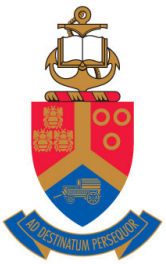


UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COS 301 Capstone Project 2017

Vulknut Software Engineering

Compiled By

Peter Boxall - u14056136
Claude Greeff - u13153740
Marin Peroski - u13242475
Johan du Plooy - u12070794
Bernhard Schuld - u10297902

3D VR Presentations

GitHub Repository: Valknut Software Engineering

Contents

1 System Requirements and Design

1.1 Introduction

1.1.1 Purpose

This document serves to outline the overall description and requirements of the system. This document also serves as a guideline to the developers in order to ensure the final product meets these requirements, and indicates to the client what the required technologies are in order to be able to use this system.

1.1.2 Scope

The overall objective of this project is to provide any given user with a toolkit, with which the individual can create a 3D virtual reality presentation with ease. Our goal is to make it simple to use, enabling virtually any user to utilize the power of 3D, without having to build 3D objects completely from scratch. The user would custom build a 3D environment built upon a variety of available templates offered, or by selecting a set of 3D models and skyboxes when choosing to create a project from the ground up, taking user experience to a whole nother level.

1.1.3 Definitions, Acronyms and Abbreviations

1.1.3.1 MEAN

MongoDB, Express.js, AngularJS (or Angular), and Node.js

1.1.3.2 VR

Virtual Reality

1.1.3.3 MVP

Minimum Viable Product

1.1.3.4 MTBF

Mean Time Between Failures

1.1.3.5 e2e

End-To-End

1.1.4 Software Description

Presentation software such as Microsoft's Power Point, have been around for almost 2 decades now. However due to many constraints and limited hardware capabilities, these presentations are only available as a 2D representation. With Virtual and Augmented reality growing ever more populer and easily accessible, should it not be possible to add a 3rd dimention to our presentations? This software aims to do exactly that.

This software will make it possible for virtually anyone familiar with a computer to design and edit 3D presentations, to be viewed via the use of VR headsets such as the HTC VIVE and the Occulus Rift. After selecting to create a new project for example, a user would be presented with a 3D world that will be mostly empty at first. The user can then select from a variety of customization options and toolboxes to modify and add content to their world.

When the user is satisfied with the world they see before them, they can then create a method of presentation, such as recording a tour through the world, that can even feature their voice if the user so desires. At any point during presentation recording, the user can still modify any part of

their surrounding world, if for example they change their mind about the placement of an object, or even the lighting in a room to name but a few.

Finally when the user has finished setting up the intire presentation, they can choose to export the final product as a 360 degree video, that can then be viewed on any device that supports 360 videos. Then at the time of presentation, everyone in the audience will be able to join in so long as they have a VR headset at their disposal.

1.2 Design

1.2.1 Software Methodology

We will follow the Agile development methodology. The principles this methodology is based on advocates planning, constantly evolving development, early delivery and continues improvements, and it encourages flexibility as well as maintainability.

The agile development process is built on four main principles:

1. Individual and team interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

The Agile development approach allows for frequent opportunities for clients to be involved in. Requirements are then reprioritized according to client specifications and they are elaborated on. The process of Agile development is based on the following actions:

- Short timeboxes of iterative development.
- Early and repeated client/user feedback.
- Reprioritization of work based on the client/user so that emergent requirements can be handled.
- Selecting a specific approach of which there are a variety of options including, Extreme Programming, Scrum, Lean Development, and Feature-Driven Development.

Some of the benefits of using the Agile development include stakeholder engagement, transparency, early and predictable delivery, predictable costs and schedule, allows for change, focus on the client, and ultimately improving the quality of the software.

For the above-mentioned reasons, we had chosen to utilize the Agile Software Methodology as it was the most applicable satisfying our needs as well as our client's.

1.2.2 Development Technique

During our first meeting with EPI-USE they had mentioned that we should make use of a development technique called MVP. A MVP is the most basic version of a product that can still be released. The point of this technique would be that early adopters would see the potential that the final product could offer, and give developers valuable feedback needed to guide them forward.

