

An Analytics algorithm for performance assessment in VR training

Michael Kentros



*Thesis submitted in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science and Engineering*

University of Crete

School of Sciences and Engineering

Computer Science Department

Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Associate Prof. George Papagiannakis

To my family

Abstract

Virtual Reality (VR) hardware and software solutions are rapidly evolving, providing developers with innovative technologies and tools to build applications, in the form of educational virtual experiences. In addition, the popularity of learning and training through such realistic and low-cost VR simulations is growing. Correspondingly, computer graphics and developers utilize the existing VR tools in combination with 3D game engines to create VR immersive experiences tailored around education. However, the current platforms for 3D interactive environments focus on producing mostly embedded gamification tools that serve the entertaining capabilities of VR technology.

Task performance assessment is a vital part of the learning process, and by providing valuable feedback it guides the learner towards improvement. In this thesis we addressed this issue, by developing a platform and algorithms that enable developers to accurately, rapidly and systematically author the automatic task performance assessment process of VR training scenarios. We introduce three generalized components, non-dependent to the context of the VR simulation, a) the VR analytics assessment framework, b) a Machine Learning (ML) algorithm capable of VR assessment and c) the VR Session Logger. In more detail, our analytics assessment framework utilizes user analytics for computing the user's score through an authoring tool for defining performance evaluation rules, whereas by employing supervised ML, our agent is capable of learning these rules directly from a subject matter expert's (SME) VR data. Furthermore, we present our novel algorithm for logging accurately VR sessions by recording the user's movement and tracking the resulting effects.

Ένας αλγόριθμος αναλυτικών στοιχείων για αξιολόγηση της απόδοσης εκμάθησης στην εικονική πραγματικότητα

Περίληψη

Τόσο το λογισμικό όσο και το υλισμικό που απευθύνεται στην εικονική πραγματικότητα (ΕΠ) εξελίσσεται με ραγδαίους ρυθμούς, παρέχοντας στους προγραμματιστές καινοτόμα εργαλεία για να αναπτύξουν εφαρμογές, όπως εκπαιδευτικές εικονικές εμπειρίες. Επιπροσθέτως, η δημοτικότητα της εκμάθησης και της εκπαίδευσης μέσα από ΕΠ προσομοιώσεις υψηλού ρεαλισμού και παράλληλα χαμηλού κόστους αυξάνεται. Έτσι, προγραμματιστές των γραφικών υπολογιστών αξιοποιούν τα υπάρχοντα εργαλεία ΕΠ σε συνδυασμό με τις μηχανές παιχνιδιών για τρισδιάστατα γραφικά, για να δημιουργήσουν εμβυθιστικές εφαρμογές εξατομικευμένες στην εκπαίδευση. Εντούτοις, οι σημερινές πλατφόρμες για κατασκευή τρισδιάστατων περιβαλλόντων, εστιάζουν κυρίως στην παραγωγή ενσωματωμένων εργαλείων παιχνιδιοποίησης για την υποστήριξη των ψυχαγωγικών δυνατοτήτων της τεχνολογίας ΕΠ.

Η εμπεριστατωμένη αξιολόγηση της απόδοσης είναι ένα αναγκαίο κομμάτι στην διαδικασία της μάθησης, το οποίο παρέχοντας εποικοδομητική κριτική οδηγεί τους μαθητευόμενους στην βελτίωση. Σε αυτήν την διατριβή απευθυνθήκαμε σε αυτό το πρόβλημα, κατασκευάζοντας μία πλατφόρμα και αλγόριθμους που παρέχουν την δυνατότητα στους προγραμματιστές να ορίσουν την αυτοματοποιημένη αξιολόγηση της απόδοσης σε εκπαιδευτικές εφαρμογές ΕΠ, με ακρίβεια, ταχύτητα και συστηματικότητα. Παρουσιάζουμε τρία γενικευμένα δομικά στοιχεία, μη-εξαρτόμενα από το περιεχόμενο της προσομοίωσης ΕΠ, α) το σχεδιαστικό πρότυπο αναλυτικών στοιχείων για αξιολόγηση σε ΕΠ, β) έναν αλγόριθμο μηχανικής μάθησης (ΜΜ) ικανό για αξιολόγηση εργασιών σε ΕΠ και γ) τον καταγραφέα συνεδρίας ΕΠ. Πιο συγκεκριμένα, το σχεδιαστικό πρότυπο αναλυτικών στοιχείων αξιοποιεί δεδομένα ανάλυσης για να υπολογίσει την βαθμολογία του χρήστη, ορισμένα μέσω ενός συγγραφικού εργαλείου στην μορφή κανόνων αξιολόγησης, ενώ αξιοποιώντας επιβλεπόμενη μάθηση, ο αλγόριθμός μας είναι ικανός να μάθει τους κανόνες απευθείας μέσα από τα δεδομένα ΕΠ. Επιπλέον, παρουσιάζουμε έναν καινοτόμο αλγόριθμο για ακριβή καταγραφή συνεδριών ΕΠ, καταγράφοντας τις κινήσεις των χρηστών και ιχνηλατώντας τα αποτελέσματά τους.

