... Deisboeck, T. S. (2001). An immersive virtual environment for DT-MRI volume visualization applications: a case study. In *Proceedings Visualization, 2001. VIS '01*. (pp. 437–584). IEEE.

<https://doi.org/10.1109/VISUAL.2001.964545>

Zhu, B. S., & Bi, F. (2012). The Study on 3D Urban Planning Based on VR and GIS. *Advanced Materials Research*, 446–449, 3714–3717.

<https://doi.org/10.4028/www.scientific.net/AMR.446-449.3714>

Zollmann, S., Schall, G., Junghanns, S., & Reitmayr, G. (2012). Comprehensible and interactive visualizations of GIS data in augmented reality. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

https://doi.org/10.1007/978-3-642-33179-4_64



Zollmann, S., Poglitsch, C., & Ventura, J. (2017). VISGIS: Dynamic situated visualization for geographic information systems. *International Conference Image and Vision Computing New Zealand*.


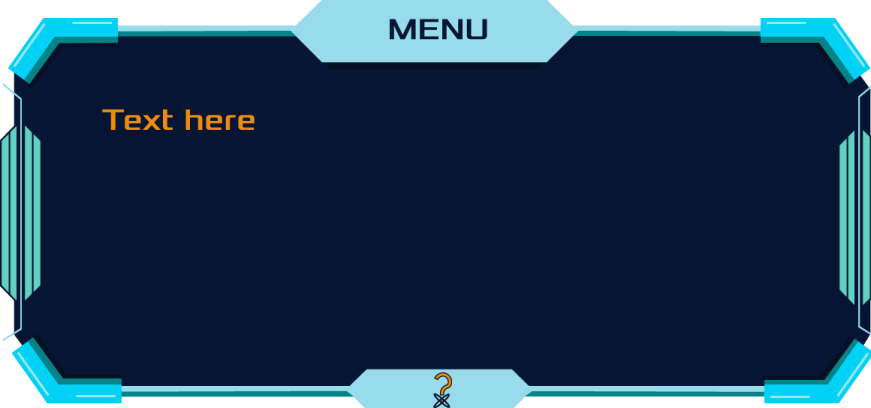
<https://doi.org/10.1109/IVCNZ.2016.7804440>

Appendix

A1 - Design style guide

Table A1-1: Style Guide justification

UI Design Element	Justification
Typography	<p>The selected font families are:</p> <p>Xolonium - used for menu boxes, information boxes. This sans serif font is similar to those that you would find in a sci-fi films, apps or video games and therefore to further immerse the user into the product, it was fitting to keep consistent to the genre.</p> <p>Xolonium in capital letters is used for titles such as MENU and OPTIONS in order to show the user clearly what the boxes are used for.</p> <p>Lower case is used for text inside of the boxes in order to preserve space and not overcrowd the box.</p> <p>Space Marine - the second font family used in the experience. This font has the primary use of notifying the user that an error has occurred through a information box. It is only available in capital letters with the idea that it conveys the feeling of urgency or error.</p>
Colour scheme	 <p>Using the planets and the solar system as the basis for this, we collated some of the more common colours found to convey the idea of space; comparing this also to other products currently available such as sci-fi films and video games. The dark blue creates a good contrast with the yellow which means that it increases its accessibility. This was tested through a colour blindness/deficiency app, Oracle, in order to make sure we made the visuals as inclusive as possible.</p>
Slider	 <p>Following on from the colour scheme, the slider was arranged so that the colours could help differentiate certain time frames. So as not to rely on just colour and improve its accessibility, the tag was added in order to mark a specific point in time. The design itself was produced through references from other video games and sci-fi films and tv shows as</p>

	inspiration.
Button	 <p>Using the colour scheme, we combined two of the colours that contrasted the most in order to produce this button. The shape was constructed to look rigid, as if it was a part of a spaceship.</p>
Window	 <p>Combining research from video games, films and images we created a HUD window that could display multiple aspects such as MENU, OPTIONS, ALERT or ERROR. Rigid block-type shapes were used in order to give the idea of being part of some machinery. As the prototype itself is full of data, we decided to make the window simple in that you can access text as well as access help/settings '?' if the user needed to adapt their experience. Using the colours from the above colour scheme, we combined these in order to give depth, shadows and highlights to the window in order for it to stand out in a 3D space.</p>