1.3 System Requirements

The Virtual Reality presentation software will have various requirements that will need to be fulfilled in order to deliver a viable product.

1.3.1 Functional Requirements

The following functional requirements will be met:

1.3.1.1 360 video export

The software must be able to export a presentation as a 360 degree video. The reason for this exporting format is to allow for a viewable format that will be as universal as possible to all VR capable devices ranging from mobile phones to VR headsets.

1.3.1.2 VR Device Viewable

The software must be able to be viewable and intractable with the use of a Virtual Reality headset such as the Oculus Rift and HTC Vive. A user must be able to preview his or her virtual reality presentation within a virtual reality environment if they own the appropriate hardware as well as edit the environment in virtual reality space.

1.3.1.3 Selectable Skyboxes

There needs to be a variety of skyboxes that a user can select from in order to match the theme of the environment. The skyboxes also need to be compatible with the Unity engine in order to be incorporated into the scene properly.

1.3.1.4 Template Environments

Template environments must be available to the user. These environments will vary to allow for a range of scenes to be catered for. A small set of template environments will be created by the development team and the rest by the community (community driven software approach).

1.3.1.5 Various Object Imports Into Scene

A variety of objects will need to be available for placement in the scene. These objects will need to be compatible with the Unity engine as well as relate to some possible environment. Users should be allowed to import their own objects as well as objects created by the community.

1.3.1.6 Editor Pont of View

The placement and control of the point of view of the user who is currently editing the scene (whether using a VR device or not) is vital. The user will need to be able to either walk on the terrain of the environment or be able to fly in the environment for better perspective of the presentation.

1.3.1.7 Native Interface Design

The interface for the software will need to run in the Windows environment as the application is being developed for the Windows operating system. It will also need to be easy to use and navigate.

1.3.2 Non-Functional Requirements

The following non-functional requirements will be met:

1.3.2.1 Image Importing

Images the creator wants to display in the scene will need to be able to be imported into it. Various image formats such as .jpg's, .png's and bit map images will need to be catered for, eliminating a formatting issue.

1.3.2.2 Slideshows

Imported images must be able to be incorporated into a slideshow or multiple slideshows. These will allow the creator to describe a topic in detail through various images that relate to the subject matter.

1.3.2.3 Video Importing

Videos allow for a rich and detailed explanation of the topics the user is trying to cover. The software must be able to incorporate .mp4 video formats into the presentation that can be viewable in the virtual reality space.

1.3.2.4 Audio Importing

The software will need to be able to incorporate audio snippets or songs into the scene. The required format type that needs to be catered for will be an .mp3 file. Voice recordings will also fall part of the audio file requirement as they are an important asset that can be incorporated into the presentation. This will allow the creator to direct a user's attention to a certain part of the scene. It will also allow the creator to describe parts of the presentation to the viewer. The software must be able to import voice recordings users make and save with third party applications.

1.3.2.5 Community Driven Content

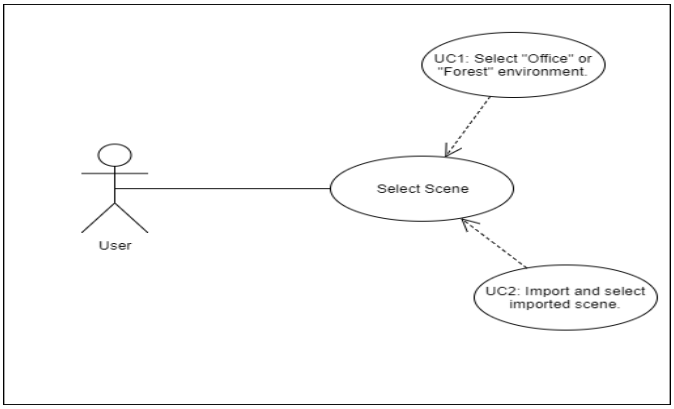
The software will need to be able to allow users to create their own content and upload it to a central server which can then be viewed and downloaded by any user who wishes to use the created content. This will cover environmental scenes, 3D objects, presentation templates, skyboxes and other Unity based assets that are supported by the software.