Attestation

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I certify that this dissertation reports original work by me during my university project except for the following (*adjust according to the circumstances*):

- This thesis was primary developed on top of Unity3D Game Engine which belongs to Unity.
- The steamVR SDK unity plugin for tracking the user's movement and input was used for the implementation of the demos, which belongs to Valve. As well as, the NewtonVR Unity plugin which was used for the interaction of the user with the virtual environment.
- The implementation of this project was influenced by the MAGES SDK which belongs to the ORamaVR company.

Acknowledgments

I would like to thank my supervisor, Professor George Papagiannakis, for his support, motivation, and providing the main idea for this thesis.

Special thanks to my co-workers, namely Paul Zikas for providing necessary help for the writing of this thesis, through available research and academic work, Nikos Lydatakis for always pushing me to raise my programming standards, Steve Kateros for providing helpful ideas through our debates, Stratos Geronikolakis for evaluating my work and giving me the needed strength to keep going and finally Ilias Chrysovergis for our countless brainstorming sessions on Machine Learning.

Finally, I would like to thank my family and especially Despoina Gavgiotaki for mentally supporting me, in order to go through difficult and stressful phases of this thesis. I am always grateful for their support.

Table of Contents

Abstract	3
Ένας αλγόριθμος αναλυτικών στοιχείων για αξιολόγηση της απόδοσης εκμάθησης στην εικονική πραγματικότητα .	4
Attestation.....	5
Acknowledgments.....	6
Table of Contents	7
List of Tables	9
List of Figures	10
1 Introduction	12
1.1 Background and Context.....	12
1.2 Scope and Objectives	13
1.3 Achievements	13
1.4 Overview of Dissertation	14
2 State of the Art.....	15
2.1 Training in virtual environments	15
2.2 Analytics assessment in 3D Environments.....	16
2.3 Authoring tools in training and learning	19
2.4 Supervised Machine Learning for Evaluation	21
2.5 Our publications and previous work related to this thesis	22
3 Analytics System for VR training scenarios	25
3.1 Overview of the analytics system	25
3.2 Assessment.....	27
3.2.1 Assessment Manager	28
3.2.2 Scoring Factors Design	29
3.2.3 Scoring Factors Prototypes.....	31
3.3 Setting up Scoring Factors	34
3.3.1 JSON Files	35
4 Supervised Learning based VR assessment.....	38
4.1 Machine Learning for VR assessment	39

4.1.1	Data Collection.....	40
4.1.2	Training CNN models.....	41
4.1.3	Predicting at runtime.....	43
4.2	VR Session Logging.....	45
4.2.1	Logging Data.....	45
4.2.2	Non-Deterministic Physics.....	46
4.2.3	Propagated Logging.....	47
5	Demos	50
5.1	VR Analytics based Assessment	50
5.2	Machine Learning Assessment.....	53
5.3	Logging VR Sessions.....	53
6	Evaluation.....	56
6.1	Self-assessment	56
6.1.1	Analytics Based Assessment	56
6.1.2	Machine Learning Assessment.....	57
6.1.3	VR Logger	57
6.2	Qualitative Evaluation	58
6.3	Metrics	64
6.3.1	Machine Learning Optimization	64
6.3.2	VR Logger Performance Metrics.....	67
7	Conclusion.....	69
7.1	Achievement of Goal.....	69
7.2	Future Work.....	69
	References	72
	Appendix 1 – Analytics Assessment	76
	Appendix 2 – VR Session Logging	79
	Appendix 3 – Supervised ML Functions	81

