Bachelor thesis

Degree Project in Computer Science and Engineering, 15 credits

Math Visualizations in VR

Computer Engineering, 180 credits

Halmstad 06/03/2023
Abraham Al-bashki, Ahmed Oogle

HÖGSKOLAN
I HALMSTAD

Abstract

Visualization of linear algebra with virtual reality (VR) can be helpful for math teachers. A study has shown more engagement, focus, and active learning through immersion and interactivity, supplementing opportunities for visualizing abstract concepts. This project aims to provide math teachers with an accessible tool for linear algebra concepts in VR. The focus will be on abstraction for the coding by providing a graphical user interface (GUI) as a middle ground, premade virtual objects to reuse, and a manual with linear algebra examples. Furthermore, there are several math visualization tools available for VR. However, there is currently no information about a tool made specifically for math teachers focusing on linear algebra to prepare lessons for VR environments.

Unity is the preferred game engine for this project. The object-oriented design used in this project is polymorphism, Model-View-Controller (MVC), and the command pattern. The system development methodology is extreme programming (XP). The system development methodology is extreme programming (XP). Unit tests are used to drive development; integration tests are used to test between classes; paired t-test is used to test the student's learning with the tool and, additionally, test the experience of using the tool with a usability test. Furthermore, integration tests, usability tests, and paired t-test are used to analysis the result. Moreover, the linear algebra objects is created as data models. A controller uses linear algebra objects to update the GUI and the handles for visual manipulation. Undoing and saving are handled in the controller.

Two simple linear algebra lessons are created by two teachers. The student access the saved lessons in VR and uses handles to interact with the VR world. Moreover, linear algebra expressions of the objects manipulated are shown to the student. The test results showed positive results.

This framework can assist in empowering the creation of three-dimensional (3D) worlds and VR experiences, making them more accessible to a wider variety of educators by removing the requirement that teachers possess coding abilities. As a concept, it can be promising for math teachers to have control over the design of their own VR world without caring about the data science behind it.

Contents

Contents

List of Figures

1 Introduction

Understanding 3D mathematical structures and their properties is a difficult task to do. Math teachers at Halmstad University are interested in providing an accessible programming interface for visualizing 3D mathematical structures in a VR environment through the University's Digital Laboratory Center (DLC). The development of VR can be applied to improve the mathematics education level [1].

In this project, we intend to develop a framework that makes it possible for math teachers with no programming experience to create VR interactive worlds where to explore concepts from linear algebra. Teachers will be provided with a GUI so that they do not have to write code.

Users of the framework are math teachers that will be able to design the VR tools. Students or pupils will, in turn, be able to immerse in a world that includes linear algebra objects and hopefully gain a better understanding (see fig 1).

1.1 Problem statement

The traditional way of teaching 3D mathematical structures is challenging to prepare and present to students [2]. According to Boyle [3], VR improves students' engagement and focus, encouraging students to become active learners through interactivity and immersion and also supplementing students' opportunities to be explored in a class by visualizing abstract concepts. There are many online tools for visualizing higher-dimensional objects, such as Mathpix [4] or CalcPlot3D [5]. In these tools, a 3D object is projected onto a two-dimensional (2D) display. The math teachers would like a tool that acts as a proof of concept to prepare visualizations and make experiments of their lessons in a VR environment, and the DLC at the university of Halmstad provides the headset required to make this tool.

1.2 Purpose

This project aims to develop an accessible tool for math teachers to create visualizations of their lessons on linear algebra in a 3D environment with fully immersive VR. Ideally, if the result of this project is as expected, it can be included in DLC's offer to school and University teachers.

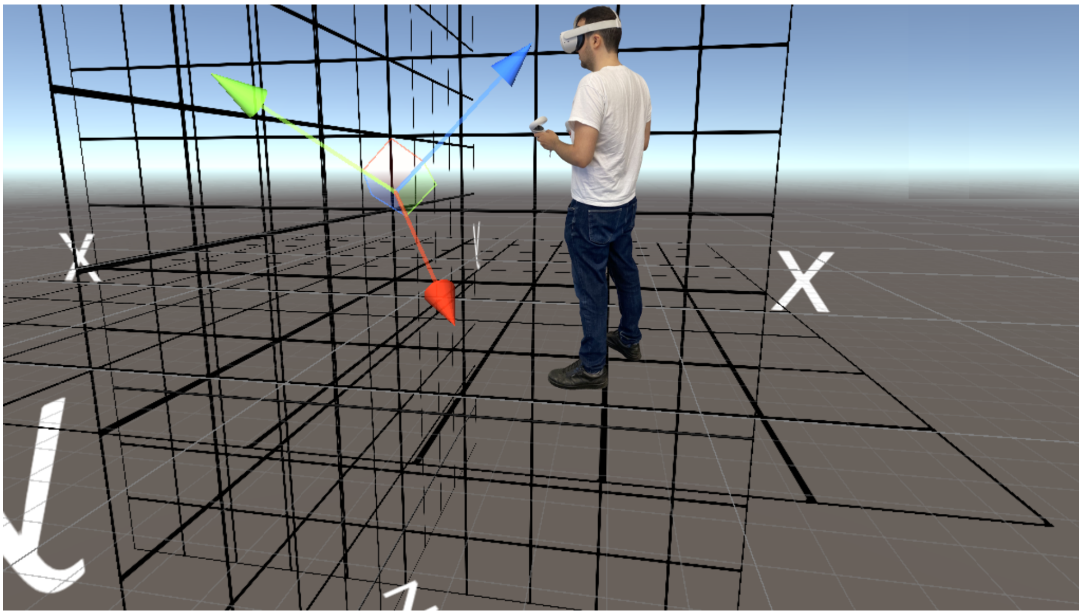


Figure 1: An example image of a conceptual use of VR for linear algebra in the perspective of a student or pupil. The image was made with Unity3D.

1.3 Goals

The first goal is to create a user-friendly interface. This interface will provide abstractions converted from complicated and long steps into simple buttons, sliders, and input with no care for the programming language. The GUI is able to affect the code, and the code brings useful options that are aligned with linear algebra and VR from the perspective of an educator teaching math without any prior coding experience. The options should provide the teaching of math with a way to complement the lessons with a VR experience.

The second goal is to create ready-made components, giving teachers a way to instantiate objects that would typically be used for their lessons. This is to make it quicker and smoother to use the tool with only teaching math in mind.

The third goal is to create a manual for teachers to familiarize themselves with the interface quickly through concise descriptions and the use of examples. This goal is also dependent on the first goal for the manual to be as clear as possible.

A usability test will be done at the end to check if the goals have been achieved. The goals depend much on the opinion of the users that teach math to students or pupils.

The project goals can be condensed to

- a GUI for math teachers to create VRs for linear algebra,
- a library of components that math teachers can reuse, and
- a user guide with examples.

1.4 Requirement specifications

Given that the tool will be deployed in DLC, we need to target Oculus Quest 2.

The framework used to build the GUI for teachers should provide the following:

1. The operations between vectors such as addition, subtraction, dot product, and cross product.
2. The operations between vectors and matrices, such as addition, subtraction, and matrix multiplication.
3. A scaling feature to resize the virtual objects uniformly or non-uniformly in real-time.
4. A rotation feature to rotate the virtual objects in real-time.
5. A relocation feature to relocate the virtual objects in real-time.
6. Creating a default camera with its rotation according to the six degrees of freedom that are rotating along the x-axis (pitch), rotating along the y-axis (yaw), rotating

along the z-axis (roll), moving along the x-axis (sway), moving along the y-axis (heave), and moving along the z-axis (surge).

7. Able to choose different types of manipulation for linear algebra objects such as a collection of sliders, a collection of buttons, or just manually controlling.
8. Toggle on or off to show the coordinates of linear algebra objects.
9. Creating shapes by putting dots (vertices) in space.
10. Creating lines with the help of linear functions.
11. Creating animated linear transformation with the press of a single button.

The library of components should include the following:

1. Premade 3D coordinate systems.
2. Premade dots, lines, and vectors.
3. Premade virtual objects of common solid geometry such as a cube, a sphere, and more.
4. Some premade virtual objects from the real world, such as a house, a mountain, some different biomasses, and more.

The user guide should include the following:

1. A description of what every feature in the GUI does.
2. A description of how to access and use premade virtual objects.
3. Example of how to create animated linear transformations of vectors relative to a 3D coordinate system.
4. Example of how to generate planes in a 3D coordinate system and use them to show concepts of linear algebra.
5. Example of how to generate virtual common solid geometry in a 3D coordinate system and use them to show concepts of linear algebra.
6. Example of how to generate virtual objects from the real world in a 3D coordinate system and use them to show concepts of linear algebra.
7. Example of how to change to another coordinate system of a virtual object while retaining the coordinates.
8. Example of how to change to another coordinate system of a virtual object while retaining the direction.

1.5 Limitations

Although this project is about visualizing math in VR, the attention is only focused on linear algebra. The developed framework will act as a proof of concept and, therefore, may not have the technical requirements and precision for professional use. The framework will be tested by math teachers, and they can evaluate their classes with VR, while the project team will evaluate the framework and the GUI. The project team will not evaluate the use of VR in classes.

Moreover, only translation, scaling, and rotation will be covered, even if linear transformations have reflection, vertical/horizontal contraction and expansion, and even projections. Vertical / Horizontal Shear is also a linear transformation, but it is a composition of vertical/horizontal expansion and rotation.

2 Background

2.1 Related work

There is a VR tool called Calcflow [6], for higher-level mathematics students. Calcflow features include visualizations of vector addition and vector calculus graphs, cross product, parametrized functions, spherical coordinate mapping, and double and surface integrals [7].

Calcflow has a 3D parametric graphing utility and several other tools where users can explore example graphs or input their own. It helps students to visualize and interact with mathematical structures and graphs in a 3D coordinate system. Calcflow is available on the HTC Vive headset [8], but not on Oculus Quest 2. To run Calcflow on Oculus Quest 2, the Oculus VR software must be installed on a compatible PC and connected to the headset using a Link cable.

Another VR tool available on Oculus Quest 2 App Lab is CalcVR (Calculus in VR) [9]. CalcVR uses VR to learn in-depth concepts about Calculus in 1,2, and 3D as well as Analytic Geometry. It covers graphs, curves, surfaces, vectors, and functions in several variables. This VR tool creates an environment ideal for exploring 3D objects encountered in multivariable calculus. The material offered by CalcVR may include lessons, playgrounds, activities, explorations, and Quizzes.

CalcVR is also available on the Google Play Store and iOS App Store. The CalcVR smartphone app uses a Google Cardboard headset [10], that allows users to visualize multivariable calculus abstractions within a VR environment. The tool also has premade modules to help students understand crucial multivariable calculus ideas in VR.

Even though there are several math visualization tools available in VR, some of them mentioned above, there is no information about a tool made specifically for math teachers focusing on linear algebra to prepare lessons for VR environments. This project requires using the Oculus Quest 2 headset to show visualizations, which makes it target an even more specific objective.

2.2 VR

VR is a simulated environment where users can explore and interact with a 3D world. One can experience VR with a VR headset, a head-mounted device that displays images of a computer-based model of the environment in real time. VR headsets often come with hand controllers, which enable users to interact with objects within a VR tool.

Different VR headsets are available; some are fully standalone, not requiring any external devices to function, while others require a tethered connection to a PC or a console. Sony PlayStation VR [11], and HTC VIVE Pro [12], are tethered VR headsets that can operate only when connected to requirement-fulfilling external devices. This guarantees to deliver a high-quality VR experience. Oculus Quest 2 [13], is a standalone VR headset that is a great option for users with no high-powered PCs.

There are three main types of VR: Fully Immersive VR [14], Non-Immersive VR [15], and Semi-Immersive VR [16].

The fully Immersive VR combines real and virtual elements and allows users to use their physical motions to interact with objects in the virtual environment, often with a VR headset. Non-Immersive VR allows users to interact with a virtual environment through a mouse or joystick where they can control activities or characters. Semi-Immersive VR is between immersive and Non-immersive VR; an excellent example of Semi-Immersive VR is CAD systems.

Oculus Quest 2 is a fully immersive standalone VR headset created by Meta. The headset has two touch controllers that can track their location in 3D space. It allows users to be immersed in a 3D experience and roam freely around digital play spaces. Oculus Quest 2 runs on a Quest system software based on Android 10 source code [17].

2.3 Development environment

Visualizing 3D math in VR requires development tools for creating a 3D interactive simulation adapted into VR. There are many options available for this purpose. However, this section will describe the most well-known alternatives, from low-level solutions to game engines catering to the VR industry.

2.3.1 WebXR, WebGL and OpenGL

WebXR is the low-level solution for bringing VR into web browsers. However, WebXR API can neither get models to be managed nor loaded and can neither render nor texture models. The 3D rendering technology in web browsers can be provided elsewhere, such as with the low-level graphical APIs WebGL and OpenGL. Furthermore, it is recommended to use OpenGL together with WebGL's virtual camera functionality [18],[19]. OpenGL is a graphical API and involves less complicated math, data management, and other complex tasks than WebGL [18],[19]. Moreover, WebGL is a JavaScript API based on OpenGL for Embedded Systems (OpenGL ES) [20],[19], and OpenGL can be cross-platform and is language-independent [19].

2.3.2 Game engines

Game engines may vary from libraries to whole programs [21], and some are easy to understand for beginner developers [22]; their purpose is also to create some sort of simulation with interactivity for the created products to start becoming games [23]. Furthermore, one of the objectives of this project is to create a small library with components to use and reuse; thus, it is appealing to use a game engine to program a library with components that simplify the process of math teachers creating their tools.

There are many state-of-art game engines [24] and these state-of-art game engines can vary in stats [25] from each other as they focus on their strengths at some sort of cost. For example, Cry Engine is superior in rendering [25] and global illumination [26] compared to Unity3D, but cannot cross-platform into VR as Unity3D can [27][28]. With this in mind, the game engines Unity3D, Unreal Engine, Cry Engine OGRE3D, and Godot are compared as the game engines are the top ones in stats [25]. Cry Engine, OGRE3D, Godot, and Unreal have the best rendering capabilities, while Unity3D is one step worse at that [25]. Rendering is when the game engine calls the Graphics processing unit (GPU) through graphical APIs such as DirectX or OpenGL; by taking care of the GPU, the game engine lets us render 3D objects into the scene, and even materials, textures, lighting, and setting up the camera [29]. Cry Engine, Godot, Unreal Engine, and Unity3D have the best physics engine while this time OGRE3D is much worse at that [25]. The physics engine is composed of *collision detection* and *rigid body dynamics*, known as “physics” in the game development community; the physics engine in Unity and Unreal Engine is supported by the middleware PhysX, made by NVIDIA [21]. Although, Unity’s physics engine is optimized for a smoother gaming experience rather than correctness in physics [30].

In Unity, graphical APIs are automatically used, and different graphical APIs can be chosen depending on the application created for the corresponding platform, such as windows, android, and more. Windows-based applications in Unity can choose either DirectX or OpenGL; android-based applications in Unity can choose either Vulkan or OpenGL ES. Another type of low-level solution that Unity supports is OpenXR, which works with every type of VR hardware. However, windows-based, macOS-based, and android-based settings and their corresponding graphical API have to be considered for cross-platforming into VR [31].

2.4 Linear algebra for education

One of subjects in 3d geometry is linear transformations. Transformation is called linear because it can satisfy $f(x_1 + x_2) = f(x_1) + f(x_2)$ and $f(\alpha x) = \alpha f(x)$ all linear transformations can be represented by scaling, rotation, reflection, projection, and combinations of mentioned (see fig 3, & 4).

2.4.1 Linear algebra course at Halmstad University

Linear algebra is a compulsory course in an engineering program specialized in either computer, electronics, or mechatronics. In the course syllabus for the spring 2022 at Halmstad University [32], the course goal is basic knowledge and proficiency in geometric problems. The student shall have knowledge and understanding of formulating mathematically, mainly geometric problems of the sort, in terms of linear expressions and equations. After the linear algebra course, the student should also have the following skills and capabilities:

- To solve a system of equations, presumably linear equations.
- To calculate, with planes and lines that span three dimensions, the distance, projections, area, points of intersection, and more.
- To master matrix calculation.
- To describe and calculate the projections, reflections, and rotations with matrices.
- To find the eigenvalues and eigenvectors and to represent linear transformation with diagonal matrix.

Projections, reflections, and rotations are linear transformations, while translation is not. Furthermore, what was not mentioned in the course syllabus for linear algebra, is that uniform scaling is defined as a linear transformation and can be represented by a diagonal matrix. Furthermore, it is not mentioned that some calculations in 3D would use cross product and the dot product that is introduced in the course.

However, it is not the first-time vectors are introduced, both in the first part of physics [33] and mathematics in high school. Already, highschoolers learn operations such as scaling, vector addition, subtraction, and even expressing a resultant vector with orthonormal basis vectors [34]. Moreover, an affine transformation is covered in the course book [35].

2.4.2 Linear and affine transformations

In linear algebra, affine and linear transformations, are mentioned [35]. Transformation is about transforming a collection of points into something, while linear transformation has more definitions attached to it. The affine also means it does not cut through the origin. Mastering matrix calculation is essential in using linear transformations.

The affine transformation preserves the points that form a line to remain points in a line and preserves the ratio of distances. Additionally, linear transformation fixes the origin. The linear transformation can be represented as follows:

$$TP = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1}$$

$$Q' = TP \tag{2}$$

$x, y, z, a, b, c, d, e, f, g, i$ are real numbers. P and Q' are 3x1 matrices that represent coordinates, and T is a 3x3 matrix. Equation 1 shows that the origin will remain fixed if a zero vector is transformed compared to equation 6, which depends on an additional term to get the zero vector.

The property of only multiplication makes it possible to combine a single matrix from any combination of rotation and scaling matrices, thus making it possible to transform any coordinates with one matrix containing the same set of scaling and rotation.

The affine transformation includes rotation, scaling, and translation. The affine transformation is very similar to linear transformation but can be used to express a displaced plane in 3D space [35]. The mathematical equation of affine transformation can be expressed as follows:

$$x' = ax + by + cz + j \tag{3}$$

$$y' = dx + ey + fz + k \tag{4}$$

$$z' = gx + hy + iz + l \tag{5}$$

$$TP + \vec{v} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} j \\ k \\ l \end{bmatrix} \tag{6}$$

$$P' = TP + \vec{v} \tag{7}$$

x', y', z', $x, y, z, a, b, c, d, e, f, g, i, j, k, l$ are real numbers. P and P' are 3x1 matrices that represent coordinates. $\vec{v}$ is a matrix 3x1 that represent vector and T is a 3x3 matrix. In equations, 3 to 5, the symbols x', y', z' represent the new coordinates and correspond to point P', see equation 7. In Equation 6, the transformation matrix corresponds to the symbol T, the $(x, y, z)^T$ corresponds to point P, and the $(j, k, l)^T$ corresponds to vector $\vec{v}$. $\vec{v}$ represents translation as it can move P away from the origin, while the transformation matrix can represent rotation, scaling or something else.

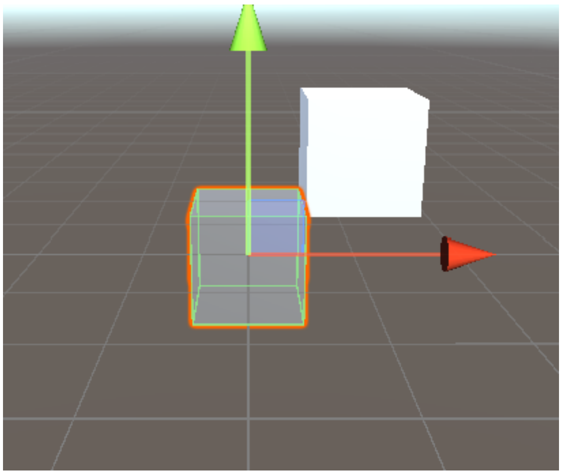


Figure 2: Translation. The gray opaque cube shows the state before an arbitrary translation, and the white cube shows the state after the same translation. The image was made with Unity3D.