1.4 Application Design

1.4.1 Use Cases

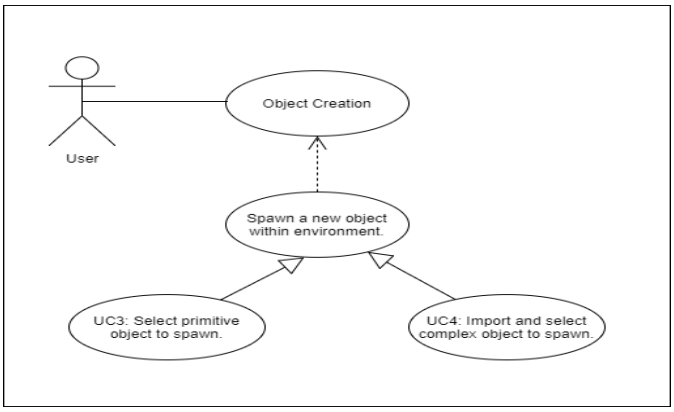
The following services are provided for the presentation creator.

- Scene Selection:



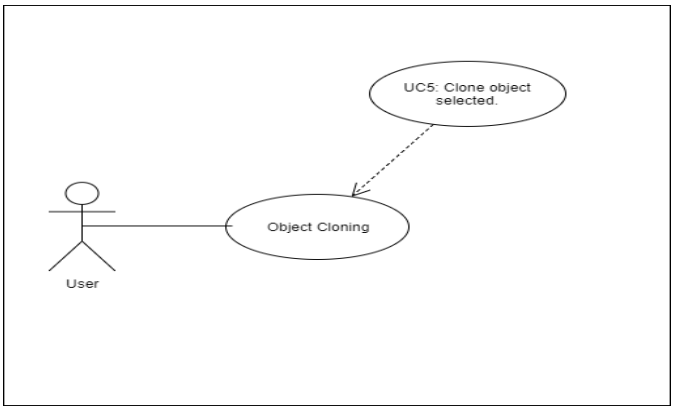
Precondition: The user has the "Office" and "Forest" scene in their asset project folder.
Postcondition: The user is spawned within the selected environment.

- Object Creation:



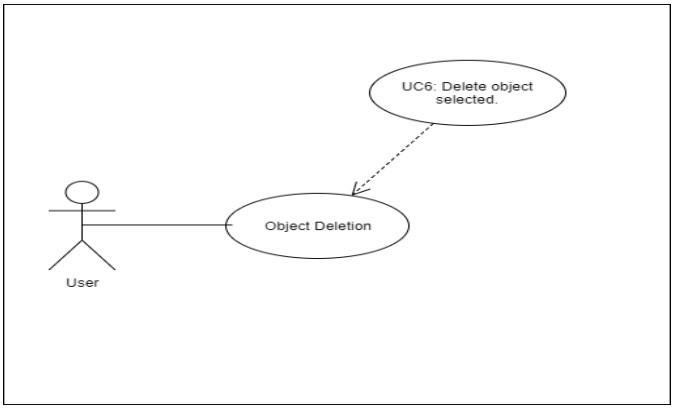
Precondition: The user is inside an environment.
Postcondition: The selected object is spawned in front of them.

- Object Cloning:



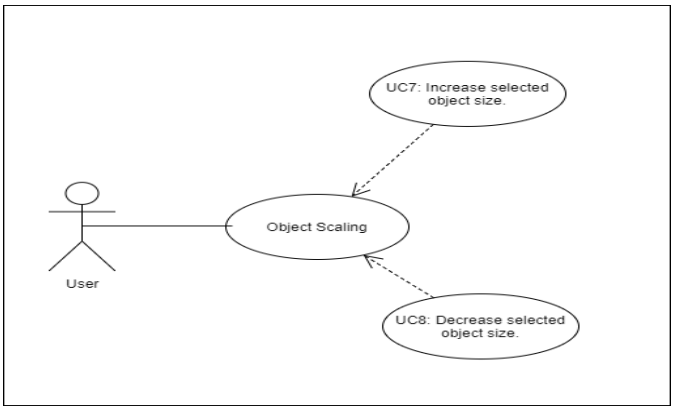
Precondition: The user selectes the specific object to clone.
Postcondition: A duplicate of the selected object is spawned.