List of Tables

<i>Table 1. Description of the potential usage of each scoring factor, as well as their needed parameters.</i>	<i>34</i>
<i>Table 2. Example of the format logged data are stored in file for a specific virtual hand.</i>	<i>48</i>
<i>Table 3. Performance metrics for VR Logger Module in our VR playground environment.</i>	<i>67</i>
<i>Table 4. Performance metrics for VR Logger Module in our VR stress test scenario.</i>	<i>68</i>

List of Figures

Figure 1. Graph taken from [24]. Portraying the importance of the assessment component in Serious Games.....	16
Figure 2. Real-time monitoring of aircrafts inside the Microsoft Flight Simulation.	17
Figure 3. Logical Architecture of RAGE. A generic learning analytics system for serious games.	17
Figure 4. Performance insights from mobalytics.com, which uses the API provided from the game League of Legends, to get raw data of a player's session, analyze it and give feedback to the user in a meaningful way.....	18
Figure 5. The Emergo methodology for serious game case development. Taken from [35].	20
Figure 6. Usage of the FOCUS system, where recommendations are presented on top of the learning material (book).	21
Figure 7. In the left image the collaboration capabilities of M.A.G.E.S. are showcased in a Resuscitative Endovascular Balloon Occlusion of the Aorta VR training scenario. On the right image the nasal swab test conducted for testing people with possible COVID-19 is depicted.....	23
Figure 9. Interaction using the pinch gesture; the user can move a 3D object around the environment.....	23
Figure 8. The overview of the application which focuses on the restoration of Knossos (right hologram) or Sponza (left hologram).	23
Figure 10. (Left) The instructor avatar we created for guiding the patients through the hand motor exercises. (Right) A patient performing the exercises in the virtual environment.	24
Figure 11. Workflow pipeline overview of our analytics platform	26
Figure 12. Algorithm of performance score calculation for each task.	27
Figure 13. Scoring factor usage example. In this diagram we describe how multiple scoring factors can be combined to achieve accurate evolution of user performance in each task.	28
Figure 14. Architecture diagram of Scoring Factors Components	31
Figure 15. UI template for the question scoring factor.	32
Figure 16. JSON example analytics file that stores all user defined parameters.	36
Figure 17. On the left the uninitialized analytics editor panel is shown of a specific task. On the right the filled panel is presented with the developer's set parameters and the references to the needed assets.	37
Figure 18. Diagram describing our VR ML training architecture.	39
Figure 19. The UI used for classifying each generated position.	40
Figure 20. Our sequential convolutional model structure with the output dimensions on each layer.	41
Figure 21. Structure of multi-view model. The four CNNs are created similarly as shown in figure 17.	43
Figure 22. Algorithm of prediction at runtime.....	44
Figure 23. On the left showing the result of a correct orientation and on the right a wrong one.	45
Figure 24. Algorithmic explanation of propagated logging.....	48
Figure 25. On the left only the transformations of the user's hands and camera are logged. On the middle image the user has grabbed the object adding to it the component responsible for logging. On the right image the object is thrown propagating the logging component to the rest of the cubes.	49

Figure 26. First task (left) the user needs to answer the question. Second task (middle) the user needs to clean the dust spots on the cube. Third task (right) the user needs to place the box on the shelf.	51
Figure 27. On the left the buttons for performing and undoing a task are shown. On the middle the panel showing the results of the previous task. On the right the panel for presenting the results of each scoring factor is shown...	52
Figure 28. Right: The UI used for feeding information to the user. Left: The UI spawned when user performs an error.	52
Figure 29. Our demo showcasing our ML implementation. The green cube serves as the area of interest with the four cameras around it for generating images of the statue's position.....	53
Figure 30. VR playground for showcasing the capabilities of our VR Logger.....	54
Figure 31. Image from our playback demo, where the instructor is shown opening a drawer.	55
Figure 32 Evaluation of task 1 of the Cardboard Box Preservation scenario.	59
Figure 33. Evaluation of task 2 of the Cardboard Box Preservation scenario.	60
Figure 34. Evaluation of task 3 of the Cardboard Box Preservation scenario.	61
Figure 35. The evaluation demo comparing the neural network with the scoring factor. The left statue and placement box uses our ML implementation while the right use the placement factor.	62
Figure 36. Users evaluation on the assessment accuracy of each implementation.	63
Figure 37. Average attempted tries for each placement.	64
Figure 38. On the left the accuracy graph for 20 epochs is depicted, on the middle for 50 and on the right for 70. These graphs depict the training process with different random seeds.	65
Figure 39. The confusion matrices for the above training runs, showcasing how each model behaves to new data (testing dataset). On the left the confusion matrix for 20 epochs is depicted, on the middle for 50 and on the right for 70.....	65
Figure 40 Confusion matrices graphs for Batch_size=10 on left and Batch_size=6 on the right. Both produced from the same random seed and dataset.	66
Figure 41. CNN accuracy on each epoch. The SGD optimizer was used for the left graph while the Adadelta optimizer was used for the right.	67
Figure 42. Images of the stress test scenario. On the left the cubes are in idle. On the right force was added to them.	68