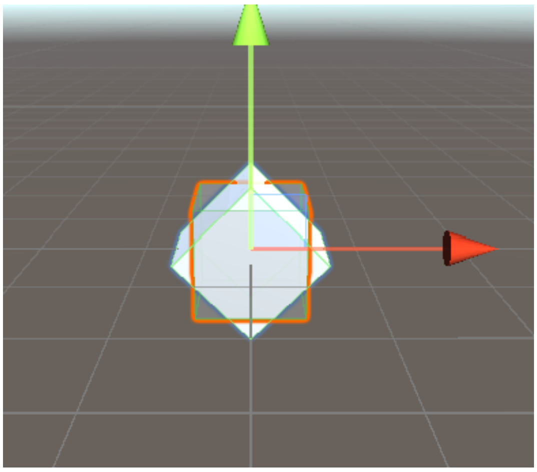


Figure 3: Rotation. The gray opaque cube shows the state before an arbitrary rotation, and the white cube shows the state after the same rotation. The image was made with Unity3D.

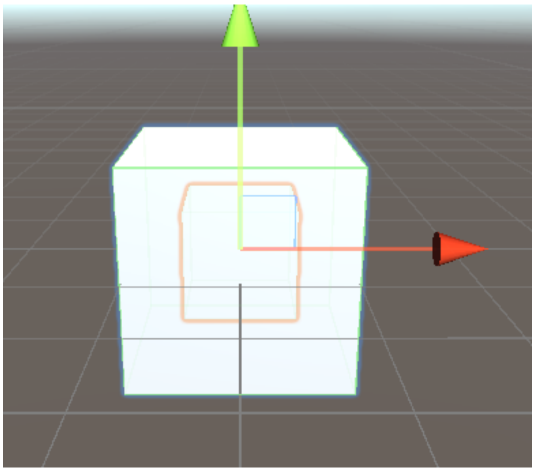


Figure 4: Isotropic scaling. The cube inside the larger cube shows the state before an arbitrary scaling, and the large white cube shows the state after the same scaling. The scaling in this case is uniform. The image was made with Unity3D.

2.4.3 Isotropic scaling

Isotropic scaling is one way of saying uniform scaling (see fig 4). Scaling is done by a single value, where its matrix form can be represented by a diagonal matrix or simply just a scalar value [35]. The scaling part is represented in the following:

$$P' = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & k \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

$$P' = \begin{bmatrix} k \cdot x \\ k \cdot y \\ k \cdot z \end{bmatrix} \tag{9}$$

The variables x, y, z, k are real numbers. The symbol P' is a 3x1 matrix that represent the new coordinates.

2.4.4 Rotation

In linear algebra, two-dimensional rotation can be represented by a matrix, which only needs a single angle, but this changes in 3D. 3D rotation (see fig 3) needs to be represented by three matrices because each axis has one matrix. Furthermore, the 3D rotation matrices can be combined, but it does not change that each rotation needs three angles, also called the Euler angles.

There are many kinds of 3D rotation with Euler angles because the rotation comprises three basic rotation matrices, and the order matters. Each respective Euler angle represents the angle around an axis perpendicular to the axes represented by the other two

Euler angles. The rotation part is represented in the following:

$$R(\alpha, \beta, \gamma) = R_y(\alpha) \cdot R_x(\beta) \cdot R_z(\gamma) \tag{10}$$

$$R(\alpha, \beta, \gamma) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$$R(\alpha, \beta, \gamma) = \begin{bmatrix} \cos(\alpha)\cos(\gamma) + \sin\sin(\beta)\sin(\gamma) & \cos(\gamma)\sin(\alpha)\sin(\beta) - \cos(\alpha)\sin(\gamma) & \cos(\beta)\sin(\alpha) \\ \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma) & -\sin(\beta) \\ \cos(\alpha)\sin(\beta)\sin(\gamma) - \cos(\gamma)\sin(\alpha) & \cos(\alpha)\cos(\gamma)\sin(\beta) + \sin(\alpha)\sin(\gamma) & \cos(\alpha)\cos(\beta) \end{bmatrix} \tag{12}$$

The variables α, β, γ are real numbers. The z, y, and z notations represent the local axes in 3D. The symbol R is a 3x3 matrix that represent the rotation matrix. In equation 10 and 11, the sequence of the $z - x - y$ rotation combination is read from right to left in matrix form. A single rotation matrix can represent any sequence of the twelve Euler combinations.

2.4.5 Translation

The translation moves a collection of points to a different position with coordinates relative to the origin (see fig 2). The translation part is represented in the following:

$$P' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} j \\ k \\ l \end{bmatrix} \tag{13}$$

$$P' = \begin{bmatrix} x + j \\ y + k \\ z + l \end{bmatrix} \tag{14}$$

The variables x, y, z, j, k, l are real numbers. The symbol P' is a 3x1 matrix that represent the new coordinates.

3 Method

Many different methods and tools can be used in the project regarding designing, programming, testing, planning, and steering the project itself. In this project plan, we present some of these choices.

3.1 Methodology

This section goes through the choice tools for development, object-oriented design, the workflow of the project team, and the resources for this project.

3.1.1 Choice of development tools

Game engines automatically use graphical APIs and will not be bothered by the complexity of a graphical API such as OpenGL. There are free game engines for educational use [[36], [37], [38]], but the project team has two reasons for selecting a game engine. First, the game engine synergizes with VR technology; secondly, it conforms with cross-platforming towards most, if not all, vendor-specific VR sets. The cross-platforming allows better reach to others. Unity3D, Unreal Engine, and Godot Engines fulfill these by supporting the low-level solution, the OpenXR API. However, Unity3D will instead be used in this project for two reasons. Unity3D can use C# scripting language, similar to Java [39] [40] which is familiar to the project team. Unity3D has also long been in its branch, but less than Unreal Engine [41]; still, Unity3D has enough ground in this field to be relatively problem-free in its framework. Godot Engine fulfills the first reason [42], but not the second one. Consequently, as relatively new, Godot Engine has fewer third-party guides and tutorials [43] and fewer academic articles than the others.

Furthermore, Unreal Engine also uses the language, C++, similar to the C language familiar to the project team, but memory management is explicitly not needed for this project. Moreover, high-level languages usually do an excellent job of taking care of such low-level operations in the background [44]. Moreover, for Unity3D we do not have full access to the source code; in Unreal Engine, full access can be given [45]. Unity3D, Unreal Engine, and Godot Engine have good documentation, although Godot Engine does not have an extensive community like the others [43].

Visual Studio will be our Integrated development environment (IDE) for the C# scripting with Unity3D. Blender, an open-source computer graphic software tool, will be used for creating complicated 3D structures for Unity3D.

The hardware used in the VR part of this project will be the Oculus Quest 2 because it is the only available VR set for us. A computer with Windows 11 as the operative system will also be used to run the software development tools.

3.1.2 Choice of object-oriented design

The design pattern used in this project is primarily the MVC pattern. This pattern controls the data flow through the controller, while the view part uses data through the controller when needed. The data model (or data models) is manipulated by the controller and gives its reference to the controller.

Another design pattern used in the project is the command pattern, which encapsulates each change of the objects for linear algebra. The encapsulation of the change in the objects makes it easier to undo changes in chronological order.

Another object-oriented design used in this project is polymorphism. Polymorphism allows different objects to be treated in the same way for the purpose of linear algebra.

3.1.3 Workflow

The choice of system development methodology is XP. XP is a code-centric process intended to improve the programming quality by having multiple short development cycles. XP is based on a specific planning approach, available customers, and continuous testing. There are many advantages of using XP, including error avoidance, stable software, and responsiveness to customer requirements.

3.1.4 Resources

The Oculus Quest 2 is owned by Digital Laboratory Center (DLC) at Halmstad University, which can be booked through our supervisors. A computer for using the software tools, which is decided by the project, is contributed by a group member. If an additional component is required in the project, then a meeting is arranged for all parties involved.

3.2 Testings

Using testing frameworks for testing is prepared through Unity. The assembly definitions are created in the same folders as the scripts in the project folder. Moreover, an assembly definition and a script for testing are created in an isolated folder for testing. The testing assembly definition must reference the NUnit and the NSubstitute assembly files. Furthermore, the assembly definition must reference each other, but circular reference is not allowed. After the setup, the NUnit and NSubstitute frameworks are imported into the script for testing. Further explanation of unit testing, integration testing, and usability testing is explained in the sections below.

3.2.1 Unit testing

Unity has two types of tests: Unity's editor and play mode tests. The one mainly used for unit testing is editor tests because there is no need to play the code and have the game loop be active. Unit testing is also an advantage in keeping track of code changes.

The modus operandi of the project team continuous unit testing will be done as part of the XP (see fig 17). Unit tests are used to drive development. Before any code is written, an automated failing test is created. After some success, the unit-tested code is integrated into the main code of its related goal.

Unit testing is done on every function that can be tested, and modified condition/decision coverage (MC/DC) is enforced. The private fields and private were changed to internal fields and functions because of the InternalsVisibleTo attribute written in the assembly definition. The InternalsVisibleTo attribute allows internal fields and functions to be visible for the class specified, which is a class for testing in this case. Moreover, the NSubtitute framework is used to mock classes that are used as a parameter for other classes.

3.2.2 Integration testing

Unity's play mode testing is chosen to test the integrations between the classes because, most often, the gaming loop has to be active. Furthermore, as the system runs by itself, the tests use the InternalsVisibleTo attribute to check the internal fields and compare them with expected values.

3.2.3 Usability testing

Usability testing begins in the later testing phase. The users testing the teacher's part are foremost with backgrounds in teaching linear algebra and then those with backgrounds that have taken the linear algebra course. However, the questions prepared for the latter target group are limited compared to the prior target group. The testing will be using our tool. The lack of people that teaches math cannot be avoided and gives a less accurate evaluation; thus, some margin of error must be considered.

The manual created for the framework is helpful for evaluating the usability test. The definition of a usability test can be cut into five parts [46]

- effective,
- efficient,
- engaging,
- error-tolerant,
- easy to learn.

The respective definitions are

- effective in the way of how accurately the framework is used;
- efficient in the way how quickly the use of the GUI, for the user, can be completed;
- engaging in the way of how well the GUI draws the user into the interaction and how pleasant and satisfying it is to use;

- error-tolerant in the way of how well the framework prevents errors from the user and can help the user recover from mistakes if it occurs;
- easy to learn in the way of how well the framework supports the initial learning for the user.

The efficient part could be timed. However, there is no way to compare with other users the efficient part of what is quick or not, which will then be formulated as questions for the user; the questions can be compared with the user's experience of using similar tools. Moreover, the engaging and the easy to learn parts can be formulated into questions. Furthermore, the error-tolerant part can be partly tested with unit testing and integration testing and partly formulated into questions for the user.

3.2.4 Paired t-test

Paired t-test is a statical procedure used to estimate whenever there is a difference between the population means measured before and after treatment on the same population. Additionally, other types of approaches define as paired t-tests, but those are not used in this report. The paired t-test begins with a null hypothesis (H_0) and an alternative hypothesis (H_1). The null hypothesis is assumed true and is always an equation claiming that the difference between the population means is zero, meaning the treatment had no effect (see eq 15 and 16). However, the calculation for the difference between the population means is replaced by the following difference in sample means ($\overline{x_{diff}}$), the assumed difference of the population means from the null hypothesis (see eq 15), and the standard deviation of each sample mean ($\overline{s_{diff}}$).

$$H_0 : \overline{\mu_{diff}} = 0 \tag{15}$$

$$H_1 : \overline{\mu_{diff}} > 0 \tag{16}$$

The paired t-test fulfills all the interference conditions by having the following sample size of 30 or more, the sample size is 10% less than the population, and the sample is random [47]. In equations, 15 to 20, the $\overline{\mu_{diff}}$ is the difference between the population means before and after using treatment. The $\overline{\mu_{diff}}$ is calculated using the data set after treatment as a minuend and the other data set as a subtrahend. Moreover, equation 16 assumes $\overline{\mu_{diff}}$ is greater than zero and thus considers the upper tail when using the t-distribution table with a degree of freedom based on sample size (n) minus one. Moreover, the significant level (α) is usually set to 0.05 [48].

$$\overline{x_{diff}} = \frac{\sum_{i=1}^{n} x_{diff_i}}{n} \tag{17}$$

$$\hat{s}_{diff} = \sqrt{\frac{\sum_{i=1}^{n} (x_{diff_i} - \overline{x_{diff}})^2}{n-1}} \tag{18}$$

$$\hat{s}_{diff_x} = \frac{\hat{s}_{diff}}{\sqrt{n}} \tag{19}$$

$$t = \frac{\overline{x_{diff}} - \overline{\mu_{diff}}}{\hat{s}_{diff_x}} \tag{20}$$

$$p < \alpha \tag{21}$$

In equations 17 to 20, the sample mean is $\overline{x_{diff}}$, the standard deviation of the sample is $\hat{s}_{diff}$, the standard error of the sample mean is $\hat{s}_{diff_x}$, the sample size is n, and the t-score is t. The equations 17 and 18, use summation that sums up the iterations in x_{diff_i} based on how much n that i increments. The equation 21 uses the α for the significance level and p corresponds to the p-value.

To conclude the paired t-test, the p-value has to be obtained and compared with the significance level. The p-value is obtained with the help of using the calculated t-score to find the p-value in the t-distribution table with n–1 degree of freedom. Furthermore, if the p-value is less than the significance level, the null hypothesis is rejected in favor of the alternative hypothesis. Meaning the result of the paired t-test is statically significant.

3.3 Result analysis

The analysis is done to measure if the requirement specifications were achieved with the available tools for testing, as mentioned in section 1.4. The following requirement specifications of the tool are listed down below, and how to test is written afterward as follows:

- The GUI for the teacher should provide the cross-product, vector addition, vector subtraction, and dot-product.
- The GUI for the teacher should provide matrix multiplication, matrix addition, matrix subtraction, translation, scaling, and rotation.
- The library of components that teachers can use are premade dots, lines, vectors, common solid geometry, and objects from the real world.
- The user's camera has six degrees of freedom.

The first and second points are tested with an integration test, respectively. The integration tests have the correct result beforehand that compares to the results produced by the tool by comparing the values. Furthermore, an integration test also tests the third point. The third integration test checks if the GUI can instantiate the object type correctly by comparing it with the expected object type. The third test also compares the increased number of objects with the expected. At the last point, a question related to the difficulty of using the camera is prepared in the usability test.

The following specifications of the instruction manual are listed down below, and how to test is written afterward as follows:

- The user guide should include examples of generating common solid geometry in a 3D coordinate system.
- The user guide should include examples of generating objects from the real world in a 3D coordinate system.

The usability testing will be done on the participants, and the questions prepared will also include the difficulty in doing these examples or lessons.

Furthermore, the students will be testing the lessons that the teacher designs to test their engagement with the help of VR (see section 1.1). The validity of the test is determined by a paired t-test using the null hypothesis and the alternative hypothesis, respectively (see eq 15 and 16). The students will be taught the lessons' content without VR and then be questioned. They will later enter the designed VR lessons and be asked the same questions as the previous ones after leaving VR. A paired t-test evaluates the difference in understanding the lessons between augmenting and not augmenting with VR. The sample size for this test is 30.

When the questioning is done, the feedback is collected and calculated to get the p-value (see section 3.2.4). The p-value is compared with α, and if the p-value is less than α, then the H_0 is rejected in favor of the H_1. When the H_1 wins, then it is statically significant that VR improves the understanding of simple linear algebra lessons (see Appendix E). The improvement of understanding simple linear algebra lessons with VR indicates VR helps the students engagement and focus.

4 Implementation

This section presents the implementation of the tool. It starts off the first subsection with what is to be considered and used in Unity. The following subsections explain the coding process and the object-oriented design used in different cases. Figure 6 shows a UML diagram showcasing the classes used for the project.

4.1 Getting started with Unity

Starting Unity begins by showing the scene where the objects called game objects are put in and rendered dynamically. There is also a project folder with entities called assets that can be used in the scene but are stored outside the scene till further use. An asset can be imported 3D models, textures, sprites, sounds, and scripts. Furthermore, every entity in the scene is a game object or a composition of game objects; game objects are containers for attaching components. Moreover, a game object always has an attached Transform component.

4.1.1 Unity hierarchy system

The project often uses the hierarchy system to create compositions of game objects that only need to have the root game object controlled for the rest to be automatically managed. The hierarchy reflects a parent-child relationship in that the parent determines the following position, rotation, and scaling relatively.

The hierarchy enforces a change of basis where the new origin is the center point of the parent relative to its children. Consequently, the children follow the parent; when it does, the child moves along. Furthermore, the parent-child relationship also affects the scale and the rotation. When the child's local rotation is zero, it will follow the parent's rotation without changing its local rotation values. Moreover, when the child's local scale is one and its parent's scale changes, then the child has the same scale without its local scale changing.

4.1.2 Components used in the project

Many components can be chosen to be attached to a game object [49], but there is always a Transform component attached to a game object. The Transform component has access to methods such as setting the child and parent relationship, changing the Euler angles, the local scales, the position, and some other useful ones. In the script, the Transform component also takes in Quaternion values to change its rotation [50].

Getting a game object rendered in 3D to something visible, then a Mesh Renderer together with a Mesh Filter component are needed. The Mesh Filter has premade assets such as a cube, sphere, and more. Moreover, the vertices of the Mesh can be accessed through the Mesh Renderer. The vertices are a collection of points representing the Mesh, which can have transformation applied mathematically. However, applying translation on the vertices will move the game object's center point, and another downside is that the changes are not reflected in the Transformation component [51][52]. On top of the two mentioned components, another component called Line Renderer can be used to draw lines in the scene [53].

Another component used in the project is that Unity has Box colliders and Mesh colliders, and there are separate colliders, but the most used ones are the two first types of colliders. The box collider creates a cuboid around the game object, and this area can be used to detect hits from a method that uses ray cast, a “ray” that the camera sends out in the direction of where the camera is “looking.” Mesh collider is a more expensive type in performance, but it considers the game object’s Mesh [54].

Lastly, a practical component is a constraint type, which enforces some constraints on the attached game object based on a targeted game object. The constraints can be only scaling, rotation, position, and even a constraint that forces the game object attached to aim its local forward vector towards the targeted game object [55].

4.1.3 The MonoBehaviour base class and the event functions

The MonoBevaiour class is a base class that every script inherits, and that is required for scripts to attach themselves as a component to a game object [56]. Scripting in Unity does not involve the typical main method as a starting point, but instead, Unity calls event functions in a so-called game loop (see fig 5). A script inheriting MonoBehaviour can implement these special event functions in the game loop. Unity calls these event functions in a predetermined order and other event functions in scripts inheriting MonoBehaviour. However, what is not predetermined is that the event functions of the same type will be called in some random order. Meaning during the initialization stage, the scripts dependent on other scripts cannot be in the first event function, Awake [57]. The downside of MonoBehaviour is that scripts deriving from it cannot be called through the constructor. The AddComponent method can add a script deriving Monobehaviour to a game object and get the Instance back [56]. However, the use of constructors allows for encapsulation.