- Object Deletion:



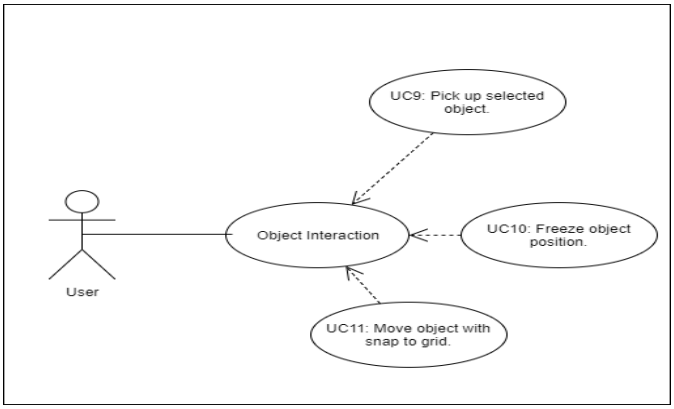
Precondition: The user selects the specific object to delete.
Postcondition: The object is removed from the environment.

- Object Scaling:



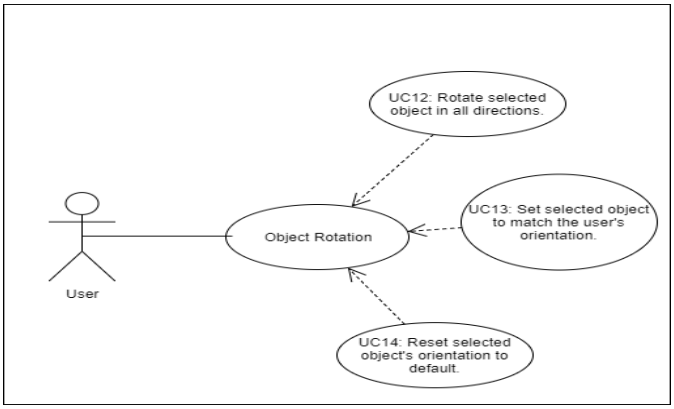
Precondition: The user selects the specific object to scale.
Postcondition: The object is scaled up or down.

- Object Interaction:



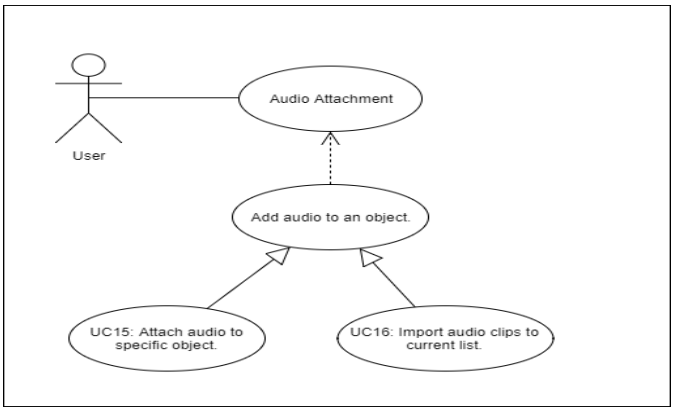
Precondition: The user selects the specific object.
Postcondition: The object is picked up.
Postcondition: The object is frozen in it's current position.
Postcondition: The object is moved using the snap to grid movement.