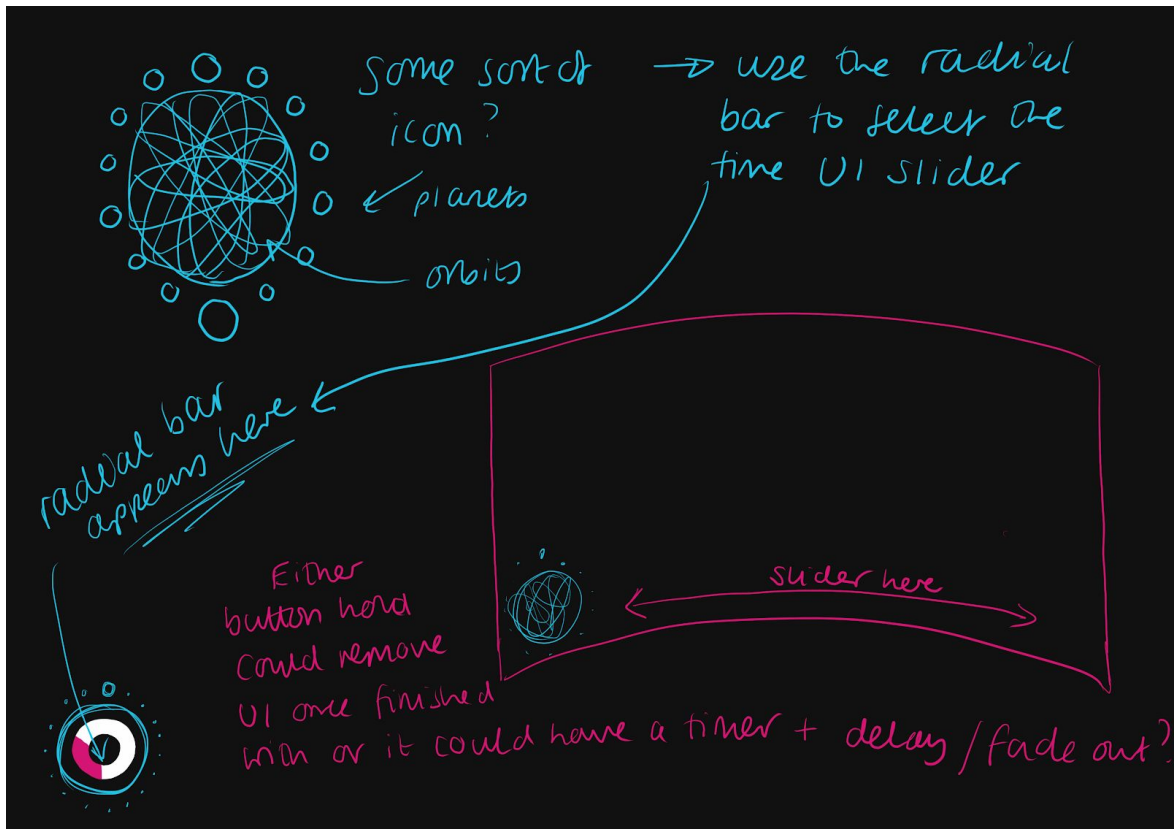


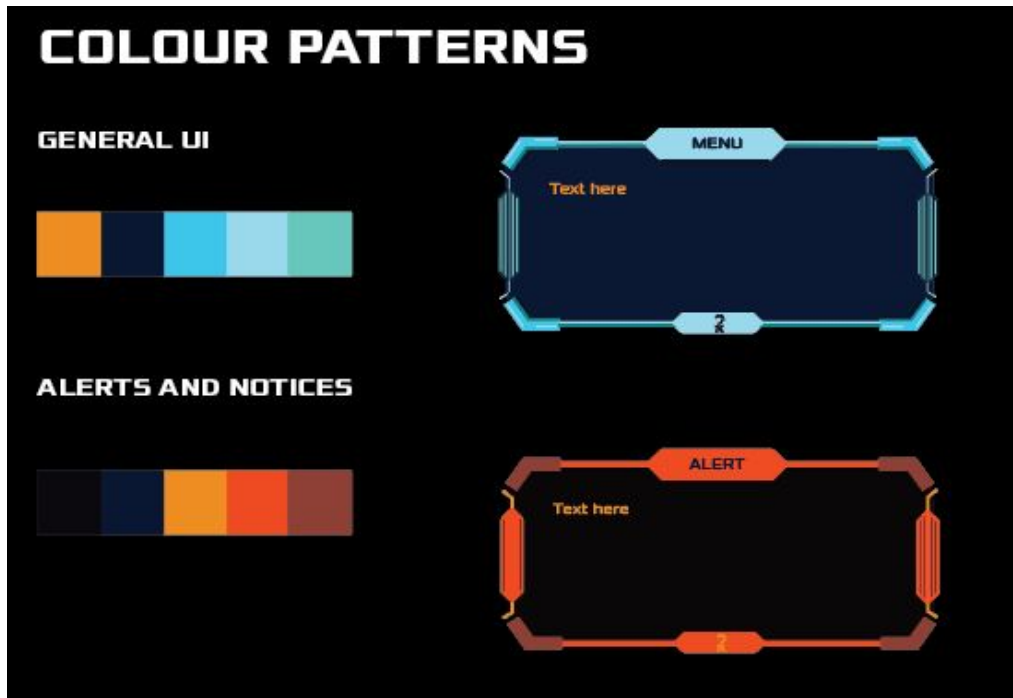
Figure A1-1: Sketches and ideas for the UI



Figure A1-2: Tests on the application of the UI slider and sketches/iterations made by the team