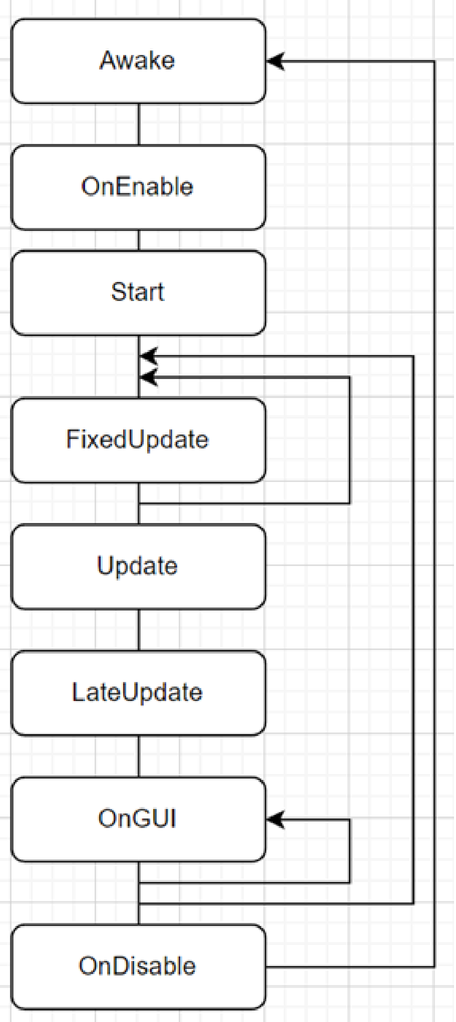


Figure 5: FixedUpdate can happen more than once per frame depending on the low frame rate; the FixedUpdate is for physics. The initializing happens at Awake, OnEnable, and Start. LateUpdate is called after the Update. OnGUI is called multiple times during the frame update. Update is called upon once per frame. OnDisable is called only when the game object is decommissioned. The project team made the image.

4.2 Linear algebra object

The concept of a linear algebra object is an object that can have linear transformation and translation applied. The data model of such an object is encapsulated in a class implementing an interface called LA_Object. The interface provides translation, rotation, and non-uniform scaling.

The classes implementing LA_Object represent some shapes, but dimensions as a parameter were introduced to decrease the number of classes representing shapes using the MeshRenderer component. Furthermore, the values of the dimensions are encapsulated in Unity's provided Vector3 data type and are used to apply scalings on the object's x-axis, y-axis, and z-axis, respectively. A parameter with Vector3 data type as input to relocate the object, another Vector3 as Euler angles to change the object's rotation, and one other for color. In contrast, a vector class implements LA_Object because it uses the LineRenderer component; the parameters for the vector object are the start coordinate, the end coordinate, and the color. Moreover, LA_Object also provides additional methods for interacting with a class as a data model part of the MVC pattern.

4.3 Vector operations

Vector operations are accomplished by using polymorphism. The different kinds of vector operations are in a class each and implement the Vector_Operation interface. The interface will provide functions as a part of a data model for the Controller class. The different vector operations can be activated by the GUI. Moreover, the classes implementing the Vector_Operation interface contain two instances of LA_Vector. These are carefully made to not clash with the MVC pattern with LA_Object, Controller, and the GUI by checking the list of the data type LA_Object in the related Controller instance.

4.4 Interaction of linear algebra objects with the MVC design pattern

To manipulate the linear algebra objects educational manner, the Handles class was created. The Handles class provides handles used to manipulate the object through either translation, scaling, or rotation. Moreover, the LineRenderer component was used to create the handles.

The handles for translation are three different colored arrows pointing from the object's center point to their respective axis. Furthermore, the handle for scaling is gray colored and represents a line with a rectangle at the end. Lastly, the handles for rotation have the same color scheme as the translation handles and are represented by a circular line around their respective axes.

The Handles class does not provide interaction and is the view part of an MVC pattern. The one handling the user input is the myLeftRayCast class, but its responsibilities were delegated by the Controller class that derives the MonoBehaviour class. The Controller class is still the class that controls the data flow through Unity's event functions. The data model of the MVC is some class implementing the LA_object, but the target data model for the handles is not static. The user has to press on a potential data model to make the myLeftRayCast class call a method in the Controller class to replace the current target of the data model. The user input can also change the type of handles used.

4.5 GUI for teachers and students

This section goes through the GUI:s used in the project.

4.5.1 IMGUI in the MVC pattern

IMGUI, which stands for (Immediate Mode GUI), is a user interface system used in game development to provide quick and easy GUIs. IMGUI is based on an immediate mode rendering model, wherein UI components are rendered immediately within the game loop. This allows for the development of a quick and code-driven UI system that avoids the burden of managing and developing UI components. With IMGUI, developers can create UI elements such as buttons, text fields, and sliders without creating and managing complex UI components and their states. However, IMGUI has some limitations; it needs to work better to develop complicated user interfaces or support dynamic layouts.

The GUI is part of an MVC pattern, and it shows the values of the data model in the GUI. The buttons are triggered by the user, but the data flow will still go through the Controller class.

4.5.2 uGUI in the MVC pattern and as a stand-alone

The uGUI works in VR, which is why it is used. The uGUI can be either instantiated many times through the IMGUI or one time when the Controller class is instantiated. In the latter part, an instance of the LA_VRCanvas is created, and it works the same way as Handles class, except there is no interaction directly. Furthermore, the LA_VRCanvas will update its text on each change of its targeted LA_Object.

4.6 Undo and save

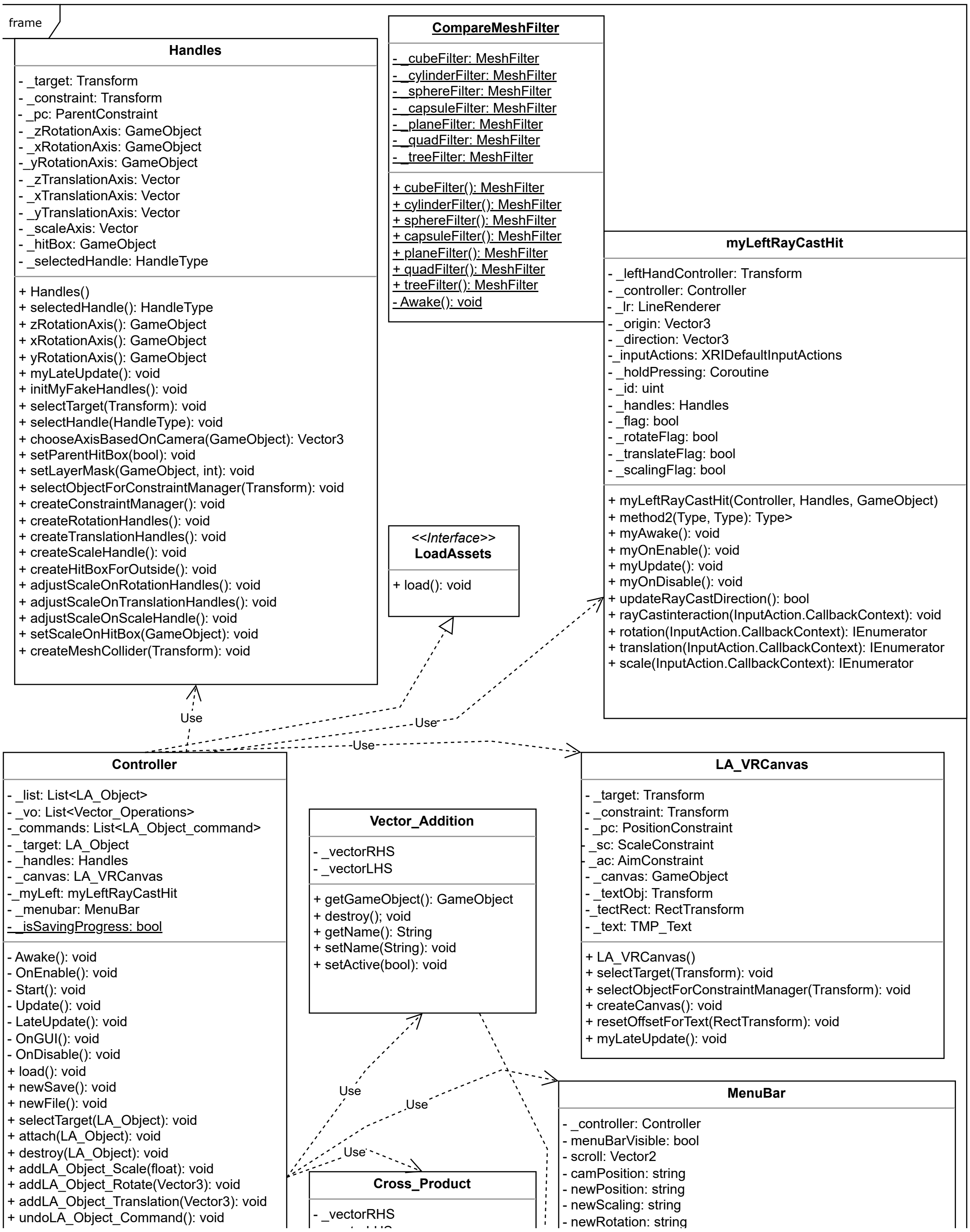
The functionalities of undoing and saving lie partly in the Controller class because the data flow is also there. The saving and the undoing are done with buttons. The save and undo buttons in the GUI are created by the MenuBar class, which uses IMGUI and is also a part of an MVC pattern. Further explanation of the undo and save is explained in the sections below.

4.6.1 Undo with the command pattern

As mentioned in section 3.1.2, the command pattern is implemented. The concept is to encapsulate each update of a linear algebra object into classes implementing the ICommand interface. The classes implementing the ICommand all have LA_Object in the parameter for the constructor. Furthermore, the classes implementing the ICommand also have some input corresponding to its purpose, such as scaling using a number as input. The ICommand interface also provides an execute method and an Undo method. The classes only use the execute and Undo methods once in that order.

4.6.2 The save asset interface and load assets interface

When the user presses the saving button, the newSave function in the Controller class is called by the MenuBar class. The newSave function clears the current saved assets before saving all the LA_Objects and all Vector_Operations as assets in the project folder. The saved assets are loaded by the Controller class at the start of when running the code. The controller class implements the loadAssets interface, while the classes implementing LA_Object, and Vector_Operation also implement the saveAsset interface.


frame
Handles
- _target: Transform
- _constraint: Transform
- _pc: ParentConstraint
- _zRotationAxis: GameObject
- _xRotationAxis: GameObject
-_yRotationAxis: GameObject
- _zTranslationAxis: Vector
- _xTranslationAxis: Vector
- _yTranslationAxis: Vector
- _scaleAxis: Vector
- _hitBox: GameObject
- _selectedHandle: HandleType
+ Handles()
+ selectedHandle(): HandleType
+ zRotationAxis(): GameObject
+ xRotationAxis(): GameObject
+ yRotationAxis(): GameObject
+ myLateUpdate(): void
+ initMyFakeHandles(): void
+ selectTarget(Transform): void
+ selectHandle(HandleType): void
+ chooseAxisBasedOnCamera(GameObject): Vector3
+ setParentHitBox(bool): void
+ setLayerMask(GameObject, int): void
+ selectObjectForConstraintManager(Transform): void
+ createConstraintManager(): void
+ createRotationHandles(): void
+ createTranslationHandles(): void
+ createScaleHandle(): void
+ createHitBoxForOutside(): void
+ adjustScaleOnRotationHandles(): void
+ adjustScaleOnTranslationHandles(): void
+ adjustScaleOnScaleHandle(): void
+ setScaleOnHitBox(GameObject): void
+ createMeshCollider(Transform): void
CompareMeshFilter
- _cubeFilter: MeshFilter
- _cylinderFilter: MeshFilter
- _sphereFilter: MeshFilter
- _capsuleFilter: MeshFilter
- _planeFilter: MeshFilter
- _quadFilter: MeshFilter
- _treeFilter: MeshFilter
+ cubeFilter(): MeshFilter
+ cylinderFilter(): MeshFilter
+ sphereFilter(): MeshFilter
+ capsuleFilter(): MeshFilter
+ planeFilter(): MeshFilter
+ quadFilter(): MeshFilter
+ treeFilter(): MeshFilter
- Awake(): void
myLeftRayCastHit
- _leftHandController: Transform
- _controller: Controller
- _lr: LineRenderer
- _origin: Vector3
- _direction: Vector3
-_inputActions: XRIDefaultInputActions
- _holdPressing: Coroutine
- _id: uint
- _handles: Handles
- _flag: bool
- _rotateFlag: bool
- _translateFlag: bool
- _scalingFlag: bool
+ myLeftRayCastHit(Controller, Handles, GameObject)
+ method2(Type, Type): Type>
+ myAwake(): void
+ myOnEnable(): void
+ myUpdate(): void
+ myOnDisable(): void
+ updateRayCastDirection(): bool
+ rayCastinteraction(InputAction.CallbackContext): void
+ rotation(InputAction.CallbackContext): IEnumerator
+ translation(InputAction.CallbackContext): IEnumerator
+ scale(InputAction.CallbackContext): IEnumerator
<<Interface>>
LoadAssets
+ load(): void
Use
Use
Use
Controller
- _list: List<LA_Object>
- _vo: List<Vector_Operations>
-_commands: List<LA_Object_command>
- _target: LA_Object
- _handles: Handles
- _canvas: LA_VRCanvas
-_myLeft: myLeftRayCastHit
- _menubar: MenuBar
- _isSavingProgress: bool
- Awake(): void
- OnEnable(): void
- Start(): void
- Update(): void
- LateUpdate(): void
- OnGUI(): void
- OnDisable(): void
+ load(): void
+ newSave(): void
+ newFile(): void
+ selectTarget(LA_Object): void
+ attach(LA_Object): void
+ destroy(LA_Object): void
+ addLA_Object_Scale(float): void
+ addLA_Object_Rotate(Vector3): void
+ addLA_Object_Translation(Vector3): void
+ undoLA_Object_Command(): void
Vector_Addition
- _vectorRHS
- _vectorLHS
+ getGameObject(): GameObject
+ destroy(); void
+ getName(): String
+ setName(String): void
+ setActive(bool): void
LA_VRCanvas
- _target: Transform
- _constraint: Transform
- _pc: PositionConstraint
_sc: ScaleConstraint
_ac: AimConstraint
- _canvas: GameObject
- _textObj: Transform
-_tectRect: RectTransform
- _text: TMP_Text
+ LA_VRCanvas()
+ selectTarget(Transform): void
+ selectObjectForConstraintManager(Transform): void
+ createCanvas(): void
+ resetOffsetForText(RectTransform): void
+ myLateUpdate(): void
Use
Use
Use
MenuBar
- _controller: Controller
- menuBarVisible: bool
- scroll: Vector2
- camPosition: string
- newPosition: string
- newScaling: string
- newRotation: string
Cross_Product
- _vectorRHS

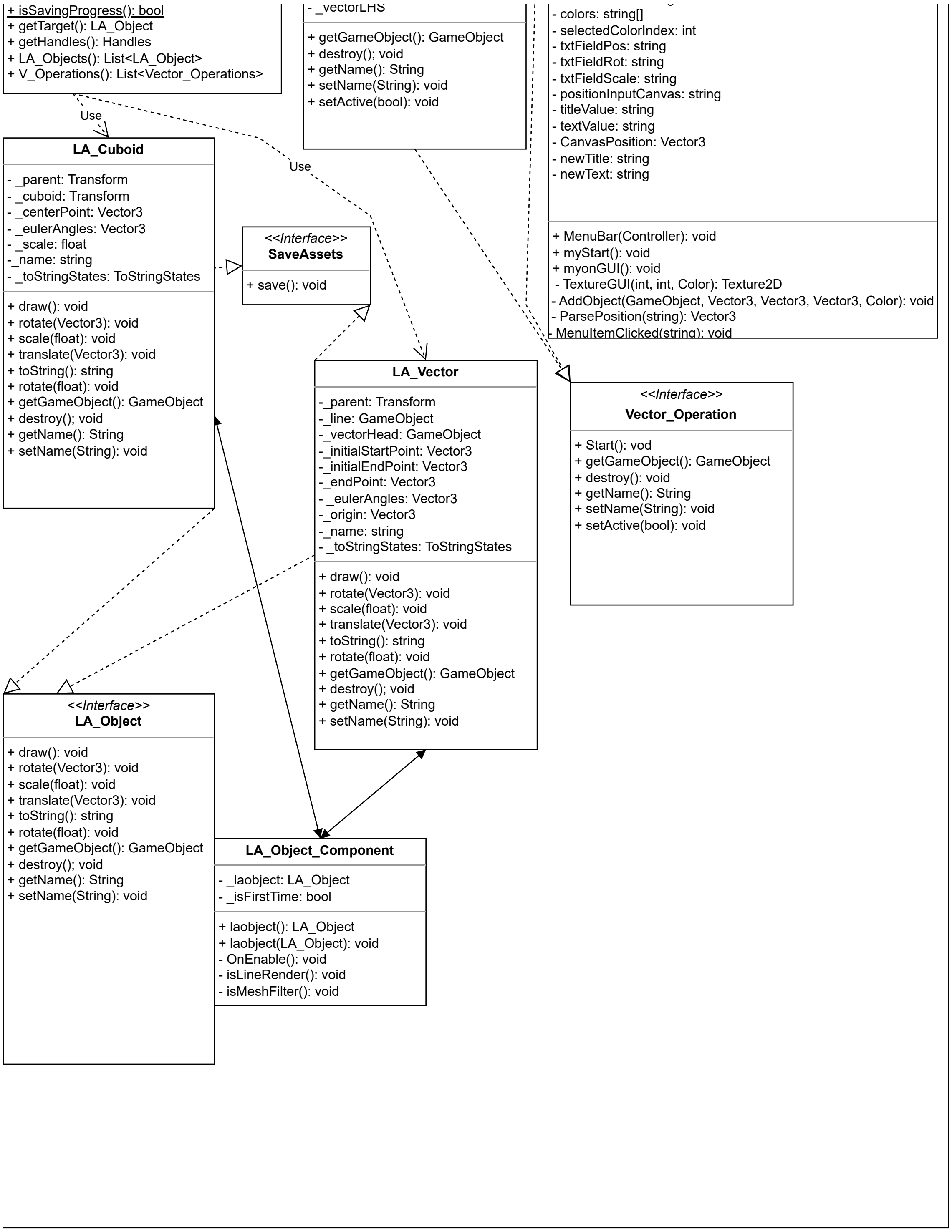


Figure 6: UML Class Diagram showcasing the structure and relationships of classes in the project. Several types of MVC patterns are used in this diagram, and the Controller class is the controller in each case. The classes updated by the Controller class are the Handles class, the LA_VRCanvas class, MenuBar. The class manipulated by the Controller class are classes implementing LA_Object and class implementing Vector_Operation. The UML does not show all the classes that implement LA_Object and LA_Vector-Operation. The class myLeftRayCast handles some responsibility for user input delegated to it by the Controller class. The LA_Object_Component is attached as a component to the created game objects. The project team made the image.