- Object Rotation:



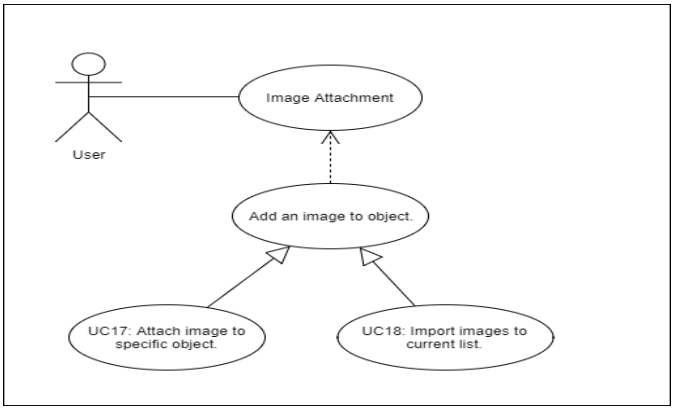
Precondition: The user selects the specific object.
Postcondition: The object is rotated accordingly.

- Audio Attachment:



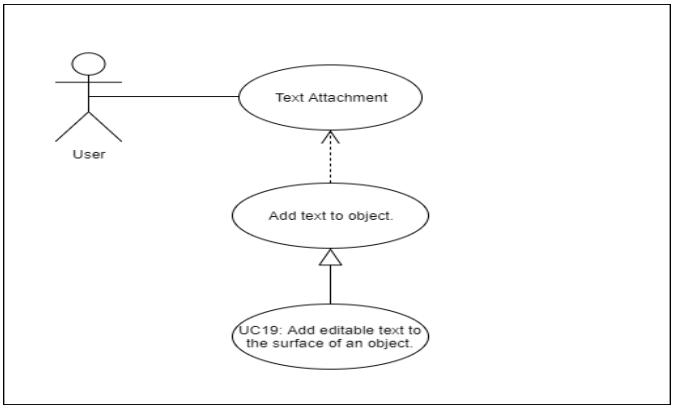
Precondition: The user selects the specific object to whom the user wishes to attach the audio clip to.
Postcondition: The audio clip is attached to that specific object.

- Image Attachment:



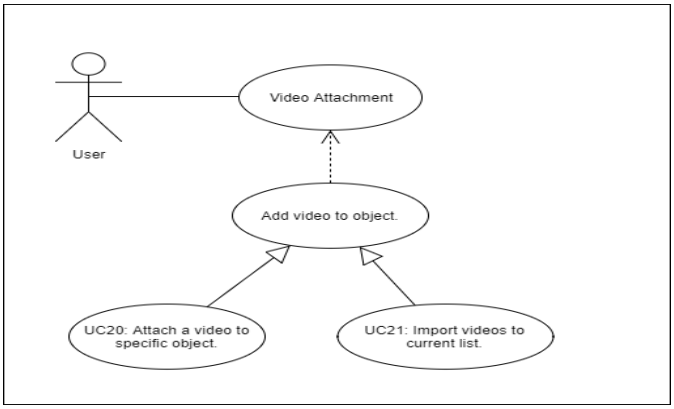
Precondition: The user selects the specific object to whom the user wishes to attach the image to.
Postcondition: The image is attached to that specific object.

- Text Attachment:



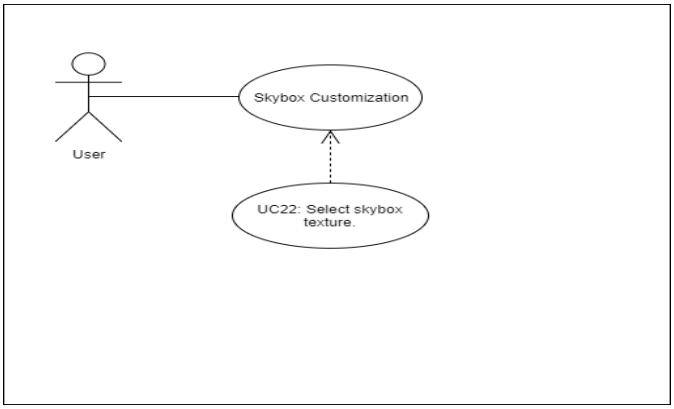
Precondition: The user selects the specific object to whom the user wishes to attach the text to.
Postcondition: The text is attached to that specific object.