Figure A1-3: Versions of the colour pattern for the menu



COLOUR PATTERNS

GENERAL UI



ALERTS AND NOTICES



Figure A1-4: UI colour pattern

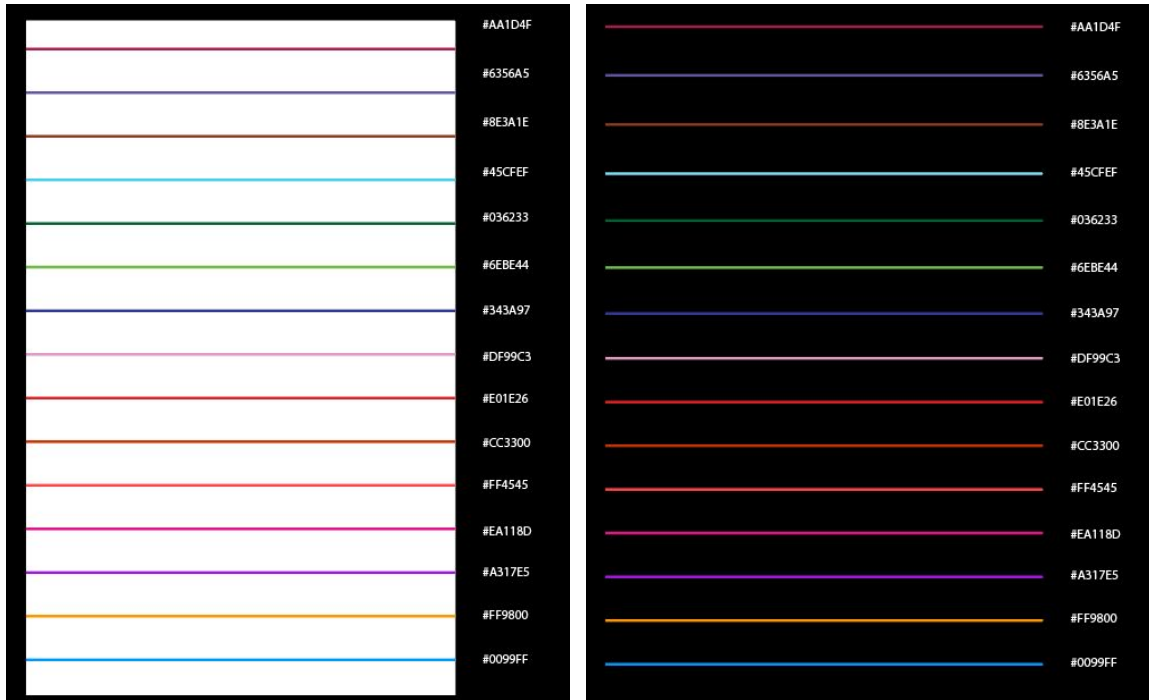


Figure A1-5: UI Orbit Colour Pattern for AR and VR