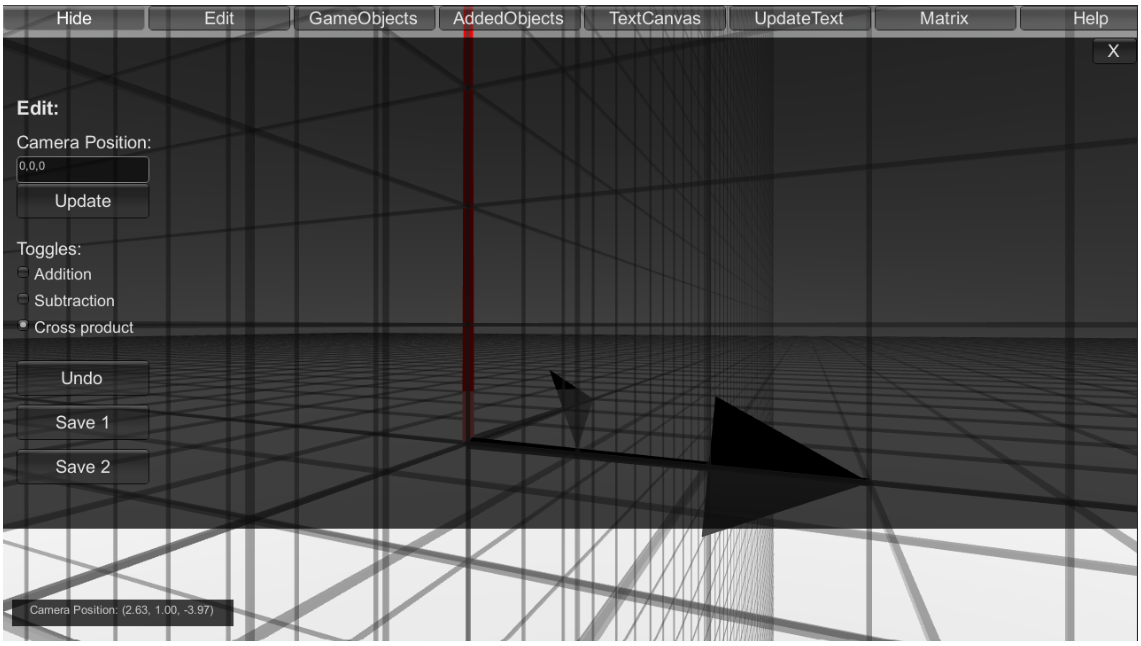
5 Results

The result of this project is a tool for creating VR worlds containing simple linear algebra lessons and as well a tool for connecting the student to the created VR worlds. The following subsections present how the lessons are made by math teachers with the tool, the student perspective in the lessons, and the test results of the unit tests, integration tests, paired t-test, and the usability testing presented.

5.1 Creating lessons with the tool

A lesson for cross-product is made by a teacher through the tool. The lesson's goal is for students to learn the properties of cross-product: the right-hand rule and the length of the resultant vector.

When the math teacher starts the tool, a GUI pops up, listing a row of buttons that can be pressed. The teacher presses the Help button and information on how to move the camera, zoom out, and move the objects themselves (see fig 8). The teacher presses the Edit button and then checks the checkbox for the cross-product toggle. Lastly, the teacher saves the VR world by pressing the save button (see fig 7).



Edit:

Camera Position: 0,0,0

Toggles:

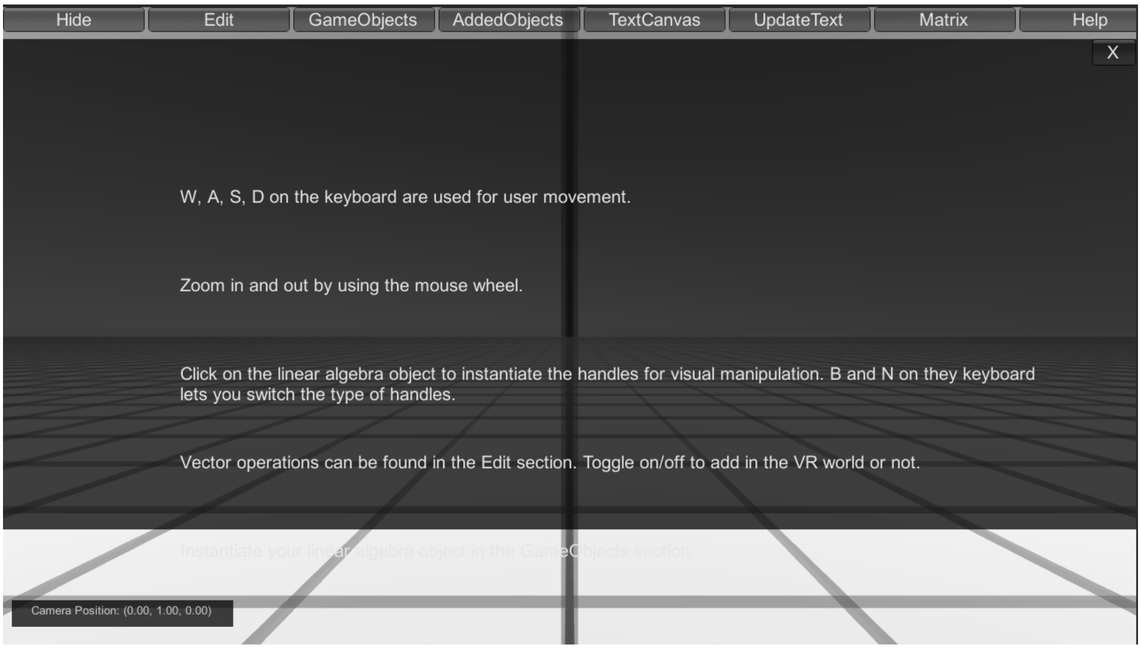
○ Addition

○ Subtraction

● Cross product

Camera Position: (2.63, 1.00, -3.97)

Figure 7: The panel shows the checkbox that when checked, can instantiate the required vector operation. The save buttons and the undo buttons are also in this panel.



W, A, S, D on the keyboard are used for user movement.

Zoom in and out by using the mouse wheel.

Click on the linear algebra object to instantiate the handles for visual manipulation. B and N on they keyboard lets you switch the type of handles.

Vector operations can be found in the Edit section. Toggle on/off to add in the VR world or not.

Instantiate your linear algebra object in the GameObjects section.

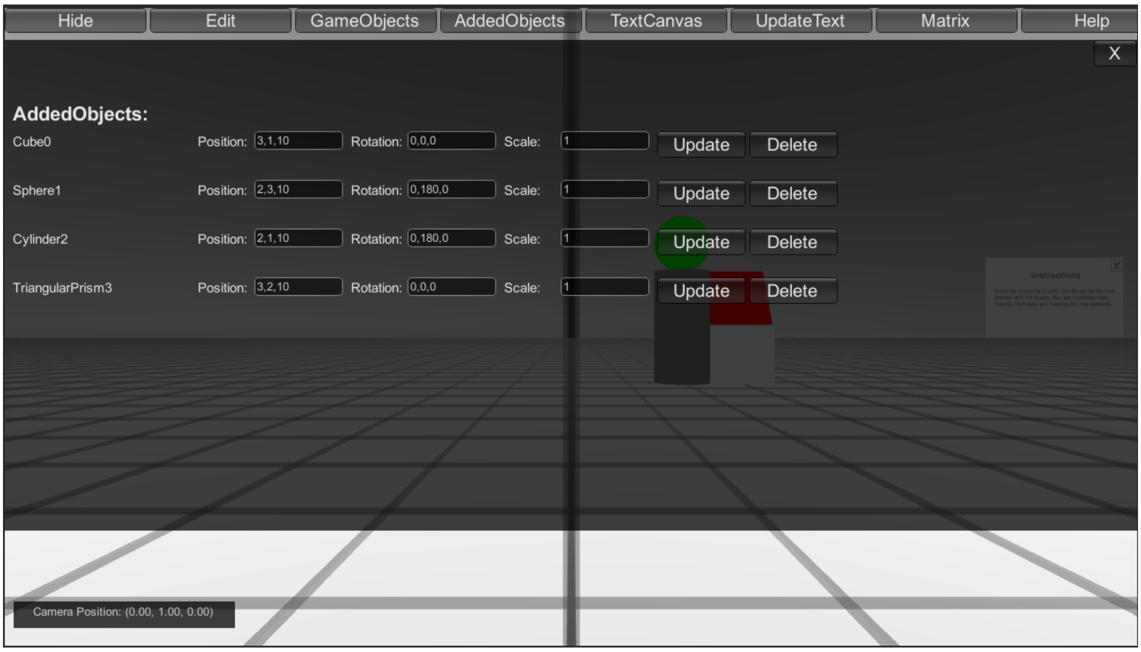
Camera Position: (0.00, 1.00, 0.00)

Figure 8: The panel gives some hints and tips.

Another lesson for using translation and scaling on real-life objects is made by a teacher through the tool. The lesson's goal is for students to reference the transformations done when the house is scaled by two, and the tree next to it must not overlap with the house.

The math teacher starts from the GUI and presses the GameObject button. The teacher then creates related objects for the lesson after setting the objects' position, Euler angles, dimensions, and color (see Appendix C). The teacher later presses the AddedObjects button to ensure the objects have the right values. If an object was created with wrong values, then the object can either be deleted or updated to the right values (see fig 9). Also, if the teacher wishes to, they can press the Matrix button to check the objects' mathematical expressions (see fig 11).

The teacher also wants to put instructions for the student and presses the TextCanvas button (see fig 10). Here, the instructions can be written and placed near the lesson. If any mistakes are made, the UpdateText button can be pressed to modify the current texts in the VR world (see Appendix C). Lastly, the teacher smoothly adjusts the position of an object with the cursor before saving the VR world (see fig 7).

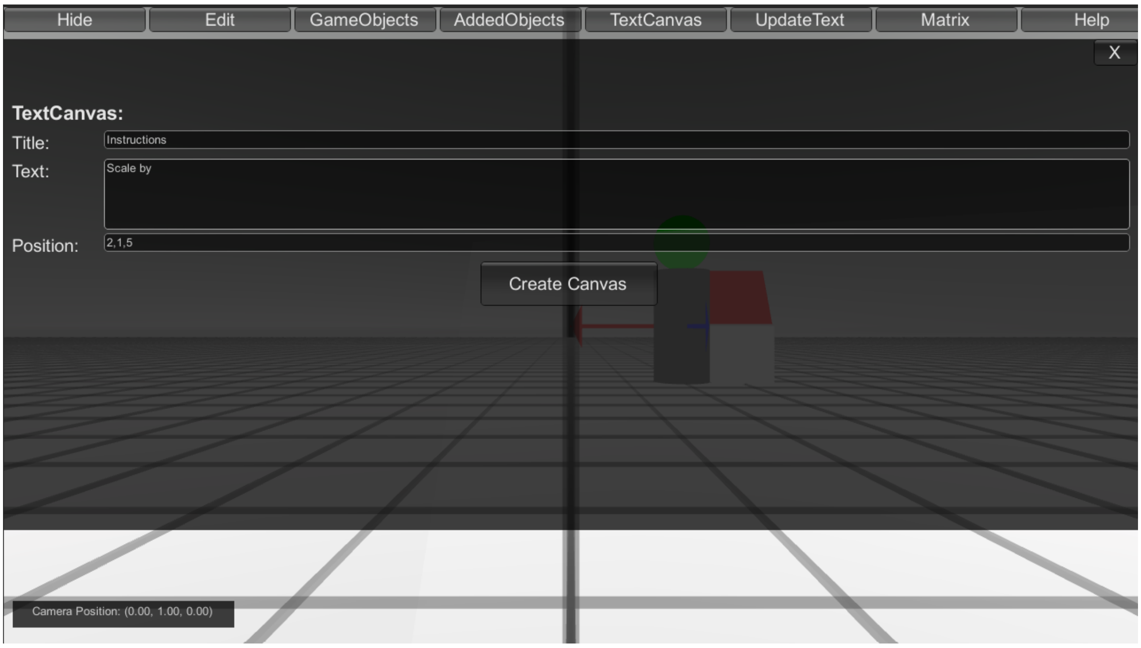


AddedObjects:

Cube0	Position:	3,1,10	Rotation:	0,0,0	Scale:	1
Sphere1	Position:	2,3,10	Rotation:	0,180,0	Scale:	1
Cylinder2	Position:	2,1,10	Rotation:	0,180,0	Scale:	1
TriangularPrism3	Position:	3,2,10	Rotation:	0,0,0	Scale:	1

Camera Position: (0.00, 1.00, 0.00)

Figure 9: The panel shows the created objects' name and their values. The objects can be either updated or deleted, if needed.



TextCanvas:

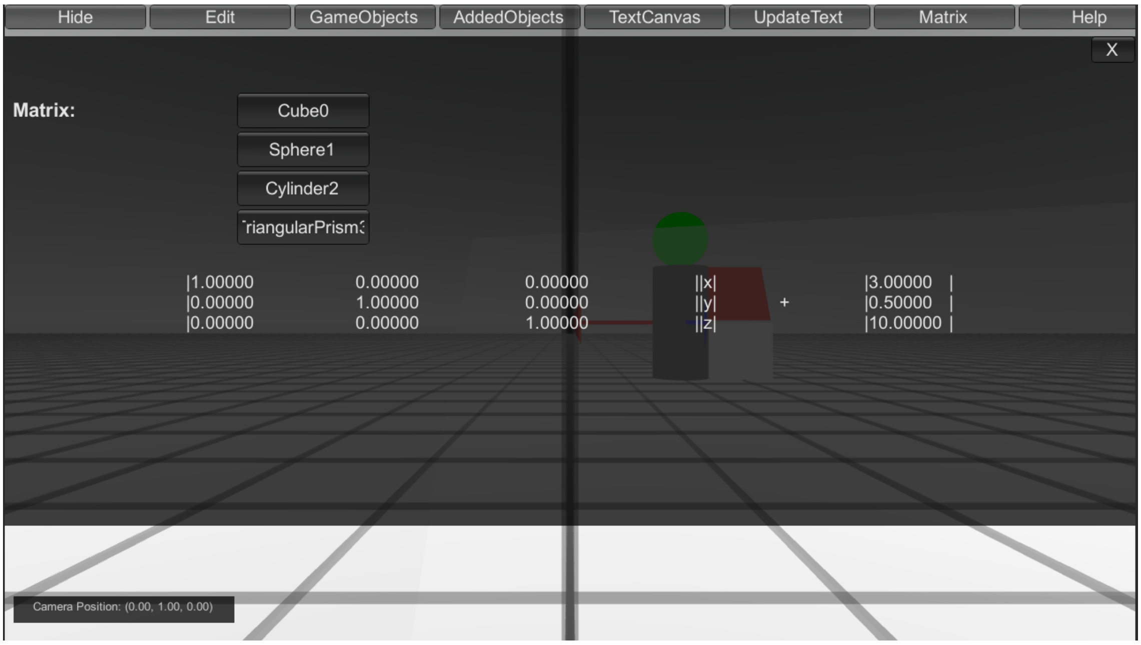
Title: Instructions

Text: Scale by

Position: 2,1,5

Camera Position: (0.00, 1.00, 0.00)

Figure 10: The panel lets you create new texts into the VR world.



```
|1.00000      0.00000      0.00000      ||x|          |3.00000  |
|0.00000      1.00000      0.00000      ||y|    +     |0.50000  |
|0.00000      0.00000      1.00000      ||z|          |10.00000 |
```

Camera Position: (0.00, 1.00, 0.00)

Figure 11: The panel shows the mathemtaical expressions of the objects in the VR world.

5.2 Entering the lesson in VR through the tool

Before connecting to the first lesson in the designed VR world, the teacher asks the student to observe the relative position of three vectors and find the similarities. The VR world contains a coordinate system where the students spawn at the origin. The student teleports from the origin by pointing the hand controller to the desired location and pressing the trigger button. At the origin, there are also two black vectors and one red vector to showcase the cross-product. The resultant red vector updates each time the black vectors are changed (see fig 12).

Furthermore, the student manipulates the black vectors with so-called handles. The type of handles can interchange between handles for translation, rotation, and scaling, respectively (see fig 12 and 14). Through playing with the two black vectors, the student should realize the following:

- The resultant red vector is always perpendicular to both black vectors,
- The resultant red vector is the longest if two black vectors are orthogonal (the angle between them is 90°), and it is zero if two black vectors are parallel (the angle between them is 0° or 180°),
- The length of the resultant red vector is proportional to the lengths of two black vectors.

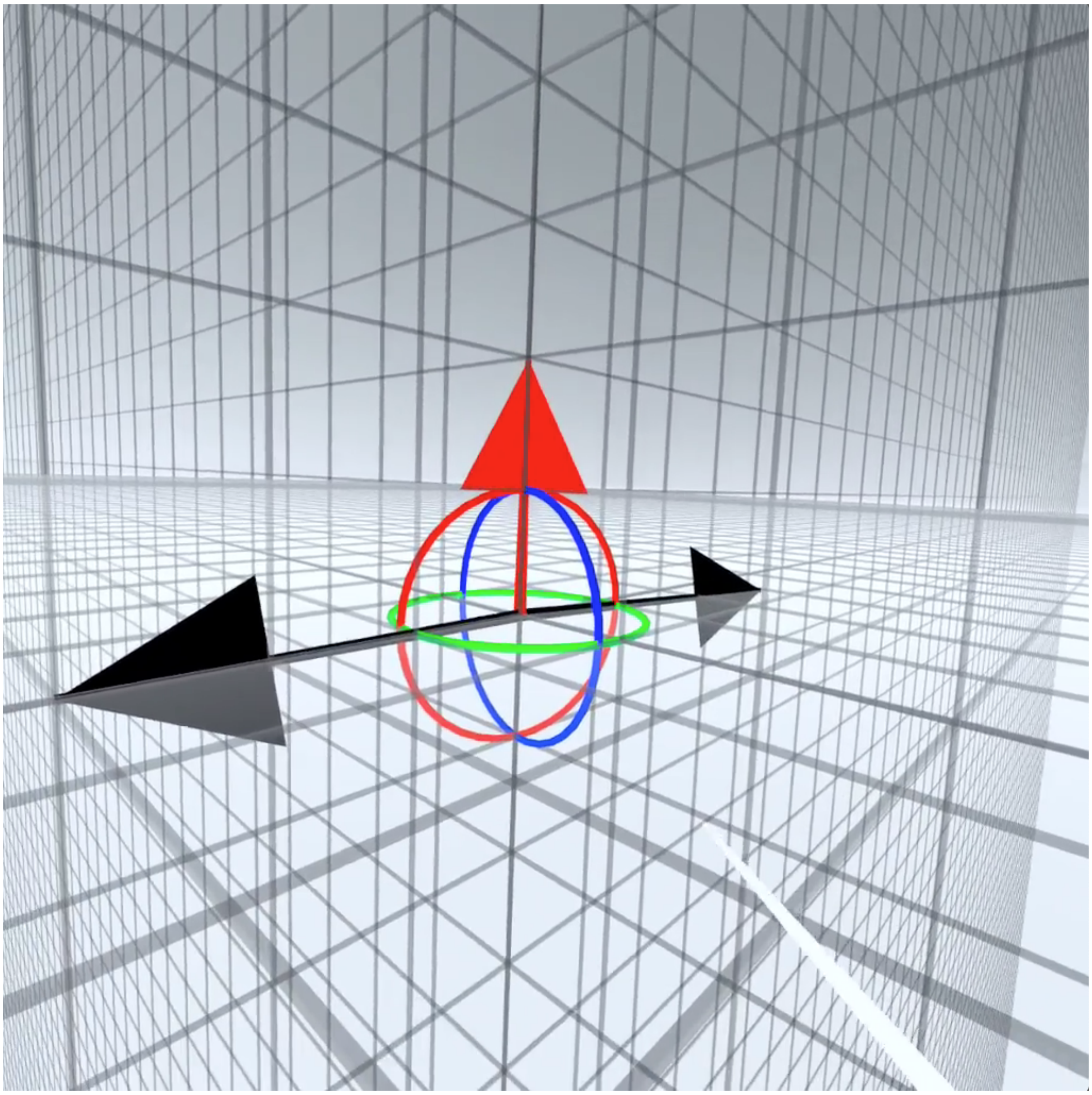
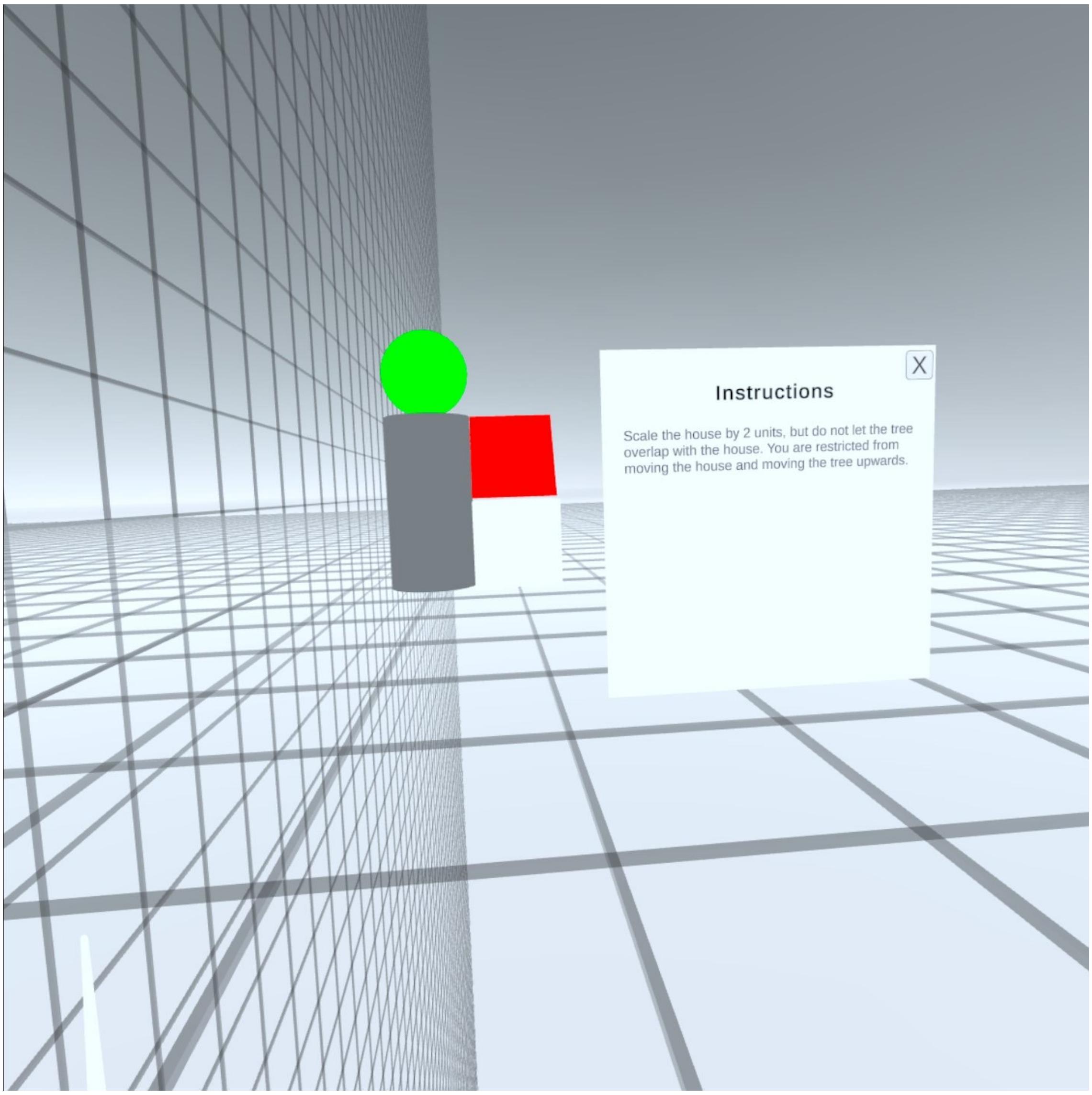