1 Introduction

1.1 Background and Context

At all levels of the educational system and even later in our workplaces we get accustomed to the process of completing assignments and expecting feedback for our performance. Throughout the years, this process has been studied and refined numerous times and with the adoption of computer science in our everyday lives, multiple parts of it are becoming digitized and automated. Among the many examples are, online multiple choice quizzes that are immediately graded; athletes that log their training sessions and expect to receive immediate feedback on their performance; supervisors that generate reports for their employees' work. All these applications share a common requirement, a clear and well-defined system for assessment.

Evaluation systems have become mandatory in all environments where individuals learn under the guidance of others. In academia, students expect their teachers and professors to rate their work with accuracy and vice versa, while also providing some form of feedback. Educators often follow systematic approaches for this procedure in order to enforce high integrity in their assessment. Once an evaluation method is defined, generating feedback for the student becomes a straightforward process. Since students can understand the reasoning behind their grade, recognizing their mistakes is sequential. In addition, by employing a structured approach, meaningful insights can be generated for the improvement of future exams, and the educational process in general, e.g. ranking on the most common wrong answers.

Nowadays, it is increasingly common for managers to establish scoring rules in their workplaces in order to systematically assess their team members' performance. Even though these systems are not comprehensive when compared with previous data they can provide useful insights. Agile tools, such as Jira¹, provide interfaces for defining the importance and the complexity of each task, and can produce insights for an individual's (or the team's) performance. Implementing accurate grading rules that fit all tasks in a workplace can be a challenging procedure, due to their variance. For this reason, scoring systems, are not fully autonomous or structured and the supervisors are in charge of deciding and altering the structure based on their experience. Research has been done on how we can employ ML to intelligently and automatically detect user activity by collecting analytics data from their everyday tasks [34] and provide insights to increase productivity [21].

Computer graphics applications are becoming more and more prevalent each day. Programmers and developers utilize them for a variety of purposes, entertainment [4], learning [16], training [17] and even research [38]. Traditional 3D serious games have been the long-standing favored approach for developing

¹ <https://www.atlassian.com/software/jira>

experiences targeting the education of students or the training of employees. Since the rapid advancement of Cross Reality (XR) technologies, this preference is shifting towards immersive simulations, that can provide a virtual experience close to reality. Every passing year, increasingly more applications are developed for virtual and augmented reality that opt to enrich the educational process in a specific field. Human performance in Virtual Reality training environments highly depends on the degree of presence that the user experiences, hence in order for an application to provide accurate and objective scoring, the experience of being in a virtual environment, even when one is situated in another, must be vividly present in the users. In addition, virtual reality technology can produce high immersive environments with countless options for developers to shape the virtual training scenario to best fit the trainee's needs. Finally, the interaction freedom that VR offers makes it applicable in multiple fields.

Consequently, the need for swift, accurate and high-fidelity XR content creation has emerged. Most of the 3D Game Engines have adapted their framework to support these technologies, focus on providing tools for creating games. Although, some of these features are shared in VR applications scoping to train their users, modules such as task authoring, assessment authoring and collaboration are not yet established. The absence of these features on the pre-established frameworks brings rise to the need for a software development kit concentrating on education and training. One of the most important attributes of such an SDK is the ability to provide a generic, systematic and accurate method for scoring the user's performance in scenarios across different educational fields.