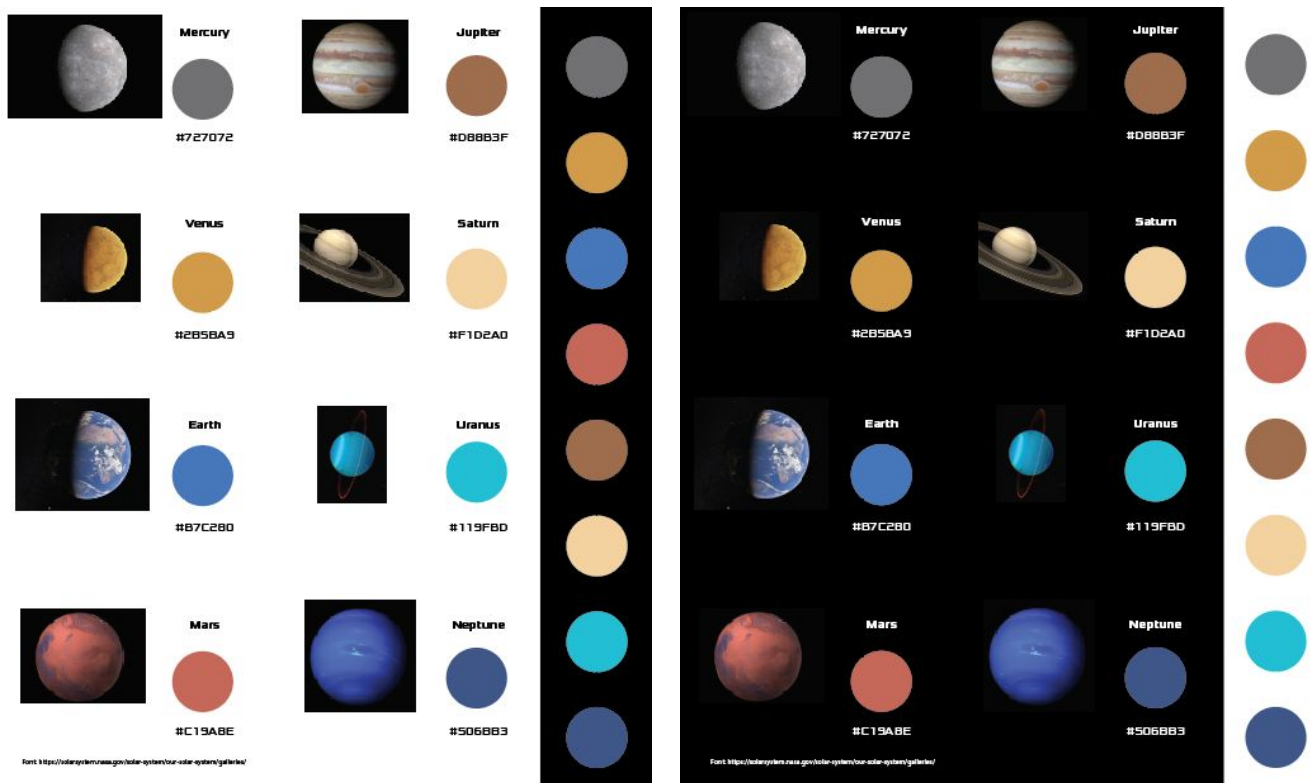


Figure A1-6: UI Planet Colour Pattern for AR and VR

A2 - Device Trade-off Tables

1. Augmented Reality Comparison
2. Mobile Virtual Reality Comparison
3. Virtual Reality Comparison
4. Windows Mixed Reality Comparison