Figure 12: Shows two black vectors and one red resultant vector. A student is rotating the black vector with rotation handles, and the red vector updates on each change.

In the second lesson, the students enter another designed VR world again. A coordinate system is still there as in the previously designed world. However, there are different objects in the VR world. The student sees a prism on top of a cube forming the shape of a house, and next to it, a sphere on top of a cylinder forming a tree. Furthermore, the student also notices a white panel containing text near the objects. The text instructs the student to scale the ”house” by two, but the ”tree” should not overlap with the house. The text also gives additional instructions that the student is not allowed to move house and not move the tree upwards (see fig 13).

Proceeding with the second lesson, the student has another panel containing the mathematical expression nearby that reflects the values of the last object manipulated (see fig 14). The student uses the mathematical expressions of the objects to scale the house by two and reference the other required adjustments. After the experience, the teacher questions the student where the is origin related and how is it related to the objects. The students must understand that each object has a center point, which is more clearly seen when the translation is zero.



Instructions

Scale the house by 2 units, but do not let the tree overlap with the house. You are restricted from moving the house and moving the tree upwards.

Figure 13: Shows a panel in VR that contains some text. The text says to scale the house by two but to not overlap with tree and can only move the tree right, left, back and forward. The panel was created and saved by the teacher.

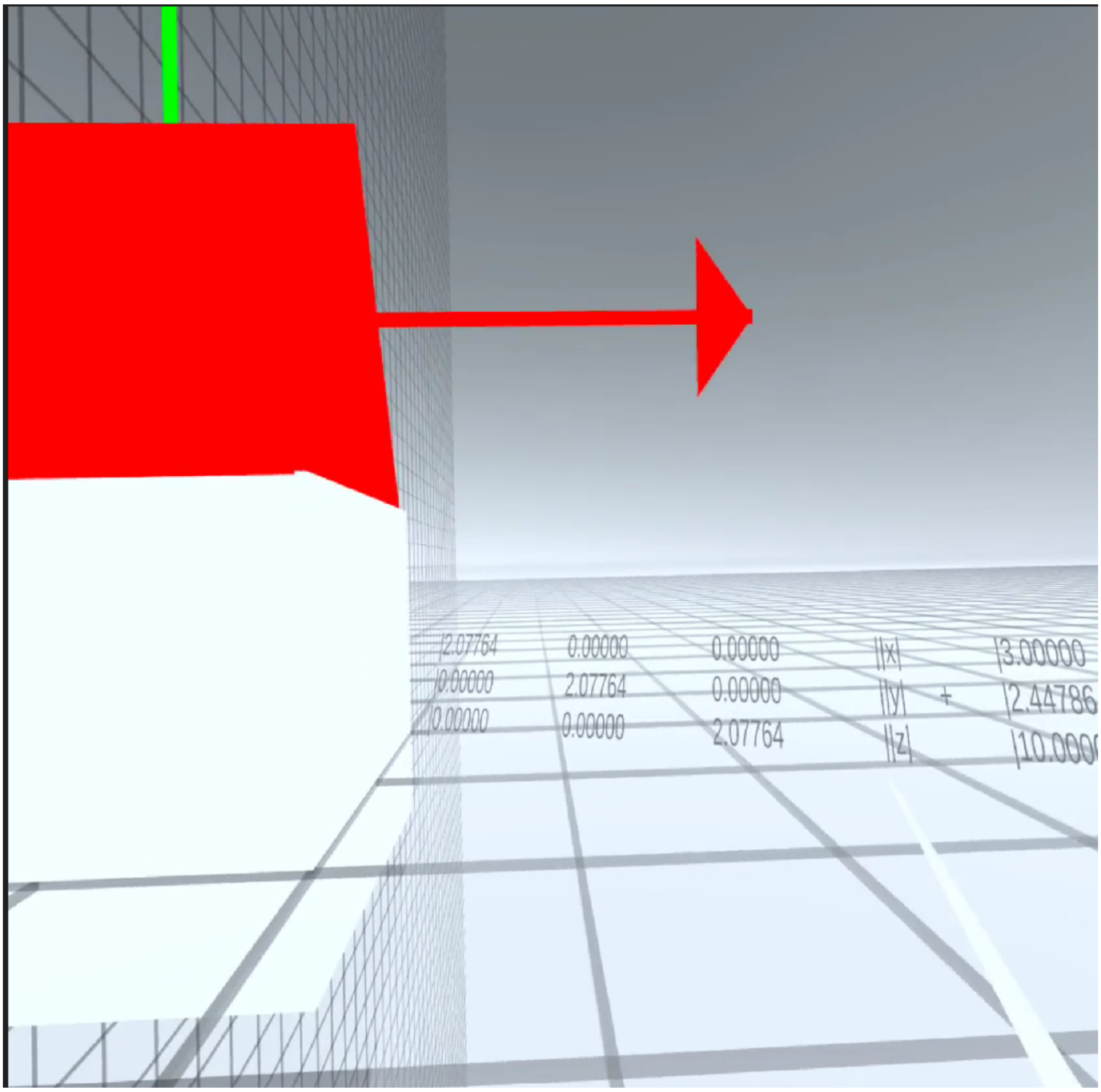

|2.07764 0.00000 0.00000 ||x| |3.00000
|0.00000 2.07764 0.00000 ||y| + |2.44786
|0.00000 0.00000 2.07764 ||z| |10.000

Figure 14: Shows a panel in VR that contains a mathematical expression manipulated with the handles. The recent mathematical expression is about the prism that was last changed. The panel follows the student and updates on each change with the handles. The handles type of handles on the prism are the translation handles.

5.3 The test results

The tests result are on the following unit tests, integration tests, paired t-tests, and usability tests.

5.3.1 Test results of the code

As the section 3.3 mentioned, the unit tests ensure code coverage based on MC/DC criterion. The automated unit tests had no problems on the last test run (see Appendix B).

As mentioned in the section 3.3, integration tests were done to ensure some requirement specifications. The first and the second integration tests were compared with the calculation done with the MATLAB program. The first integration test shows that the cross-product, vector addition, and vector subtraction calculated by the tool were the same as the MATLAB program result. Furthermore, the second integration test shows that the matrix multiplication, translation, scaling, and rotation were the same as the MATLAB program result (see Appendix A).

The third integration test showed no problems by comparing each object instantiated with the expected object data type. The object data types had the same interface but produced different geometric figures. The game objects of these 2D and 3D figures had a Mesh component used to compare the object data type. However, the game objects with 1D figures had the LineRenderer component for comparison. The same integration test also had a counter to count the game objects in the scene before and after each instantiation (see Appendix A).

5.3.2 Usability test on creating lesson with the tool

The usability test for the framework had four participants, two that taught math at Halmstad University and two others that only completed a course in linear algebra. For this test, the teachers are called group A, and the rest are called group B. Only one of the teachers filled in a questionnaire based on the usability test.

The project team asked group B to instantiate the objects and later questioned if they had problems finding them. One had trouble connecting the position at the lower left corner to be the same as the camera. The other one had trouble initially when instantiating an object in the same position as the camera. However, some movements allowed the participant to find the instantiated object, and associations were drawn, as shown in the participant's later showcasing. After the demo, group B was asked if there were any problems with the camera. Both of the participants said the rotation was strange. One of the participants added a comment that the tool needed an undo button because the GUI was too small.

The project team asked group A to create lessons with the tool. Each participant created their lesson, which is explained in section 5.1. Each lesson has its own goal that the student needs to achieve.

One of the teachers filled out a questionnaire based on the usability test. The first question on using all the features of the tool was negative. However, the answer to the second question was if a missing feature was negative. The answer to the third and fourth questions about making mistakes with the tool and the answer was that no mistakes were made. The fifth question was if the tool was more efficient than the worst visualization tool used, and the sixth question was if the tool was more efficient than the best visualization tool used. The answer to both the fifth and sixth were positive. Lastly, the seventh question was about how easy tool usage was, and the answer was positive.

Furthermore, the questionnaire also has additional comments made by the teacher. On the first question, the teacher commented that the tool covers several fields and that there was no need to use all the features. On the last question, the teacher commented that the menu was clear and needed help only initially.

5.3.3 Paired t-test on students entering VR with the tool

The paired t-test, as mentioned in the section 3.3, is on the augmentation of simple linear algebra lessons with VR, involving 30 participants. The paired t-test used data from the participants that were extracted by determining if the participants understood each lesson's contents; the data was extracted before and after VR with the same questions about the contents of each lesson (see Appendix D).

The sample data show a positive difference between using VR and not using VR, where the sample mean is 0.5, and the standard error is 0.1712. Further calculation for a paired t-test shows that it is statistically significant that VR augments learning simple linear algebra lessons ($p < 0.5$). The p-value was obtained with the help of the calculated t-score with 29 degrees of freedom ($t(29) = 0.003349$) and can be illustrated with a t-distribution with 29 degrees of freedom (see Appendix E).

6 Discussion

This section will discuss the limitations and the social aspects, the test results, and compare the tool with other existing products.

6.1 Limitations

Limitations of not reaching the following requirement specifications are as follows:

1. Able to choose different types of manipulation for linear algebra objects such as a collection of sliders, a collection of buttons, or just manually controlling.
2. Toggle on or off to show the coordinates of linear algebra objects.
3. Creating shapes by putting dots in space.
4. Creating animated linear transformation with the press of a single button
5. Premade 3D coordinate systems to instantiate.
6. Premade different biomasses to instantiate

The reason for not following through foremost is that instantiating 3D coordinate systems into the VR world was later discarded for not being clear enough during continuous discussions about what the coordinate system should do and act during several dialogs with our project providers. The other points were either too trivial to prioritize or too ambitious for the project team to complete, as this is just a concept.

Another limitation was that the usability test had too few participants with background of teaching linear algebra. The limitation of the number of teachers makes it difficult to use statistical tools to give more credibility in the usability test.

6.2 Tests

On the programming part, the unit tests helped notice part of the code or methods that broke but did not give noticeable errors when running the program. The unit tests also gave a certain amount of trust to concentrate on the programming part instead of manually testing the what-ifs.

The integration tests found a problem in vector manipulation. As a consequence of finding the problem, the related unit tests were adjusted to notice the problem, and it was fixed.

The usability test gave much feedback in the form of comments on the GUI and the manipulation of the user's camera. Most of the desired features or problems were fixed, including the rotation of the user's camera. The GUI could still be more user-friendly, as indicated by a teacher's comment that getting started with the GUI was hard enough to require help initially. However, the same teacher still answered positively on the difficulty of using the tool as a whole. The tool's usage indicates it is user-friendly for users without a programming background because the GUI has no code syntax.

Moreover, the positive answer on the tool's efficiency compared to the best visualization tool indicates a general level of better performance in doing its purpose that satisfies the teacher. Consequently, when used to create lessons, the teacher is more engrossed with the tool than other visualization tools.

The paired t-test gave a positive result for learning simple linear algebra lessons with the tool, and the result was statically significant. The significant threshold was 0.05, indicating a 5% risk of a difference exists when there is no actual difference. The significant threshold of 0.05 was bigger than the p-value calculated, which makes the test statically significant. Furthermore, the positive learning results from using VR indicate that the participants were more engaged and focused when using VR. The engaged and focus part when using VR makes sense because the participant learned by touch and visualization that was closer to reality.

6.3 Social aspects

Introduced briefly social queries of the tool.

6.3.1 Finance

Investing in this tool would require buying software developers to complete this. Furthermore, if the tool managed to be polished, the investment in VR equipment would be immense as it costs more than computers but brings less use of it based on the limitation of the current model. VR equipment needs to be able to replace computers to counteract the costs. The option might not be viable for a niche educational instrument from an economic perspective.

6.3.2 Security

It is possible to have your data taken from the company that owns the Oculus Quest 2. However, like all other computers, using Oculus Quest 2 is not different, and the GDPR has some deterring effects of using the user's data for bad purposes, for example selling it to phone scammers. The data saved in the tool is not encrypted, and solving that issue could protect it from security breaches.

6.4 Comparison with existing products

Calcflow has more mathematical presence, but the direction is not the same. Calcflow is aimed at plotting in VR, which is different from making a VR world to explore math concepts and linear algebra concepts in the case of this project. However, CalcVR has the same concept, only more furnished. However, creating your world with it might not be what it aims for, which would differ in that teachers can save their changes and then let others experience them.

7 Conclusion

Developing the framework with a user-friendly interface in Unity to assist math teachers in creating 3D worlds and integrating them into VR environments demonstrates that such a tool is feasible and can potentially improve students' learning experience.

This framework can assist in empowering the creation of 3D worlds and VR experiences, making them more accessible to a wider variety of educators by removing the requirement that teachers possess coding abilities. The time and resources required to develop and distribute VR experiences can be decreased by having the capability to generate and integrate 3D environments directly into a VR headset.

However, further research is needed to analyze this framework's success in enhancing student learning outcomes and identify areas for future advancement and improvement.

7.1 Future Work

The current version of the tool offers math teachers an interface with a GUI that allows them to prepare their lessons in a 3D environment without having to write any code. However, some issues and limitations could be addressed in future works.

7.1.1 Performance Optimization

The tool's performance is one of its main drawbacks. While the focus was on usability and functionality, performance was not a priority. Future works should therefore focus on improving the tool's performance, particularly when handling large 3D scenes and complex interactions. One potential solution is using more effective algorithms and reducing the number of draw calls.

7.1.2 Enhancing the User Interface

The tool's current user interface relies on Unity's IMGUI library, which can occasionally be slow and unstable. Future works should explore more reliable and effective alternatives for replacing IMGUI with a more efficient and stable UI system.

7.1.3 Enhancing the Mathematical Part of the Interface

There is still room for development in terms of the mathematical side of the interface. The current version of the tool allows teachers to construct austere 3D worlds using linear algebra concepts such as scaling, rotation, and position transformations. Future development can involve incorporating more mathematical implementations for different math topics into the tool. For instance, solving math that cannot be illustrated directly by improving the readability of text-based math in VR.

7.1.4 Adding Functionality

Future development should also focus on expanding the tool's functionality. For instance, the tool could allow teachers to add constraints to objects, limiting how or under what parameters students can interact with them. Additionally, more tools can be added to assist teachers in developing 3D worlds that are more engaging and dynamic.

8 Reference

References

[1] Hsu YC. Exploring the learning motivation and effectiveness of applying virtual reality to high school mathematics. Universal Journal of Educational Research. 2020;8(2):438-44.

[2] Prabowo A, Anggoro R, Astuti D, Fahmi S. Interactive multimedia-based teaching material for 3-dimensional geometry. In: Journal of Physics: Conference Series. vol. 943. IOP Publishing; 2017. p. 012047.

[3] Boyles B. Virtual reality and augmented reality in education. Center For Teaching Excellence, United States Military Academy, West Point, Ny. 2017.

[4] 3D Grapher by Mathpix;. [Acccessed: 2022-10-17]. Available from: http://grapher.mathpix.com/.

[5] CalcPlot3D - LibreTexts;. [Acccessed: 2022-10-17]. Available from: https://c3d.libretexts.org/CalcPlot3D/index.html.

[6] Calcflow on Steam;. [Acccessed: 2022-10-17]. Available from: https://store.steampowered.com/app/547280/Calcflow/.

[7] Takac M. Application of Web-based Immersive Virtual Reality in Mathematics Education. In: 2020 21th International Carpathian Control Conference (ICCC). IEEE; 2020. p. 1-6.

[8] VIVE - VR Headsets, Games, and Metaverse Life | United States;. [Acccessed: 2022-10-17]. Available from: https://www.vive.com/.

[9] CalcVR (Calculus in Virtual Reality) – Information and Resources for the CalcVR (Calculus in Virtual Reality) App;. [Acccessed: 2022-10-17]. Available from: https://calcvr.org/.

[10] Cardboard - Google VR;. [Acccessed: 2022-10-17]. Available from: https://arvr.google.com/cardboard/.

[11] PlayStation VR | Live the game med PS VR-hodesettet;. [Acccessed: 2022-10-17]. Available from: https://www.playstation.com/no-no/ps-vr/.