1.2 Scope and Objectives

The main goal of this thesis is to provide a re-usable task assessment process for VR training scenarios. The objective of this project is the implementation of two approaches: a) utilizing user analytics and computing the user's score through a structured framework, b) by employing supervised machine learning, for learning directly from the expert's input. Both of these methods opt to be generalized and non-dependent to the context of the VR simulation. In addition, these two solutions will be compared on the functionality they provide to assess the user's actions with accuracy. Furthermore, we developed authoring tools to provide rapid generation of the assessment content, while also improving the developer's experience. Finally, due to the rapid growth of the VR field, our system should be scalable and flexible to accommodate new features regarding recording the user's interaction with the environment and the resulting assessment.

1.3 Achievements

In this project we achieved to design and implement a formalized, generic assessment system that can be used for recording analytics derived from interaction events common in all VR applications, rendering it independent of the content of the VR training scenario. By employing a structured module, we were able to produce an authoring tool that rapidly reduces the developer time needed for defining the evaluation

rules. Additionally, by exploiting the vast capabilities of the Unity Editor we managed to integrate our editor on top of Unity's, reducing the required knowledge from the developer, to that of being familiar with the Unity's interface. This way non-programmer's, such as artists, can contribute to this process without the need of learning a new platform or VR programming.

Usually, the VR programmer that develops the simulation is not a specialist in the field that the VR training is referring to, thus guidance from an expert is mandatory, especially for the integrity and the accuracy of the scoring system. For this reason, we provide a solution to this issue by employing supervised machine learning algorithms, specifically convolutional neural networks that learn from images captured directly from inside the virtual environment. By enabling users to define assessment rules from inside the Virtual environment, we can delegate the problem of accurately scoring the user to experts of the field that the training simulation is referring to. In addition, ML algorithms allow for straightforward authoring, where all the parameters are learned directly from the user's virtual movement, and not from user interfaces which, when abused, tend to have negative impact in the user's immersion.

Finally, a generic implementation for recording the user's session was achieved. Our VR logger is capable of tracking and storing users' movements and their interactions, as well as the resulting events. In addition, a method for reconstructing the session was implemented in order to assess our logger's accuracy.

1.4 Overview of Dissertation

This dissertation will start by presenting the existing state of the art for the areas of, authoring tools, performance assessment in 3D virtual environments, tools editing VR content directly from the Virtual environment and logging the user's session 3D environments. On chapter 3, the implementation of our rule-based assessment system will be explained, more specifically the underlying structure, the types of analytics data we observe and how the authoring process is sped up using our authoring tool. Continuing on chapter 4, we will present our implementation of the intelligent agent that utilizes convolution neural networks to learn from an expert's data and evaluates of the user's actions. Additionally, our novel implementation for logging a VR session will be presented in chapter 4. Moving on to chapter 5, we will explain and present the demos that we developed in order to showcase our work, which consist of the *Cardboard Box Preservation* training scenario, a virtual playground that records the user's interactions and offers playback features and finally a virtual environment where our pre-trained ML agent is evaluating the user's placement of an object. Furthermore, we will provide performance metrics for our VR Logger and for optimizing the training and accuracy of our ML agent, as well as user-based qualitative evaluation on the assessment accuracy of each evaluation method. Finally, we will conclude with our planned future work for this project.

2 State of the Art

In this chapter, previous work regarding assessment in virtual environments, both VR and 3D graphics, is presented. Additionally, projects regarding assessment through machine learning and analytics insights are brought up.

2.1 Training in virtual environments

The VR technological space has seen rapid growth the last decade, both in terms of hardware and software. Virtual Reality content creation tools allow the developers to bring to life immersive experiences with new opportunities for interacting with the virtual environment that traditional 3D graphic applications lack. This led to the altering of popular belief that VR applications are meant only for entertainment purposes, but they can also serve as rich learning environments. In addition, VR applications tools provide new opportunities for collaborative learning [27] through entertaining environments, making training more meaningful for the trainee [1]. However, current state of the art VR collaborative platforms such as Facebook Spaces and VRChat, focus on social aspects and portray themselves as VR social media applications, neglecting their potential for collaborative education and learning. In addition, the larger part of VR simulations overlooks the importance of meaningful evaluation metrics in improving the trainee's skills, as highlighted in [9] (chapter "Metrics and Feedback").