AR Device Comparison

Subject/Item	Microsoft HoloLens	Magic Leap One
Price	Development Edition: £2,719 Commercial Edition: £4,529	£2,995
Resolution Per eye	1268*720	1280*960
Internal Storage	64gb	95gb
System Memory	2gb	8gb
Battery Life	2-3hrs can be used whilst charging	3-7.5hrs can be used whilst charging
Development	Unity + Visual Studio	Unity or Unreal engine
Interaction System	Hand gestures + Clicker remote	6DoF Remote Controller and Hand gestures
Tracking System	4 Environment Understanding Cameras 1 Inertial Measurement Unit 1 Depth Camera 1 Ambient Light Sensor	6DoF position and orientation
Sizing/Fit	adjustable headstrap	2 Headset Sizes 64.9mm + Less 65mm + Up
Audio system	Built in spatial sound speakers and 3.5mm audio jack	Onboard speakers and 3.5mm audio jack with spatialisation processing
Headset Weight	579g processor built into headset	345g separate tethered processing unit with clothing clip
Field of View	35°	40°
Voice Commands	Microsoft Cortana built in	speech to text
Eye Tracking	none	Eye Tracking @30fps
Network Capabilities	wifi 802.11ac	wifi 802.11ac/b/g/n bluetooth 4.2
Microphone	*4 built in spatial microphones for mixed reality capture	real world ambient audio capture
Eye Glasses support	fits over frames	prescription inserts
Haptic feedback	none	LRA haptic device (controller)
CPU	intel 32bit architecture Custom Holographic Processing Unit (HPU 1.0)	2*Denver 2.0 64bit cores 4*ARM cortex A57 64bit cores
GPU	1gb HPU	Nvidia Pascal 256 CUDA cores
Controller details	clicker replaces gesture click + hold commands	touch sensitive touchpad 8-bit resolution trigger button digital Bumper Button Digital Home Button
IPD adjustment interpupillary distance	automatic	headset sizing based on IPD.

Mobile VR/AR Comparison				
Subject/Item	google DayDream (VR)	Oculus Go (VR)	Gear VR	Epson Moverio (AR)
Price	£99	32gb £199 64gb £249	£30 without controller £45 with controller	£400 BT-300 £804 BT-350
Resolution Per eye	depends on phone	1280*720	depends on phone	1280*720
Internal Storage	depends on phone	32gb/64gb	depends on phone	16gb + microSD expansion slot
System Memory	depends on phone	3gb	depends on phone	2gb
Battery Life	depends on phone	2-3hrs	depends on phone	6hrs
Development	Android studio, Unity, Unreal Engine.	Android studio, Unity, Unreal Engine.	Android studio, Unity, Unreal Engine.	Android studio
Interaction System	Daydream Controller	Oculus Go Controller	Gear VR Controller	tethered Media Controller
Tracking System	gyro sensor worldsense dual fisheye cameras	Proximity sensor gyro sensor 3DoF sensor GPS	gyro sensor proximity sensor	accelerometer gyro sensor
Sizing/Fit	adjustable headstrap	adjustable headstrap	adjustable headstrap	worn like glasses
Audio system	mobile phone 3.5mm audio jack mobile phone speaker	spatial audio speakers built in 3.5mm audio jack	mobile phone 3.5mm audio jack mobile phone speaker	3.5mm audio/mic jack
Headset Weight	261g	470g	345g	69g
Field of View	depends on phone	100°	101°	23°
Voice Commands	not built in	not built in	not built in	none
Eye Tracking	none	none	none	none
Network Capabilities	phone based	wireless 802.11b/g/n/ac bluetooth 4.1	phone based	wireless 802.11a/b/g/n/ac wifi-miracast bluetooth 4.1
Microphone	phone based	built in	phone based	3.5mm audio/mic jack
Eye Glasses support	over the glasses	over the glasses - additional foam spacer prescription lense inserts available	over the glasses	over the glasses
Haptic feedback	none	none	none	none
CPU	depends on phone	Quad-core Qualcomm Snapdragon 821 (two 2.3GHz Kryo HP cores and two 2.15GHz Kryo cores)	depends on phone	intel atom x5, 1.44ghz quad core
GPU	depends on phone	adreno 530	depends on phone	no details
Controller details	9 axis inertial measurement unit wireless bluetooth rechargeable battery touchpad back button home button	3DoF sensor AA battery (included) touchpad trigger button back button home button	Gyro Sensor Accelerometer Magnetic Sensor touchpad back button home button	accelerometer gyro sensor GPS Home button Menu button Back button Function key (lock/brightness/2d/3d) direction control key (up/down/left/right) capacitive multitouch touchpad
IPD adjustment interpupillary distance	no adjustment	no adjustment	no adjustment	no adjustment
compatible mobile phones	Asus ZenFone AR Google Pixel Google Pixel 2 Huawei Mate 9 Pro Huawei Porsche Design Mate 9 LG V30 Motorola Moto Z Motorola Moto Z2 Samsung Galaxy S8 Samsung Galaxy S8+ ZTE Axon 7	stand-alone device	Galaxy Note9*, S9, S9+, Note8, S8, S8+, S7, S7 edge, Note5, S6 edge+, S6, S6 edge, A8 Star, A8, A8+ *Galaxy Note9 is compatible with Gear VR model number SM-R325NZVC*** only.	standalone device