[12] VIVE Pro | VIVE United States;. [Acccessed: 2022-10-17]. Available from: https://www.vive.com/us/product/vive-pro/.

[13] Oculus Quest Store: VR Games, Apps, More;. [Acccessed: 2022-10-17]. Available from: https://www.oculus.com/experiences/quest/.

[14] Patel K, Bailenson JN, Hack-Jung S, Diankov R, Bajcsy R. The effects of fully immersive virtual reality on the learning of physical tasks. In: Proceedings of the 9th Annual International Workshop on Presence, Ohio, USA; 2006. p. 87-94.

[15] Bevilacqua R, Maranesi E, Riccardi GR, Di Donna V, Pelliccioni P, Luzi R, et al. Non-immersive virtual reality for rehabilitation of the older people: a systematic review into efficacy and effectiveness. Journal of clinical medicine. 2019;8(11):1882.

[16] Gao S, Wan H, Peng Q. An approach to solid modeling in a semi-immersive virtual environment. Computers & Graphics. 2000;24(2):191-202.

[17] Introducing Oculus Quest 2, the Next Generation of All-in-One-vr;. [Acccessed: 2022-10-17]. Available from: https://developer.oculus.com/blog/introducing-oculus-quest-2-the-next-generation-of-all-in-one-vr/.

[18] Fundamentals of WebXR - Web APIs | MDN;. [Acccessed: 2022-10-17]. Available from: https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API/Fundamentals.

[19] Segal M, Akeley K. The OpenGL® Graphics System: A Specification (Version 4.6 (Core Profile)-May 5, 2022). 2022 May 5.

[20] Butcher P. A Framework for Web-Based Immersive Analytics [Bachelor's Thesis]. The University of Chester; 2020.

[21] Gregory J. Game engine architecture. AK Peters/CRC Press; 2018.

[22] Knutsen KÍ. Visual Scripting in Game Development [Bachelor's Thesis]. Metropolia University of Applied Sciences; 2021.

[23] Magnusson LV. Game mechanics engine [Master Thesis]. Østfold University College; 2011.

[24] Anderson EF, McLoughlin L, Liarokapis F, Peters C, Petridis P, De Freitas S. Developing serious games for cultural heritage: a state-of-the-art review. Virtual reality. 2010;14(4):255-75.

[25] Sharif KH, Ameen SY. Game Engines Evaluation for Serious Game Development in Education. In: 2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM). IEEE; 2021. p. 1-6.

[26] Lambru C, Morar A, Moldoveanu F, Asavei V, Moldoveanu A. Comparative Analysis of Real-Time Global Illumination Techniques in Current Game Engines. IEEE Access. 2021;9:125158-83.

[27] XR Plug-in Framework - Unity - Manual;. [Acccessed: 2022-10-17]. Available from: https://docs.unity3d.com/Manual/XRPluginArchitecture.html.

[28] Virtual Reality (VR) - Confluence Mobile - Documentation;. [Acccessed: 2022-10-17]. Available from: https://docs.cryengine.com/pages/viewpage.action?pageId=25536773.

[29] Šmíd A. Comparison of unity and unreal engine [Bachelor's Thesis]. Czech Technical University in Prague; 2017.

[30] Krstić V, Mekterović I. Unity as a Physics Simulator: Calculating Mean Free Path for Hard Disk Gas. In: 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO). IEEE; 2021. p. 613-6.

[31] OpenXR Plugin | 1.5.3;. [Acccessed: 2022-10-17]. Available from: https://docs.unity3d.com/Packages/com.unity.xr.openxr@1.5/manual/index.html.

[32] Linjär algebra 7,5 hp;. [Acccessed: 2023-02-28]. Available from: https://www.hh.se/sitevision/proxy/student/innehall-a-o/kursplan.html/svid12_464ca102168ed1f8d3b1293f/752680950/se_proxy/utb_kursplan.asp?kurskod=MA2027&revisionsnr=15&format=pdf.

[33] Skolverket Ämne Fysik 1a;. [Acccessed: 2023-02-28]. Available from: https://www.skolverket.se/undervisning/gymnasieskolan/laroplan-program-och-amnen-i-gymnasieskolan/gymnasieprogrammen/amne?url=-996270488%2Fsyllabuscw%2Fjsp%2Fsubject.htm%3FsubjectCode%3DFYS%26courseCode%3DFYSFYS01a%26version%3D3%26tos%3Dgy&sv.url=12.5dfee44715d35a5cdfa92a3#anchor_FYSFYS01a.

[34] Räkna med vektorer (Matte 1, Geometri) – Matteboken;. [Acccessed: 2023-02-28]. Available from: https://www.matteboken.se/lektioner/matte-1/geometri/rakna-med-vektorer#!/.

[35] Sparr G. Linjär algebra. Studentlitteratur; 1994.

[36] Unity Education Grant License;. [Acccessed: 2022-10-17]. Available from: https://unity.com/products/unity-education-grant-license.

[37] Educators - Unreal Engine;. [Acccessed: 2022-10-17]. Available from: https://www.unrealengine.com/en-US/educators.

[38] Education - Godot Engine;. [Acccessed: 2022-10-17]. Available from: https://godotengine.org/education.

[39] Lesson: Object-Oriented Programming Concepts;. [Acccessed: 2022-10-17]. Available from: https://docs.oracle.com/javase/tutorial/java/concepts/.

[40] Object-Oriented Programming (C) - Microsoft Learn;. [Acccessed: 2022-10-17]. Available from: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop.

[41] Ha TH. Game Development with Unreal Engine [Bachelor's Thesis]. South-Eastern Finland: University of Applied Science; 2022.

[42] Scripting languages - Godot Docs;. [Acccessed: 2022-10-17]. Available from: https://docs.godotengine.org/en/stable/getting_started/step_by_step/scripting_languages.html.

[43] Flomén R, Gustafsson M. Game developer experience: A cognitive task analysis with different game engines [Blekinge Institute of Technology]. Karlskrona Sweden: University of Applied Science; 2020.

[44] Peters A. High-level programming languages for low-level programming [Doctoral dissertation]. The University of Texas at Austin; 2020.

[45] Pirker J, Bertini M, Lux M. Open source for video games: a shortlist of game engines. ACM SIGMultimedia Records. 2022;12(3):1-1.

[46] Barnum CM. Usability testing essentials: ready, set... test! Morgan Kaufmann; 2020.

[47] Reference: Conditions for inference on a mean (article) | Khan Academy;. [Acccessed: 2023-02-28]. Available from: https://www.khanacademy.org/math/ap-statistics/xfb5d8e68:inference-quantitative-means/one-sample-t-interval-mean/a/reference-conditions-inference-one-mean.

[48] t-Tables;. [Acccessed: 2023-02-28]. Available from: https://faculty.washington.edu/heagerty/Books/Biostatistics/TABLES/t-Tables/.

[49] Manual: Introduction to components;. [Acccessed: 2023-02-28]. Available from: https://docs.unity3d.com/Manual/Components.html.

[50] Manual: Transforms;. [Acccessed: 2023-02-28]. Available from: https://docs.unity3d.com/Manual/class-Transform.html.

[51] Manual: Mesh Renderer component;. [Acccessed: 2023-02-28]. Available from: https://docs.unity3d.com/Manual/class-MeshRenderer.html.

[52] Manual: Mesh Filter component;. [Acccessed: 2023-02-28]. Available from: https://docs.unity3d.com/Manual/class-MeshFilter.html.

[53] Manual: Line Renderer component;. [Acccessed: 2023-02-28]. Available from: https://docs.unity3d.com/Manual/class-LineRenderer.html.

[54] Introduction to collision;. [Acccessed: 2023-02-28]. Available from: https://docs.unity3d.com/Manual/CollidersOverview.html.

[55] Manual: Constraints;. [Acccessed: 2023-02-28]. Available from: https://docs.unity3d.com/Manual/Constraints.html.

[56] Manual: Important Classes - MonoBehaviour;. [Acccessed: 2023-02-28]. Available from: https://docs.unity3d.com/Manual/class-MonoBehaviour.html.

[57] Manual: Order of execution for event functions;. [Acccessed: 2023-02-28]. Available from: https://docs.unity3d.com/Manual/ExecutionOrder.html.

HALMSTAD
UNIVERSITY

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se

I Appendix A - Integration Tests

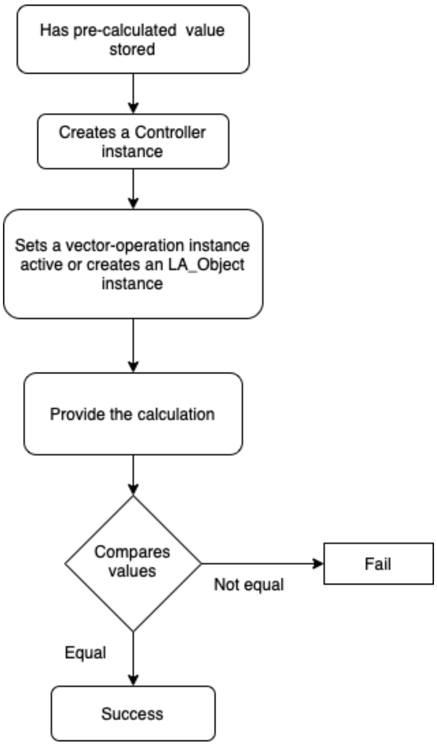

Has pre-calculated value stored
Creates a Controller instance
Sets a vector-operation instance active or creates an LA_Object instance
Provide the calculation
Compares values
Not equal
Fail
Equal
Success

Figure 15: The integration test compares if certain calculations made by the code are correct. the The image was made by the project members.

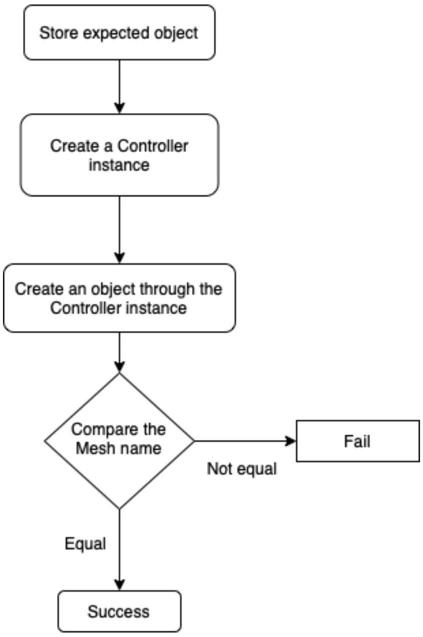
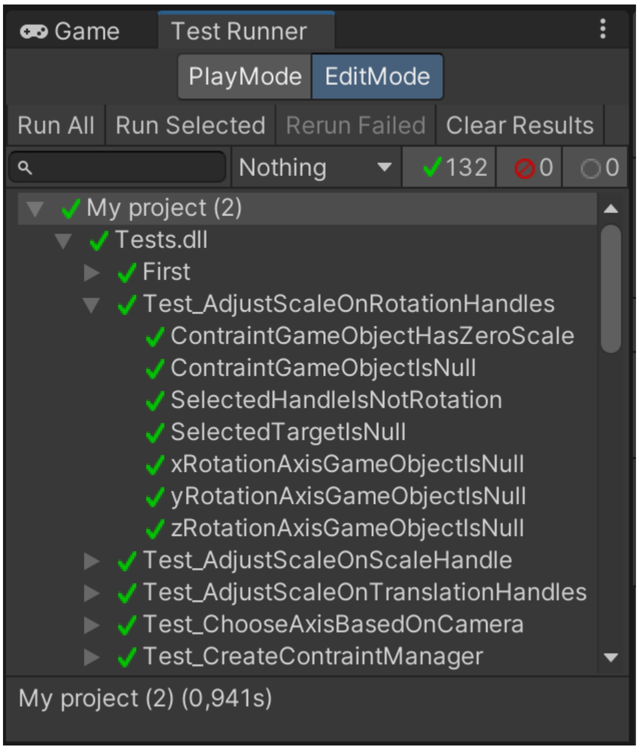

Store expected object
Create a Controller instance
Create an object through the Controller instance
Compare the Mesh name
Not equal
Fail
Equal
Success

Figure 16: The integration test compares if instantiations of 2D and 3D geometric figures made by the code are the correct types. The image was made by the project members.

II Appendix B - Unit Tests



```
▼ ✓ My project (2)
  ▼ ✓ Tests.dll
    ▶ ✓ First
    ▼ ✓ Test_AdjustScaleOnRotationHandles
        ✓ ContraintGameObjectHasZeroScale
        ✓ ContraintGameObjectIsNull
        ✓ SelectedHandleIsNotRotation
        ✓ SelectedTargetIsNull
        ✓ xRotationAxisGameObjectIsNull
        ✓ yRotationAxisGameObjectIsNull
        ✓ zRotationAxisGameObjectIsNull
    ▶ ✓ Test_AdjustScaleOnScaleHandle
    ▶ ✓ Test_AdjustScaleOnTranslationHandles
    ▶ ✓ Test_ChooseAxisBasedOnCamera
    ▶ ✓ Test_CreateContraintManager
My project (2) (0,941s)
```

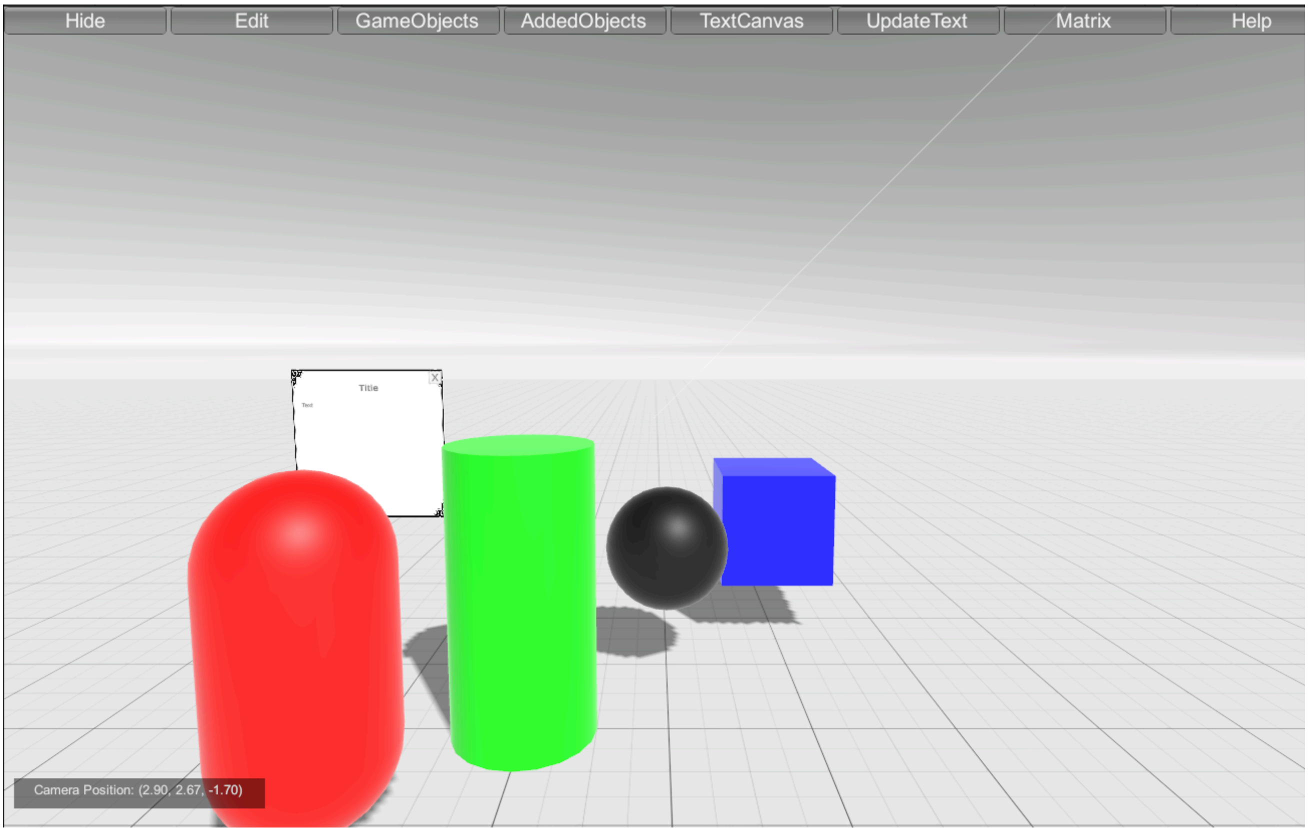
Figure 17: A snippet of the unit testings in Unity. The unit tests are automated and only require to press a button to run the all the unit tests. The image was made by the project members.

III Appendix C - User Guide

VR4MATH

User Guide

Ahmed Oogle & Abraham Al-bashki



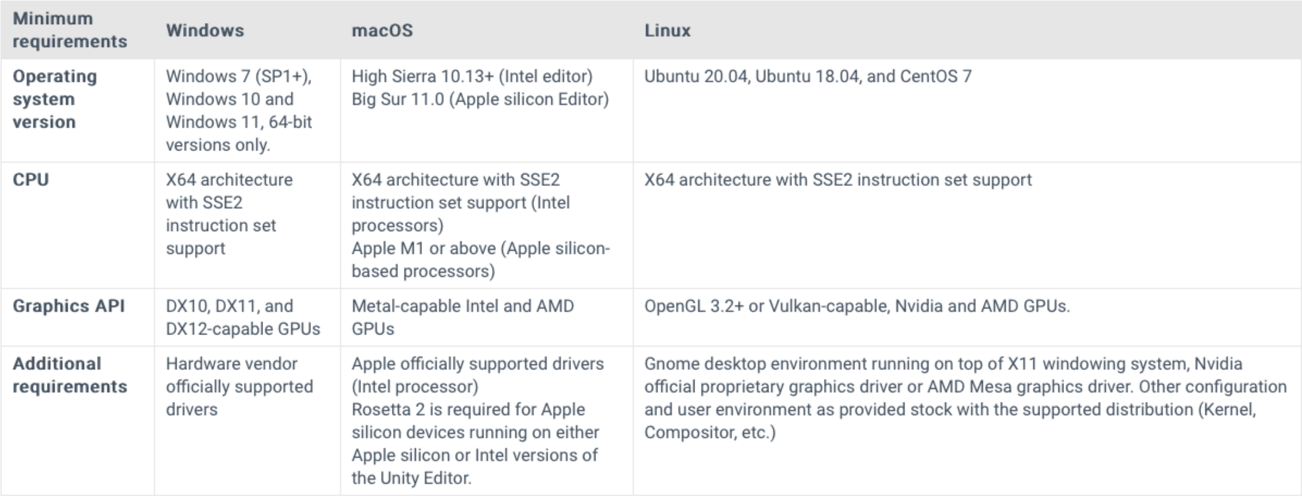
This manual is a step-by-step guide on how to use the project VR4MATH in the Unity application to create 3D worlds for visualizing math. The manual focuses on giving clear instructions for each phase of the process, along with images and graphics to make the steps easier to understand.

The manual also provides recommendations and best practices for creating practical 3D worlds. By following the manual, teachers can create engaging and interactive 3D worlds for their math classes that help students perceive math more memorably and engagingly.

Table of Contents

System Requirements

Before we begin, please ensure that your computer meets the following requirements:



Minimum requirements	Windows	macOS	Linux
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only.	High Sierra 10.13+ (Intel editor) Big Sur 11.0 (Apple silicon Editor)	Ubuntu 20.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support (Intel processors) Apple M1 or above (Apple silicon-based processors)	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers (Intel processor) Rosetta 2 is required for Apple silicon devices running on either Apple silicon or Intel versions of the Unity Editor.	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)

Setting Up the Project

How to install Unity to run the project successfully!