The main advantages that VR simulations offer, compared to traditional 3D graphics applications, are the principles of immersion and embodiment [33]. VR simulations are capable of immersing the users through high fidelity and high-realism environments, enabling the trainee to perform his tasks as he would in a real-world scenario. To achieve this level of immersion, visual 3D graphics, audio feedback and accurate methods of interacting with the 3D world are equally important. By providing physics-based interactions, the simulation has the potential to empower the user with multiple options on how to perform his tasks, utilizing his skills in a natural approach and not being restricted by a predefined implementation. VR applications can provide almost the same level of interaction fidelity as the real world, providing a psychomotor training simulation, and due to its digitized form, it can be easily populated with modules that output immediate feedback to the user. Embodiment is an equally important VR principle which describes the feeling of being inside the virtual environment. The feeling of presence allows the user to transfer the sensations and physical movements from this virtual body to his biological one and, as a result, increase his comprehension of the training material [22]. Hence, embodiment is crucial in virtual psychomotor training scenarios since it has a high impact on the muscle memory and skill transfer from the virtual to the real world [6].

VR training has already started getting exposure in various industry fields. Big retailers, such as Walmart, are using it to train their employees in scenarios that are hard to replicate, e.g. accurately reacting to fire emergencies and soft-skills for handling dissatisfied customers. BMW trains its mechanical engineers through immersive simulations focusing on safety in the workplace. In the medical field, multiple institutes are training their surgeons and their nurses in complex psychomotor operations in virtual operating rooms in order to continuously grow their muscle memory but also memorize long procedures through cognitive training [19], [39]. Finally, VR training simulations have also been adopted by the army both for training, and PTSD therapy. As an example, the US army is utilizing the VR training to create various virtual combat fields, for training in multiple branches such as navy, medicine, air force and infantry. In addition, application such as Virtual Afghanistan Village, aim in treating PTSD syndromes through exposure therapy to familiar environments from the safety of their workplace [30].

2.2 Analytics assessment in 3D Environments

Due to the complexity and diversity of virtual environments monitoring and evaluating the user's actions within the environment proves challenging. Even in non-VR 3D graphics environments, where the input of the user is limited through the keyboard and mouse, each keystroke can lead to sequential events that are hard to predict, thus hard to evaluate through a structured assessment platform.

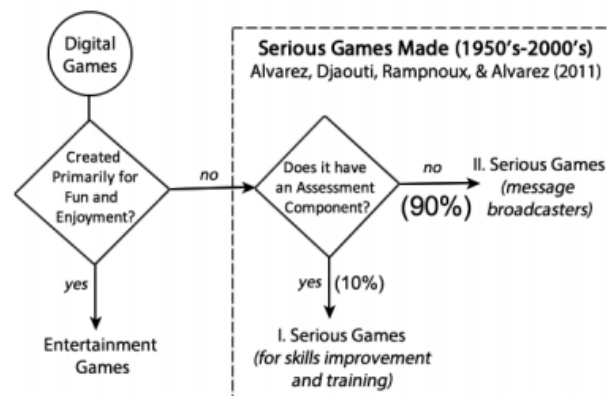


Figure 1. Graph taken from [24]. Portraying the importance of the assessment component in Serious Games.

Serious games are being explored [14] as a tool for learning and assessment for their potential to provide more valid evaluation methods compared to traditional approaches, through interactive, immersive and “fun” environments [25]. As mentioned in [24], and depicted in figure 1, an analytics assessment component is necessary for any serious game opting for improvements in skill and training. Microsoft's Flight simulator was one of the earliest serious games with multiple iterations since the first version (1982).

It provides real-time metrics for most variables of the pilot controls and developers can access them by developing a plugin using Microsoft's SimConnect SDK, shown in figure 2, to the game's engine.