Virtual Reality Device Comparison

Subject/Item	Oculus Rift	HTC Vive	HTC Vive Pro
Price	:£399	£499	starter edition: £1,048 enterprise edition: £1,299
Resolution Per eye	1080*1200	1080*1200	1440*1600
Internal Storage	PC based	PC based	PC based
System Memory requirements	8gb+	4gb+	4gb+
Battery Life	controllers only up to 20hrs - AA battery	controllers only - 6-9hrs lithium ion rechargeable	controllers only - 6-9hrs lithium ion rechargeable
Development	Unity or Unreal Engine	Unity or Unreal Engine	Unity or Unreal Engine
Interaction System	controller based	controller based	controller based
Tracking System	Accelerometer Gyro Sensor Magnetometer Constellation tracking camera external infrared tracking sensors	360-degree area coverage Accelerometer Gyroscope Front facing camera Lighthouse system V1 (SteamVR tracking)	360-degree area coverage Accelerometer Gyroscope Front facing camera Lighthouse system V2 (SteamVR tracking)
Tracking area	1.5mx1.5m - 2 sensors 2.4mx2.4m - 3 sensors (additional sensor sold separately)	4mx3m - 2 lighthouses	6mx6m - 2 lighthouse v2 10mx10m - 4 lighthouse v2 (additional lighthouses sold separately)
Sizing/Fit	adjustable elastic/velcro headstrap	adjustable elastic/velcro headstrap	adjustable headstrap
Audio system	built in integrated audio or £49 in ear headphones	3.5mm audio jack or £99 deluxe audio strap	his-res certificate headset headphone
Headset Weight	470g	470g	769g
Field of View	110°	110°	110°
Voice Commands	none	none	none
Eye Tracking	none	none	none
Network Capabilities	3rd party wireless adapter available	Bluetooth official HTC vive adapter available £299	Bluetooth official HTC vive adapter available £365
Microphone	built in	built in	built in
Eye Glasses support	over the glasses	over the glasses	over the glasses lense distance adjustment
Haptic feedback	buffered and non-buffered haptic controllers	Realistic HD haptic feedback	Realistic HD haptic feedback
CPU requirements	intel i5 or greater	intel i5 or greater	intel i5 or greater
GPU requirements	nvidia gtx960 or greater	nvidia gtx1060 or greater	nvidia gtx1060 or greater
Controller details	6DoF motion tracking Analog stick 2 joystick buttons menu/oculus button analog trigger button grab button requires AA batteries	6DoF motion tracking Menu button Touch sensitive touchpad system button dual stage trigger button dual grab buttons rechargeable lithium ion internal batteries	6DoF motion tracking Menu button Touch sensitive touchpad system button dual stage trigger button dual grab buttons rechargeable lithium ion internal batteries
IPD adjustment interpupillary distance	57-71mm mechanical slider	60.9-73.6mm mechanical adjustment	60.9-73.6mm mechanical adjustment IPD sensor