- Video Attachment:



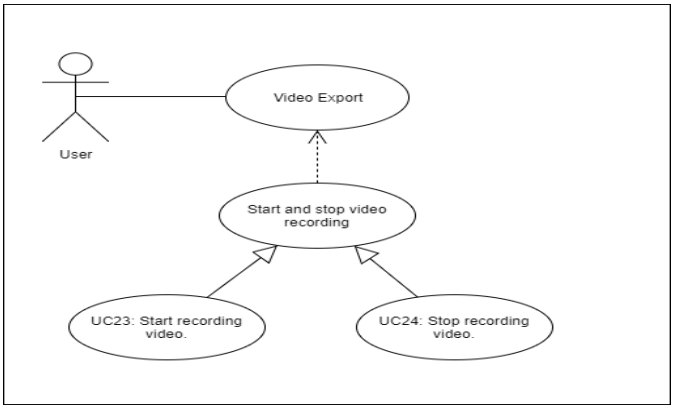
Precondition: The user selects the specific object to whom the user wishes to attach the text to.
Postcondition: The text is attached to that specific object.

- Skybox Customization:



Precondition: The user is inside an environment with a skybox.
Postcondition: The skybox texture is changed.

- Video Recording:



Precondition: The user is inside an environment.
Postcondition: A 360 degree video is recorded and saved within the asset folder.

1.5 Target Audience Characteristics

Initially we planned on focusing our attention on optimizing the presentation software for the education sector, and then extending it to include the corporate world by optimizing a different profile for a business setting. However we later realised that to broaden our target audience, we could design the presentation software to be community driven in terms of content. This would allow virtually anyone to use the software, as well as contribute to an ever-growing community-driven collection of materials, which, of corse, they would also have at their complete disposal.

1.6 Constraints

There are several constraints needed to be taken into consideration.

Platform constraints:

- Mono, an open source development platform based on the .NET Framework. Mono's implementation is based on the ECMA standards for C# and the Common Language Infrastructure.
- For development:
  - Windows 7 SP+1, 8, 10; Mac OS X 10.8+.
- For running Unity applications/games (depending on the complexity of the project):
  - Windows XP SP2+, Mac OS X 10.8+, Ubuntu 12.04+, SteamOS+.

Device hardware constraints:

- Graphics card: DX9 (shader model 3.0) or DX11 with feature level 9.3 capabilities.
- CPU: SSE2 instruction set support.

Video size:

- The exported video should be a realistic size, taking bandwidth and cap into consideration.

Community content needs to be a reasonable size (in community guidelines):

- Contributing to the complexity of a project will increase exported video size.

Community content needs to be relatively optimized (in community guidelines):

- Again, contributing to the complexity of a project will increase exported video size.

Other constraints that will be considered and in which further research will be conducted as implementation progresses include:

- Possible VR device constraints with regards to environment editing.
- Fixed set of templates.
- Unity assets only for community driven content.

1.7 Testing Framework

1.7.1 Introduction

The 3D VR Presentations project presents a unique problem when it comes to testing.
Because the primary goals of the project is to create an intuitive and easy to use interface to create presentations, the assessment will be qualitative in nature.
Thus traditional methods of testing such as unit tests and e2e tests will not be useful to determine if we are creating a viable and working product.
However some parts should still be able to be tested in that way. Further research is needed in order to determine a way of implementing unit and e2e tests and which framework will be used for them.
We are planning on utilizing usability testing at the end of each phase in order to determine how viable our current product is at that stage.
We are also looking into Agile UX Design Principles.

1.7.2 Usability Testing

In short, Usability Testing is a way to see how easy to use something is by testing it with real users.
At the end of each sprint we will test our product in its current state with real users and based on their feedback we will plan adjustments to our development. We will be using knowledge that we have gained from IMY310 to that end.

1.7.3 Agile UX Principles

I have absolutely no idea what to say here. I can't find concensus on any online sources except for what I have already described above.
Could whomever reviews this please assist?