Unity Hub is a tool that helps you manage multiple versions of Unity on your computer and allows you to easily add and remove additional components like Android Build Support needed to run the project. It is essential since you need the correct version of Unity installed together with the required components to create applications for VR Meta Quest 2. Unity Hub makes sure you can manage these dependencies and have everything you need to get started.

Before you begin, please note that the project you're working on was created using Unity version 2021.3.10f1. It is advised to use this version for the best compatibility, even though newer versions might also work.

1. Download Unity Hub from https://unity.com/download, the official Unity website.

2. Launch the installer file you downloaded to install Unity Hub on your PC.

3. Launch Unity Hub and click on the "Installs" tab.

4. Select the Unity version you want to install from the list of available versions. The project was tested with Unity version 2021.3.10f1.

5. Select the "Android Build Support" component with OpenJDK and Android SDK &NDK Tools to add Android build support.

6. Choose a location on your computer where you want to install this version of Unity.

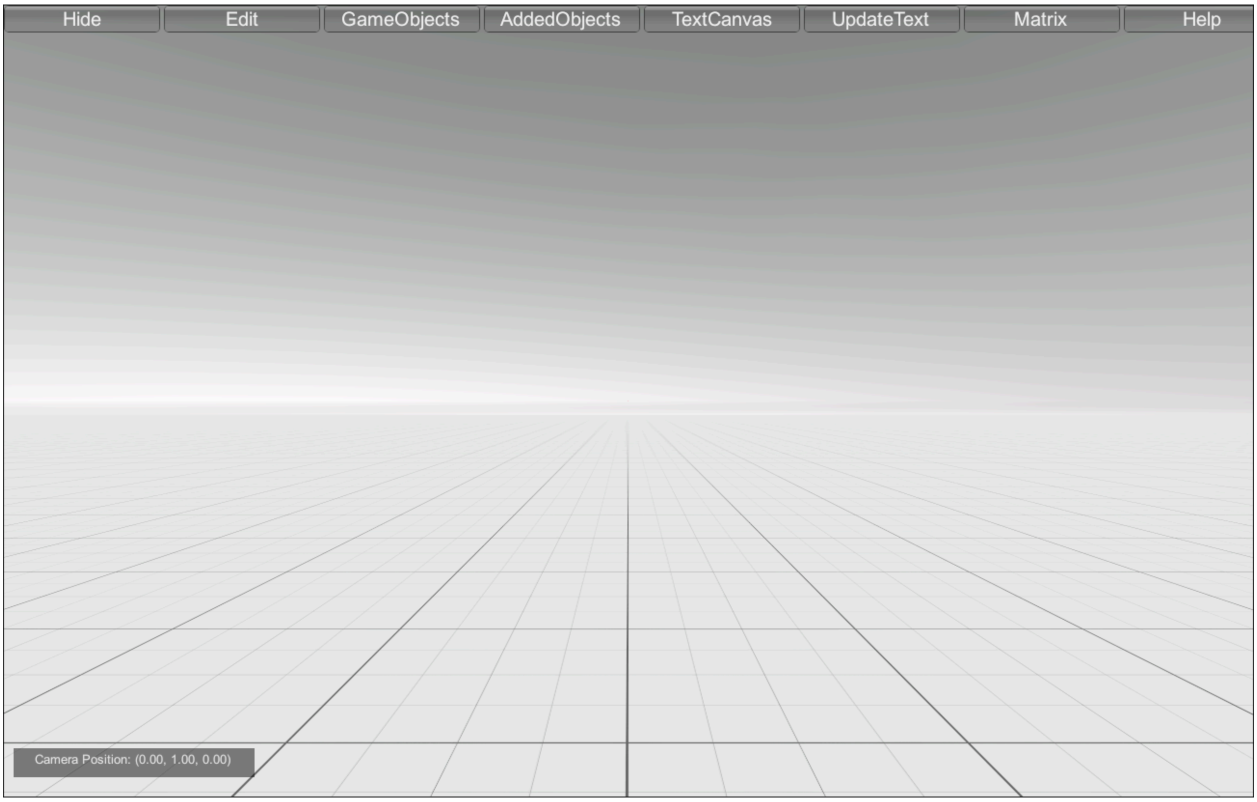
7. Click "Install," and Unity Hub will download and install the selected version of Unity and your selected components.

8. Once the installation is complete, click on the "Projects" tab in Unity Hub and click "Open" to choose the project file.

9. Once the project is opened, click on the play button at the top center, and voilà, you can start developing your application for Android using Unity.

Usage Instructions

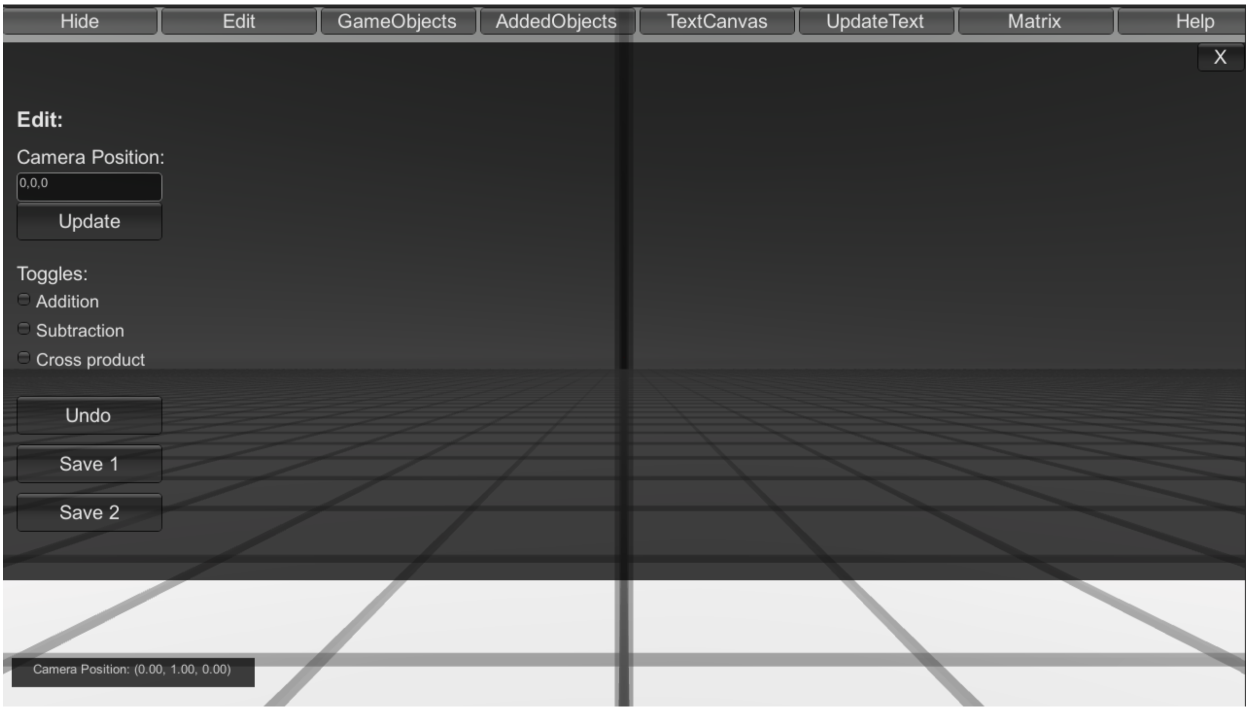
THE GUI



When you first launch the project, you will see the main menu bar at the top of the screen with eight options: Hide, Edit, GameObjects, AddedObjects, TextCanvas, UpdateText, Matrix and Help. Click on the respective menu item to display a window. The Hide menu item only hides the menu bar and all open windows.

Edit:

The Edit option allows you to adjust the camera position, save your work, and undo changes. Clicking on the "Edit" menu item will bring up a window with several options:



Camera Position: 0,0,0

☐ Addition

☐ Subtraction

☐ Cross product

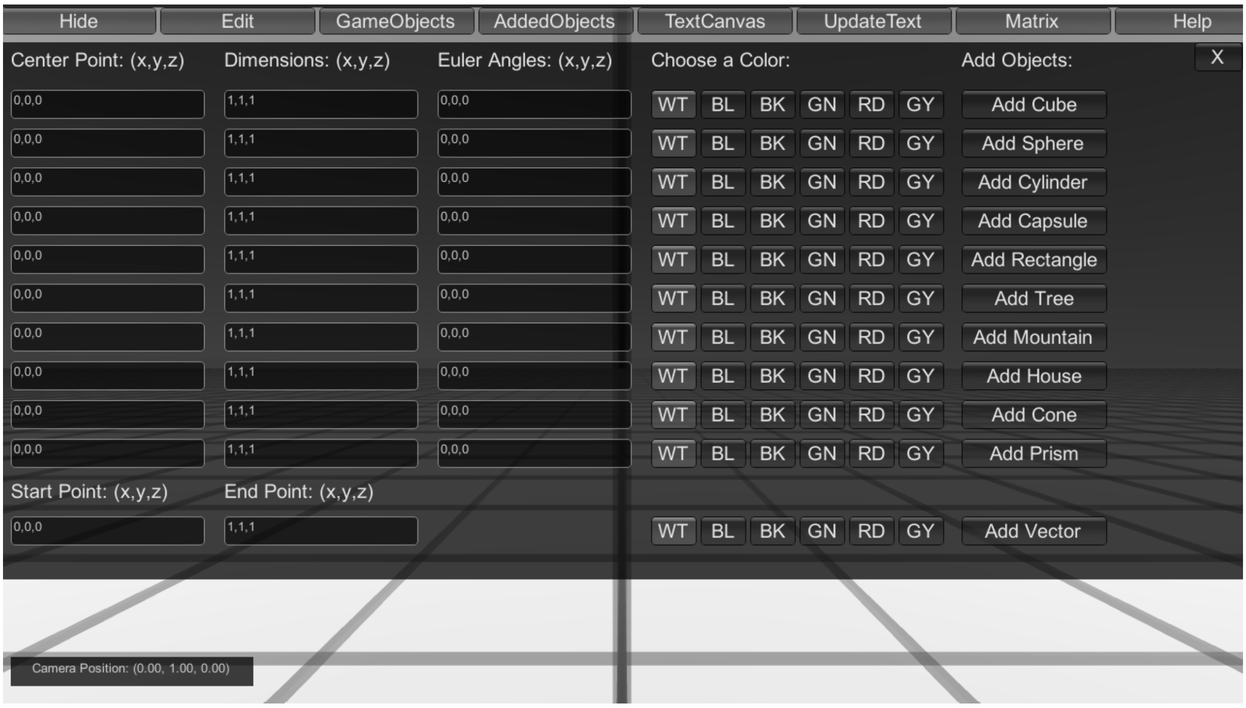
Camera Position: (0.00, 1.00, 0.00)

You can input a new value on the "Camera Position" text field to adjust the camera position, then click "Update" to apply the changes.

To reverse changes, click "Edit" and select "Undo." That will undo the last change you made to the 3D world.

Click on "Save" after clicking the Edit menu item to save your changes. That will save the current state of your 3D world to a file on your computer.

GameObjects: Add Objects



Center Point: (x,y,z)	Dimensions: (x,y,z)	Euler Angles: (x,y,z)	Choose a Color:	Add Objects:
0,0,0	1,1,1	0,0,0	WT BL BK GN RD GY	Add Cube
0,0,0	1,1,1	0,0,0	WT BL BK GN RD GY	Add Sphere
0,0,0	1,1,1	0,0,0	WT BL BK GN RD GY	Add Cylinder
0,0,0	1,1,1	0,0,0	WT BL BK GN RD GY	Add Capsule
0,0,0	1,1,1	0,0,0	WT BL BK GN RD GY	Add Rectangle
0,0,0	1,1,1	0,0,0	WT BL BK GN RD GY	Add Tree
0,0,0	1,1,1	0,0,0	WT BL BK GN RD GY	Add Mountain
0,0,0	1,1,1	0,0,0	WT BL BK GN RD GY	Add House
0,0,0	1,1,1	0,0,0	WT BL BK GN RD GY	Add Cone
0,0,0	1,1,1	0,0,0	WT BL BK GN RD GY	Add Prism
Start Point: (x,y,z)	End Point: (x,y,z)			
0,0,0	1,1,1		WT BL BK GN RD GY	Add Vector

The "GameObjects" option allows you to add various types of objects to the 3D world. To access this option, click the "GameObjects" menu button in the menu bar. When you click on it, a window will appear with the following options:

- Cube
- Sphere
- Cylinder
- Capsule
- Plane
- Tree
- Cuboid, and
- Vector

To add an object to the 3D world:

- Use the syntax "x,y,z" with commas to input a value for a game object in the given text field.
- Fill in the three text fields for the object you want to add. These fields allow you to enter the object's center point, dimension, and Euler angles for rotating the object. For example, to create a cube with the default size at the center of the 3D world, you may set the center point to (0, 0, 0), the Euler angles or rotation to (0, 0, 0), and the dimensions to (1, 1, 1).
- To Add the object in the 3D world, click the "Add" button. The object will be positioned according to your specified location, dimension, and rotation.

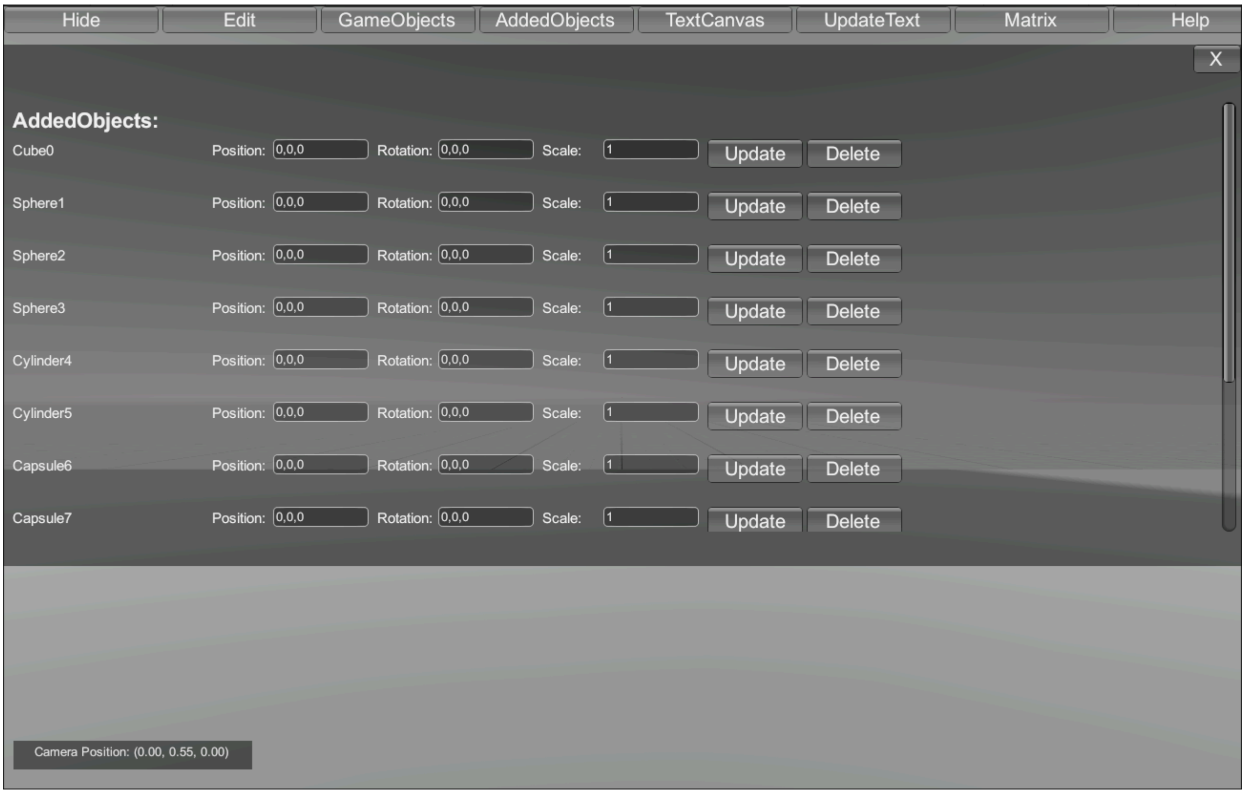
The "Vector" option allows you to add a vector object. This object represents a direction arrow with a start and end point.

To add a vector to the 3D world, do this:

- Fill in the start point and end point text fields. For example, to create a vector object starting from the origin and ending on point (1,1,1), you might specify the start point to (0, 0, 0) and the end point to (1, 1, 1).
- To Add the vector in the 3D world, click the "Add" button. The object will be positioned at the specified start point and will extend to the specified end point.

Note: When an object is added to the 3D world, the name of the object is automatically generated, and the object's type name is iterated with an integer. For example, the first cube you add will be named "Cube1", the second cube will be called "Cube2", and so on.

AddedObjects:



AddedObjects:

Cube0	Position: 0,0,0	Rotation: 0,0,0	Scale: 1	Update	Delete
Sphere1	Position: 0,0,0	Rotation: 0,0,0	Scale: 1	Update	Delete
Sphere2	Position: 0,0,0	Rotation: 0,0,0	Scale: 1	Update	Delete
Sphere3	Position: 0,0,0	Rotation: 0,0,0	Scale: 1	Update	Delete
Cylinder4	Position: 0,0,0	Rotation: 0,0,0	Scale: 1	Update	Delete
Cylinder5	Position: 0,0,0	Rotation: 0,0,0	Scale: 1	Update	Delete
Capsule6	Position: 0,0,0	Rotation: 0,0,0	Scale: 1	Update	Delete
Capsule7	Position: 0,0,0	Rotation: 0,0,0	Scale: 1	Update	Delete

Camera Position: (0.00, 0.55, 0.00)

The "AddedObjects" option allows you to view and modify the objects added to the 3D. To access this option, click the "AddedObjects" button in the menu bar.

When you click the "AddedObjects" button, a window showing a list of all the objects added to the 3D world from the "GameObjects" option will appear. The objects will be listed by their names, which are automatically generated when they are added to the scene.