Windows Mixed Reality Headset Comparison						
Subject/Item	Acer Headset Part Number: VD.R02EE.003 £199 - box.co.uk £250 - amazon	Asus Headset HC102 £430 - windows store £449 - amazon	Dell Visor VRF100 £255.87 - dell.com £250 - amazon	HP Headset VR100 £386 - £493	Lenovo Explorer £400 - lenovo.com £329 - Amazon	Samsung HMD Odyssey
Resolution Per eye	1440*1440	1440*1440	1440*1440	1440*1440	1440*1440	1440*1440
Internal Storage	PC based	PC based	PC based	PC based	PC based	PC based
System Memory	PC based	PC based	PC based	PC based	PC based	PC based
Battery Life	Controllers only (AA batteries)	Controllers only (AA batteries)	Controllers only (AA batteries)	Controllers only (AA batteries)	Controllers only (AA batteries)	Controllers only (AA batteries)
Development	Unity or Unreal engine	Unity or Unreal engine	Unity or Unreal engine	Unity or Unreal engine	Unity or Unreal engine	Unity or Unreal engine
Input/Interaction System	dual 6DoF controllers	dual 6DoF controllers	dual 6DoF controllers	dual 6DoF controllers	dual 6DoF controllers	dual 6DoF controllers
Tracking System	Accelerometer Gyro Sensor Magnetometer Proximity Sensor Inside out camera tracking	Accelerometer Gyro Sensor Magnetometer Proximity Sensor Inside out camera tracking	Accelerometer Gyro Sensor Magnetometer Inside out camera tracking	Accelerometer Gyro Sensor Magnetometer Inside out camera tracking	2 x Inside-out motion tracking cameras Proximity Gyroscope Accelerometer Magnetometer	2 x Inside-out motion tracking cameras Proximity Gyroscope Accelerometer Magnetometer
Sizing/Fit	adjustable headstrap	adjustable headstrap	adjustable headstrap	adjustable headstrap	adjustable headstrap	adjustable headstrap
Audio system	3.5mm audio jack	3.5mm dual audio/mic jack	3.5mm dual audio/mic jack	3.5mm dual audio/mic jack	3.5mm audio jack	built in spatial sound headphones
Headset Weight	440g	400g	590g	834g	380g	644g
Horizontal Field of View	100°	95°	105°	95°	110°	110°
Voice Commands	none	none	none	none	coriana	coriana
Eye Tracking	none	none	none	none	none	none
Network Capabilities	bluetooth controller pairing	bluetooth controller pairing	bluetooth controller pairing	bluetooth controller pairing	bluetooth controller pairing	bluetooth controller pairing
Microphone	adjustable "wand" microphone	3.5mm dual audio/mic jack	3.5mm dual audio/mic jack	3.5mm dual audio/mic jack	built in mic	Array Mics
Eye Glasses support	over the glasses	over the glasses	over the glasses	over the glasses	over the glasses	over the glasses
Haptic feedback	ERM (eccentric rotating mass) vibration motors	ERM (eccentric rotating mass) vibration motors	ERM (eccentric rotating mass) vibration motors	ERM (eccentric rotating mass) vibration motors	ERM (eccentric rotating mass) vibration motors	ERM (eccentric rotating mass) vibration motors
operating system requirement	windows 10 fall creators update	windows 10 fall creators update	windows 10 fall creators update	windows 10 fall creators update	windows 10 fall creators update	windows 10 fall creators update
CPU requirements	Intel Core i5 7200U, Intel Core i5 4590 or better	Intel Core i5 7200U, Intel Core i5 4590 or better	Intel Core i5 7200U, Intel Core i5 4590 or better	Intel Core i5 7200U, Intel Core i5 4590 or better	Intel Core i5 7200U, Intel Core i5 4590 or better	Intel Core i5 7200U, Intel Core i5 4590 or better