1.8 Technologies

1.8.1 Initial Technologies

In our first meeting with EPI-USE we had discussed the use of various technologies. They had given us "free will" with regards to what technologies to use. We are considering the following technolgies:

- Creating a 3D environment to design and bring to life a 3D presentation.
- Unity 3D virtual reality system tool kit library.
- HTC Vive virtual reality gear (already available).
- Import external models.
- Using plug and play libraries.
- Possibly include library templates for uses to build on.
- Community driven approach.
- Windows 10 environment.
- Docker
- TravisCI.

1.8.2 Technologies to be used

After looking into the above technologies as well as other technologies that are not listed, we have narrowed our choices down to the following:

- Unity Game Engine
  We decided to utitlize the Unity Game Engine because of the massive amount of documentation and support that it has. We also received a lot of recommendations from people who work in the industry to use Unity.

- Unity open-source plugins
  Unity has a lot of community-created plugins that might be useful with our project. Which plugins will be used will be determined as we delve deeper into the project.

- Unity VR library
  Unity has support for both the Oculus Rift and the HTC Vive. This will help should we run into issues with the specific hardware.

- Visual Studio
  We will be using Visual Studio in order to create the interface as it is quite easy to use, and because Unity comes bundled with it.

- Github
  We already have a repository on Github that we are using to collaborate on the project. Github is very user friendly and has a lot of tools that make project management easier.

The following technologies still have to be properly researched before we can commit to using them:

- JsonAPI
  We are possibly looking at JSON as a standard to notate the objects that will be created by our interface. This will allow us to handle objects that have completely different characteristics but are similar in notation structure.

- MongoDB
  We are looking at MongoDB as a way to store the objects it doesn't have a fixed structure. So we dont have to create an entire table and/or column for every type of possible object.

The following technologies were scrapped:

- Docker: Docker does not yet have a proper Windows image and the Unity engine does not support Linux

2 Test Reports

This project posed a few challenges with regards to a suitable testing environment. The best methodology that we found in order to properly test everything we were working on was to use the standard white box and black box approach, while maintaining Unity as the software that will house all of the testing. Usability testing is also another method that will be employed in the project as development progresses. Unit testing will be incorporated where possible.

2.1 Internal: Testing With Team Members

2.1.1 White Box Testing

When testing occurred with members the black box and white box methods were employed at this stage. Team members would first white box test their own work in the Unity engine in order to determine if everything was running as expected. Considering that the majority of the projects code has some form of visual impact, the team member testing their own work will first have to set an outcome that they would expect once their code has been implemented into the project. They would then run the Unity engine and test whether their code behaves the way that they expected or intended. The code produced by each team member was uploaded to their GitHub branch only if it was working in their environment.

2.1.2 Black Box Testing

Once the team member's code is uploaded, they would inform the rest of the team and at least one other member would perform a pull request from their branch. That member will be told what the expected outcome is, thus they would test their code without knowing the inner workings of it. That individual will then run the Unity engine on their local machine in order to test the code thoroughly. After all checks have been passed (does it work as intended, is the functionality indeed correct and does the program behave as expected) they would approve the pull request to master.

2.2 Usibility Testing

Usability testing will involve gathering members with various backgrounds with regards to their computer knowledge. This will allow for a thoroughly test that both advanced and simple features are adequately covered. The testing pool of users will range from roughly five to ten users. They will be asked to perform a set amount of tasks and the time taken for each individual to complete their task will be recorded. They will also be allowed to ask questions if they get stuck. These questions will also get recorded in order to help improve the software's usability (with the individuals consent). After they have completed their task we will ask them for their feedback and how they felt about the software experience overall. All of these results will be taken into consideration.

2.3 Unit Testing

Unit testing will be employed where possible, although this testing will be the least recurring one due to the nature of the project. It will be employed in code where checks need to be, for example. where object checks need to occur (if the object is NULL or not); if the object has certain properties such as an audio component attached to it; if images, audio, video, textures and materials are indeed detected and do fill up their corresponding data structures.

3 User Manual

3.1 Introduction

Welcome to the 3D VR presentation User Manual. This document aims to familiarise users to the included software package, as well as providing a quick start guide for those users who simply cannot wait to create.

What is 3D VR presentation? Imagine taking the art of digital representation to a whole new level. This software enables the user to create and edit a 3D environment that can then be presented to stakeholders as an example, to convay an idea in a much more powerful and intimit way.