To edit an object's properties, follow these steps:

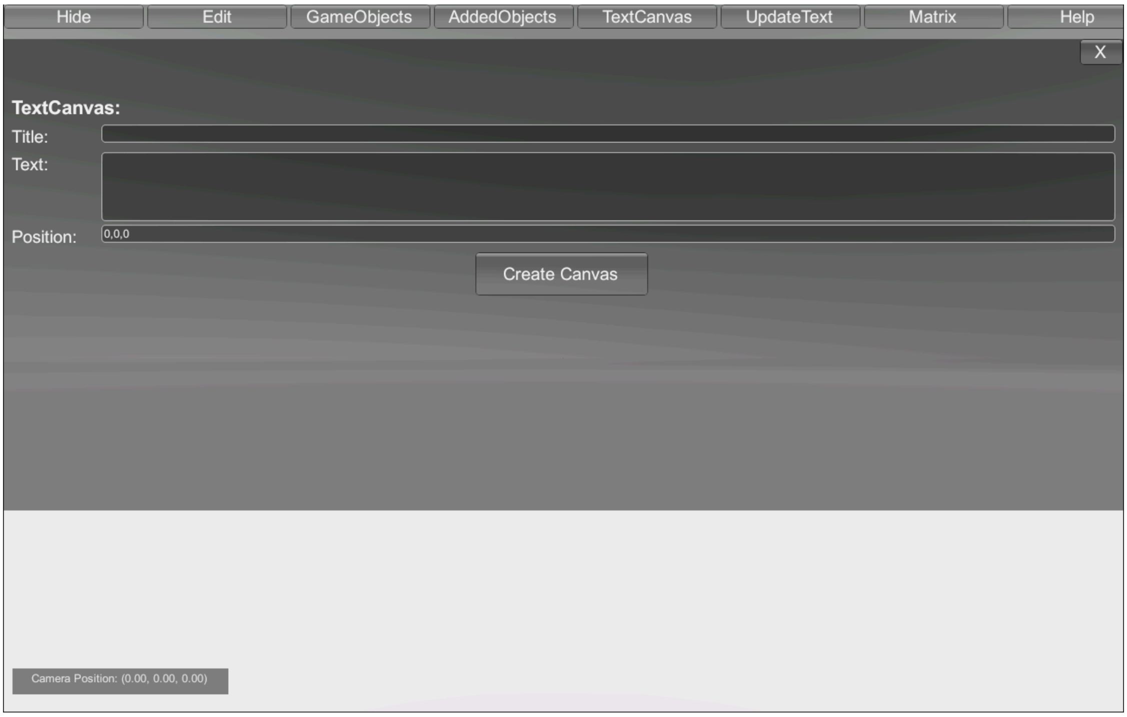
- Observe the name of the object you want to edit in the list on the right-hand side of the "AddedObjects" options window. The objects properties will be displayed in text fields.
- You may Edit the object's location, rotation, or dimension by entering new values in the relevant text fields. For example, to shift an object one unit to the right, you may modify the position from (0, 0, 0) to (1, 0, 0).

To delete an object from the scene, follow these steps:

- Find the name of the object you want to delete from the list of objects.
- Click on the "Delete" button. The object will be removed from the 3D world.

Note: Once an object has been deleted, it cannot be retrieved.

TextCanvas:



Title:

Text:

Position: 0,0,0

Camera Position: (0.00, 0.00, 0.00)

The " UpdateText " option allows you to manage the text canvases generated in the " UpdateText " option. Click the " UpdateText " button in the menu bar to access this option.

When you click on the " UpdateText " button, a window will appear on the right side of the screen; the window has a list of all the generated text canvases. When a canvas name is selected, a sub-panel with options to edit or delete the chosen canvas will show on the right side of the screen.

To update a text canvas, follow these steps:

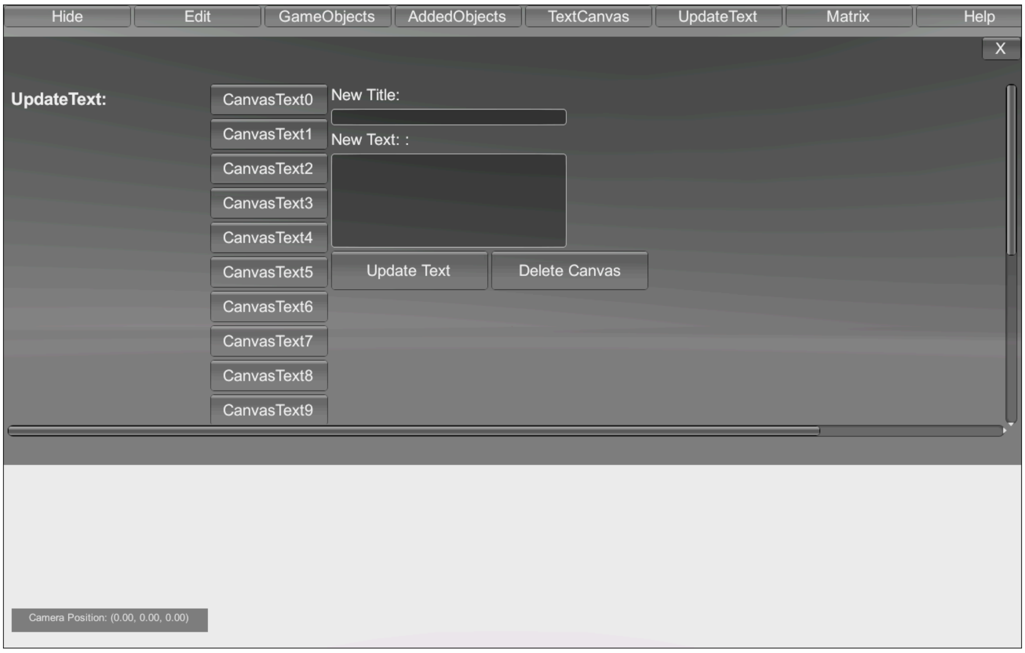
- Click on the name of the canvas you want to update in the " UpdateText " window.
- A sub-panel will appear on the right side of the screen that should display the canvas's current title, text, and position.
- Update the title, text, or position of the canvas as desired.
- Click the "Update Canvas" button to save your changes to the canvas.

To delete a text canvas, follow these steps:

- Click on the name of the canvas you want to delete in the " UpdateText " window.
- A sub-panel will appear on the right side of the screen that should display the canvas's current title, text, and position.
- Click the "Delete" button to remove the canvas from the 3D world.

Note: Deleting a canvas cannot be recovered, so make sure you have selected the correct canvas before clicking the "Delete" button

UpdateText:



The " UpdateText " option allows you to manage the text canvases generated in the " UpdateText " option. Click the " UpdateText " button in the menu bar to access this option.

When you click on the " UpdateText " button, a window will appear on the right side of the screen; the window has a list of all the generated text canvases. When a canvas name is selected, a sub-panel with options to edit or delete the chosen canvas will show on the right side of the screen.

To update a text canvas, follow these steps:

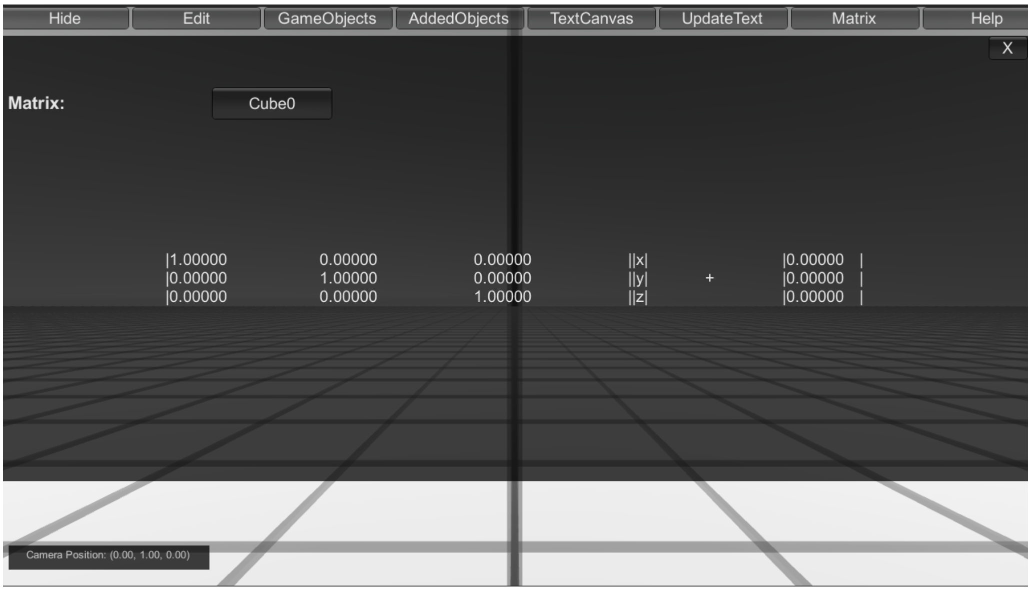
- Click on the name of the canvas you want to update in the " UpdateText " window.
- A sub-panel will appear on the right side of the screen that should display the canvas's current title, text, and position.
- Update the title, text, or position of the canvas as desired.
- Click the "Update Canvas" button to save your changes to the canvas.

To delete a text canvas, follow these steps:

- Click on the name of the canvas you want to delete in the " UpdateText " window.
- A sub-panel will appear on the right side of the screen that should display the canvas's current title, text, and position.
- Click the "Delete" button to remove the canvas from the 3D world.

Note: Deleting a canvas cannot be recovered, so make sure you have selected the correct canvas before clicking the "Delete" button

Matrix:



```
|1.00000     0.00000     0.00000     ||x|         |0.00000  |
|0.00000     1.00000     0.00000     ||y|    +    |0.00000  |
|0.00000     0.00000     1.00000     ||z|         |0.00000  |
```

```
Camera Position: (0.00, 1.00, 0.00)
```

The "Matrix" option allows you to view the transformation matrix of each game object in the scene. Click the " Matrix" button in the menu bar to access this option.

When you click on the "Matrix" button, a window will appear on the right side of the screen; the window has a list of all the generated 3D objects in the scene. The names of the objects are presented in a list format. When an object's name is clicked, a sub-panel displaying the object's transformation matrix will appear on the right side of the screen.

To view the transformation matrix of a game object, follow these steps:

- Click on the name of the object you want to view in the "Matrix" window.
- The transformation matrix of the selected object will be shown in a sub-panel that will appear on the right side of the screen.
- The transformation matrix will show the object's current position, rotation, and scale values in matrix form.
- Make use of this information to assist you in understanding how the object is positioned in the 3D world.

THE VR

How to build to the Meta Quest 2 headset?

Once you have created your 3D world, select "Save" from the "Edit" menu. You have two options for deploying your work to the Meta Quest 2 headset. The easy option is to build your project directly onto the headset, allowing you to test and improve your work. The second option is to upload your project to the Meta Quest App Lab, a platform that enables users to share their VR applications with a larger audience.

The easy option:

1. In the Unity Editor, click on "File" > "Build Settings" to open the Build Settings window.

Steps 2, 3, and 4 can be skipped because they are included in the project that has already been constructed.

2. Select "Android" as the platform and click on "Switch Platform", if it's not already selected. This will prepare your project for deployment to the Meta Quest 2.
3. Click on the "Player Settings" tab and ensure the company name and product name are filled in correctly.
4. Ensure that XR Plug-in Management's checkbox for android is enabled.

5. Put your Meta Quest 2 in developer mode by following these instructions: https://developer.Meta.com/documentation/native/android/mobile-device-setup/
6. Connect your Meta Quest 2 to your computer via USB.
7. In the Build Settings window, make sure to select Meta Quest as "Run Device."
8. Click the "Build and Run" button to generate an APK file and launch your VR application on the Meta Quest 2 headset. If you only want an APK file, click on "Build."

The second option:

Build your Unity project.

- Go to File - Build Settings
- Select Android as the platform
- Click on Player Settings and ensure that the following are set:
  Company Name, Product Name, Version, Package Name and Bundle Version Code.
- Click on Build and select a folder to save the APK file in.
- Wait for Unity to finish building the APK file.

Sign your APK file.

- Go to the Meta Developer Dashboard, assuming you already have Meta Developer Account.
- Click on the " MyApps" tab and select "Create New App."
- Give your application a name and select "Quest (App Lab)."
- Follow the prompts to create a new app, giving it a name, description, and other details as needed
- In the App Details section, click on the "No build uploaded to Production (App Lab) channel" > "Upload New Build" button and select the APK file you just built in Unity.
- Wait for the app to be uploaded and processed.
- Click the "Submit for Review" button to submit the app for review.
- Once your app has been approved, you might need to fill in other necessary details, including the app name, icon, etc.
- Once approved, your app will be available in the App Lab section of the Meta Store.

And that's it! You should be able to use these instructions to deploy your pre-built Unity project to the Meta Quest 2, whether you choose to build it directly into the headset or publish it to the Meta Quest App Lab for wider distribution.

IV Appendix D - Graphs over raw data

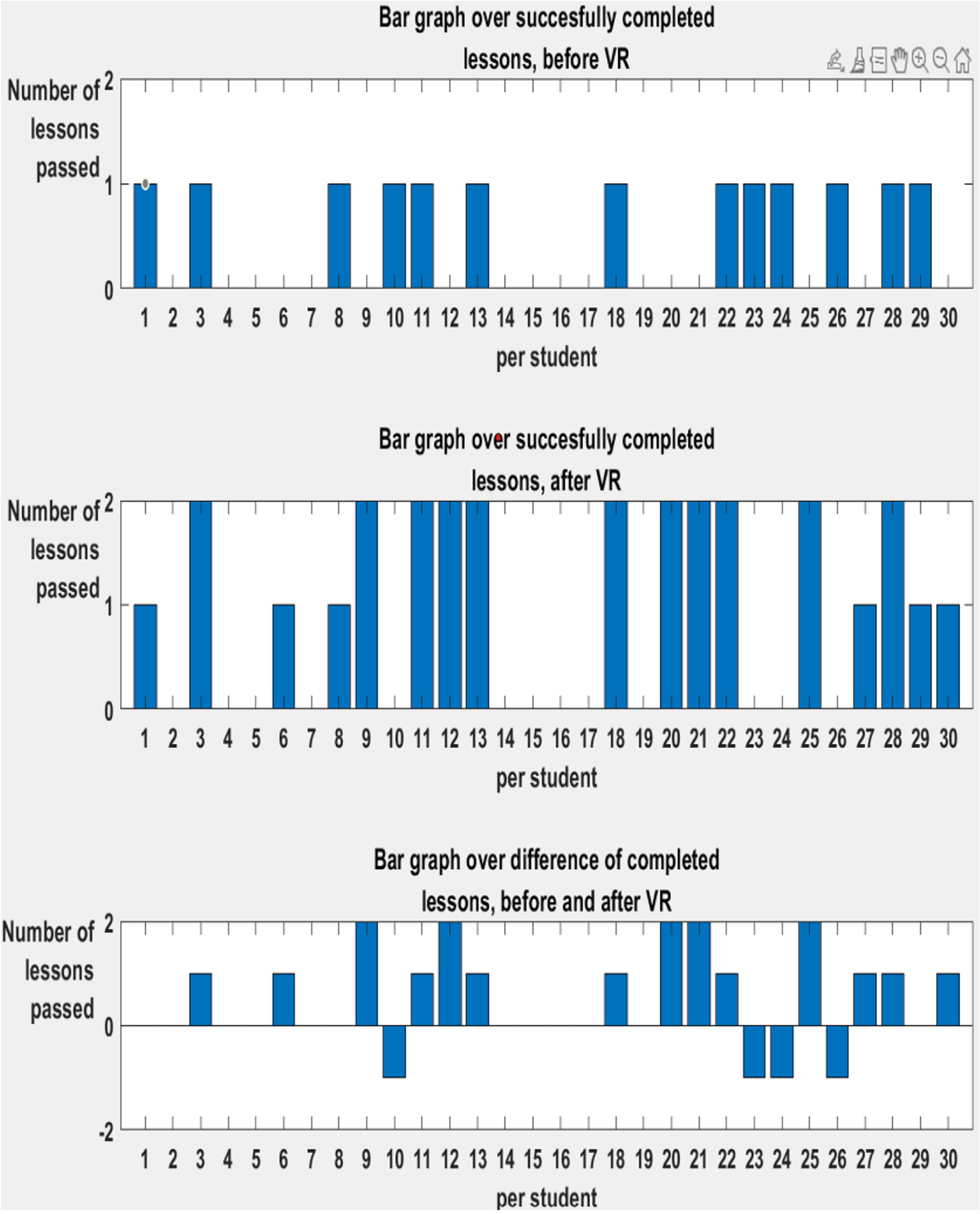


Figure 18: The students are numbered and tested before and after using VR. The first graph shows each student's successfully completed tests before using VR. The second graph shows each student's successfully completed tests after using VR. The third graph shows the difference between students' successfully completed tests before and after using VR. The tests are divided into two separate lessons. Nine students passed one more lesson after VR. Four students passed two more lessons after VR. Four students failed the previously passed lesson after using VR. The project team made this image with MATLAB.

V Appendix E - T-distribution

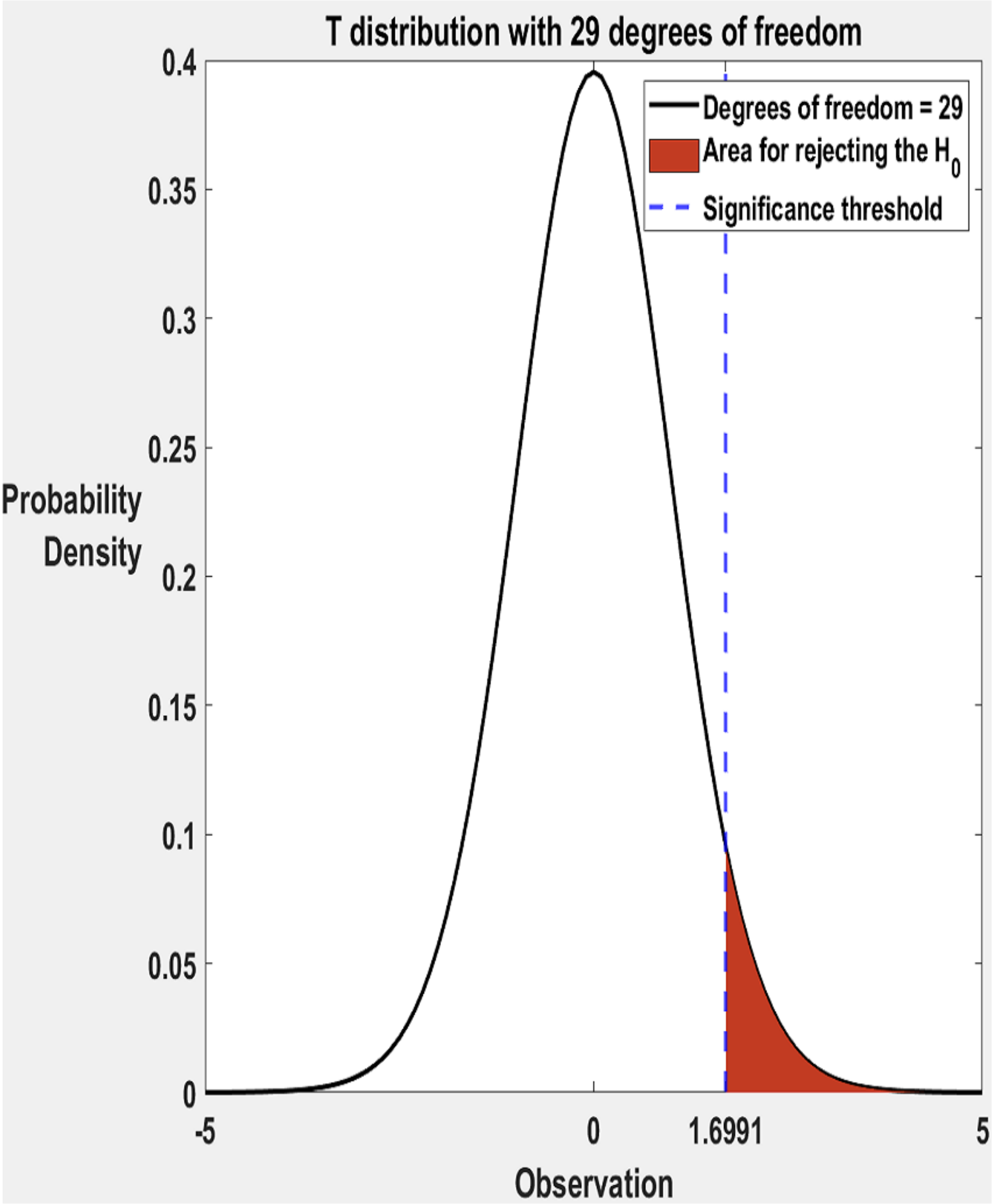


Figure 19: The significance threshold corresponds to the significance level of 0.05. The significance threshold is another way of writing the significance level. The H_0 is rejected if the p-value is inside the red area. The project team made this image with MATLAB.