GPU requirements	Integrated Intel HD Graphics 420, NVIDIA MX150, AMD Radeon RX 440/560 or better	Integrated Intel HD Graphics 420, NVIDIA MX150, AMD Radeon RX 440/560 or better	Integrated Intel HD Graphics 420, NVIDIA MX150, AMD Radeon RX 440/560 or better	Integrated Intel HD Graphics 420, NVIDIA MX150, AMD Radeon RX 440/560 or better	Integrated Intel HD Graphics 420, NVIDIA MX150, AMD Radeon RX 440/560 or better	Integrated Intel HD Graphics 420, NVIDIA MX150, AMD Radeon RX 440/560 or better
HDMI support	HDMI 1.4 HDMI 2.0	HDMI 1.4 HDMI 2.0	HDMI 1.4 HDMI 2.0	HDMI 1.4 HDMI 2.0	HDMI 1.4 HDMI 2.0	HDMI 1.4 HDMI 2.0
USB	USB 3.0	USB 3.0	USB 3.0	USB 3.0	USB 3.0	USB 3.0
Controller details	Visible light constellation LED 6DOF Tracking within HMD Camera FOV Inertial Measurement Unit Magnetic Sensor Thumb stick with mechanical select touchpad with mechanical select analog trigger button grab button windows button menu button 171g (with battery) bluetooth classic	Visible light constellation LED 6DOF Tracking within HMD Camera FOV Inertial Measurement Unit Magnetic Sensor Thumb stick with mechanical select touchpad with mechanical select analog trigger button grab button windows button menu button 171g (with battery) bluetooth classic	Visible light constellation LED 6DOF Tracking within HMD Camera FOV Inertial Measurement Unit Magnetic Sensor Thumb stick with mechanical select touchpad with mechanical select analog trigger button grab button windows button menu button 171g (with battery) bluetooth classic	Visible light constellation LED 6DOF Tracking within HMD Camera FOV Inertial Measurement Unit Magnetic Sensor Thumb stick with mechanical select touchpad with mechanical select analog trigger button grab button windows button menu button 171g (with battery) bluetooth classic	Visible light constellation LED 6DOF Tracking within HMD Camera FOV Inertial Measurement Unit Magnetic Sensor Thumb stick with mechanical select touchpad with mechanical select analog trigger button grab button windows button menu button 171g (with battery) bluetooth classic	Visible light constellation LED 6DOF Tracking within HMD Camera FOV Inertial Measurement Unit Magnetic Sensor Thumb stick with mechanical select touchpad with mechanical select analog trigger button grab button windows button menu button 171g (with battery) bluetooth classic
additional details	hinged display, allowing user to flip up display whilst working	hinged display, allowing user to flip up display whilst working	hinged display, allowing user to flip up display whilst working	hinged display, allowing user to flip up display whilst working	hinged display, allowing user to flip up display whilst working	hinged display, allowing user to flip up display whilst working
cable length	4m	4m	4m	4m	4m	4m
IPD adjustment	64mm with software adjust	64mm with software adjust	64mm with software adjust	64mm with software adjust	64mm with software adjust	65-72mm mechanical adjustment
interpupillary distance	64mm with software adjust	64mm with software adjust	64mm with software adjust	64mm with software adjust	64mm with software adjust	65-72mm mechanical adjustment

A3 - Codes

As part of deliverable D3 (working prototype) the following files are provided in digital form along with this report:

1. Unity_VR.zip
The Unity 3D project required to build the VR prototype.
2. Unity_HoloLens.zip
The Unity 3D project required to build the HoloLens prototype.
3. Prototype_VR.zip
The compiled VR prototype program.
4. Prototype_HoloLens.zip
The compiled HoloLens prototype program.
5. Sample.zip
A Jupyter (Python) notebook and required TLE data files to generate the example visualization data (Solar System and LEO orbits) used in this report.
6. Prototype.mp4
Video showing the user view of a simple interaction with the HoloLens version of the prototype.