Be sure to visit the GitHub repository at Valknut Software Engineering for the latest news and updates on the software.

3.2 Quick Start

- To launch the program, find the VR Presentation executable in the project root folder and double click to run.
- The program should then start and immediately enter full-screen mode.
- You will then be presented with a Main Menu interface that allows the choice of Two initial scenes, an Office scene, and a nature scene.

  The office scene will present you with an example of what the application is capable of in terms of objects in the world, lighting, etc.

  The nature scene has been designed as hands-on tutorial of the type of interactions you as the user can expect, such as spawning, modifying, and cloning objects to name a few.
- Select the nature scene to start with the tutorial.
- You can then use the W A D and S keys to navigate in the world in a first-person perspective.

3.3 Main Scenarios of Use

3.3.1 Features You Can Expect

3.3.1.1 Object manipulation during pickup

Virtually any object visible in the 3D world can be manipulated in terms of scale, texture, etc.

To interact with a nearby object, start by aligning it to the center of the screen, and press the E key to pick up the object and hold it in front of the camera.

You can also drop an object in hand by hitting the E key again.

Whilst the object is picked up, you can interact with it as follows:

- In crease and decrease the object's scale on all axis by using the plus(+) and minus(-) keys found on the numpad.
- Use keypad 1, 2, and 3 to increase the X, Y, and Z axis respectively.
- Use keypad 4, 5, and 6 to decrease the X, Y, and Z axis respectively.
- Use keypad 7, 8, and 9 to rotate the X, Y, and Z axis respectively.
- You can use the R key to Reset the object rotation to 0 x 0 x 0 on its X, Y, and Z axis.
- Press the C key to make a carbon copy of the object currently in your hand.

  Note that the cloned object will retain all of the modifications you made to the original.

- Hitting the G key will enable/disable SnapToGrid to help with accurate alignment.

  Note the object no longer moves smoothly when you move the mouse.

- By pressing the Q key, you can lock the object's orientation to match you precisely, however leaving this disabled will leave the orientation of the object when you first picked it up in relation to your own rotation.

3.3.1.2 Object manipulation without needing to pickup

You can also interact with objects in the world without picking them up:

- You can destroy an object by looking at it and hitting the DELETE key on the keyboard.
- By pressing the Z key, you can apply an image texture to the object in range.

  Note: Pressing Z repeatedly will cycle through a series of images available in the resources directory.

- You can also add an audio file to the object in range by pressing the Z key, which will also by default play the clip to show that it works.
- Feel like adding a video texture to a nearby wall? Walk up to it and press the V key. This will also start playing by default to show that it is working.

3.3.1.3 World manipulation

There is also a feature to completely change the skybox used in the scene.

By pressing the B key, you can cycle through a number of skyboxes, such as a night mode instead of a day skybox.

3.3.1.4 360 Video capture

When you feel you are ready, you can start recording your 360 video and navigate through the world you have created, that can then be exported and viewed in virtually any 360 video supported device, such as an HTC VIVE, VR Gear, etc.

Use the following commands to control video recording:

- Press I on the keyboard to start the 360 recording.
- You can then press O to stop the recording when you are done.
- Pressing P will then open the containing folder of your newly created 360 video.

3.3.2 Examples of Use

3.3.2.1 Rally Sport Hall of Fame

Let's say you want to create a virtual tour of all the greatest moments in Rally sport history.

you could create a virtual hall of fame and have videos showing clips of highlights lined up against the wall in chronological order with a tour guide taking you through this hall and presenting noteworthy facts about each moment in time.

By using this software, you can create such a virtual tour to comemorate the Rally sports and the many drivers who made it what it is today.

3.4 FAQs

3.4.1 Q: How can I insert my own materials for images, audio, and video?

- Navigate to the folder containing the application executable.
- Look for the Resources folder.
- In that folder, you will find an Images, Audio, and Videos folder to add your own content.

  Note: Currently the application only supports Audio files with a .ogg file format, and videos with a .ogv format.