SimConnect Monitoring AI Objects

Connect to ESP

Active entities in Microsoft ESP are monitored below:

Object Type	ESP Model	ATC Model	Tail Number	Airline	Flight Number	Latitude	Longitude	Altitude	Heading	Ground Speed
User	Cessna Skyhawk 172SP G1000	C172	N700MS			47.8968	-122.28	185.870	115.216	53.7659
Airplane	Mooney Bravo Retro	M20T	C-GZSN		7993	48.0551	-122.87	5334.27	176.964	185.443
Airplane	Boeing 737-800 Paint5	B738	N73032	Pacific	3034	47.5268	-122.18	1693.22	114.116	338.972
Airplane	Airbus A321 Paint2	A321	N5089L	World T...		49.1981	-123.17	7.17120	233.976	0
Airplane	Airbus A321 Paint4	A321	N7988E	Orbit	4822	49.1917	-123.18	7.17120	357.306	0
Airplane	Mooney Bravo	M20T	N3139R			49.1539	-121.94	10.8956	357.536	0
Airplane	Cessna Skyhawk 172SP Paint3	C172	N4442J		9173	48.2914	-122.69	2697.07	311.428	97.3995
Airplane	Airbus A321 Paint4	A321	N6266X	Orbit	6059	49.1912	-123.22	575.829	278.711	299.437
Airplane	Boeing 737-800 Paint2	B738	N68162	World T...		47.9000	-122.27	187.744	90.7494	0
Airplane	Beech King Air 350 Paint1	B350	N9594U			49.1814	-123.16	5.65303	93.4013	0
Airplane	Cessna Skyhawk 172SP Paint2	C172	N3466D		8325	48.3484	-122.70	1805.86	296.050	100.679

Monitor Aircraft

Monitor Vehicles

Monitor Ships

Disconnect from ESP

Figure 2. Real-time monitoring of aircrafts inside the Microsoft Flight Simulation.

Providing a platform capable of all-around, accurate evaluation that captures the diverse range of data that are generated from serious games, and also analyzes these data into feedback metrics effective in boosting the training of the player, proves to be an extremely challenging task. As described in [5] evaluating the assessment methods of each serious game is not a trivial task, but it is feasible to appraise a holistic platform that is applicable in different games. Multiple tools focusing on developing serious games for learning and training have been developed and reviewed [35], [7]. The Realizing on Applied Gaming Eco-System (RAGE), which architecture is show in figure 3, was developed with the goal of creating modular components specifically for serious games, focusing mostly on learning-analytics tools that can be integrated in different types of 3D games.

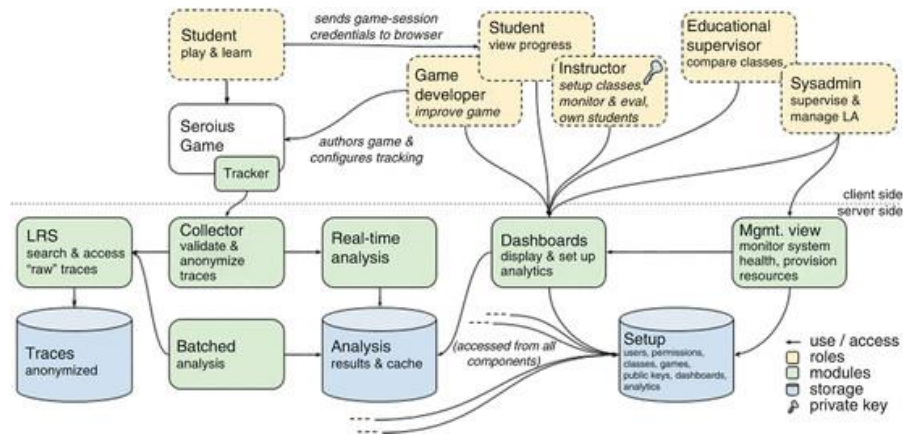


Figure 3. Logical Architecture of RAGE. A generic learning analytics system for serious games.

E-Sports stands for electronic sports and consists competitive computer games. Even though the standing of E-Sports as sports is controversial [15] the rapid growth of their popularity has attracted many investors expanding their influence [13]. This led to the need of analyzing the players' and the team's game sessions, both for competitive reasons but also for personal improvement. Although, to the best of our

knowledge, an assessment platform applicable to different type of e-sports, has not been developed, multiple performance analytics applications have been created, each targeting a specific e-sport game. Due to the big amount of these sites and the diversity of the 3D games, various types of methods for accessing the raw data have been developed. In all cases, the 3D game is responsible for providing the raw data and the two most reused methods are through an API or through options inside the game to record logs of all user's inputs and the resulting events in the 3D environment. In figure 4, the user interface of the mobalytics² application is presented.

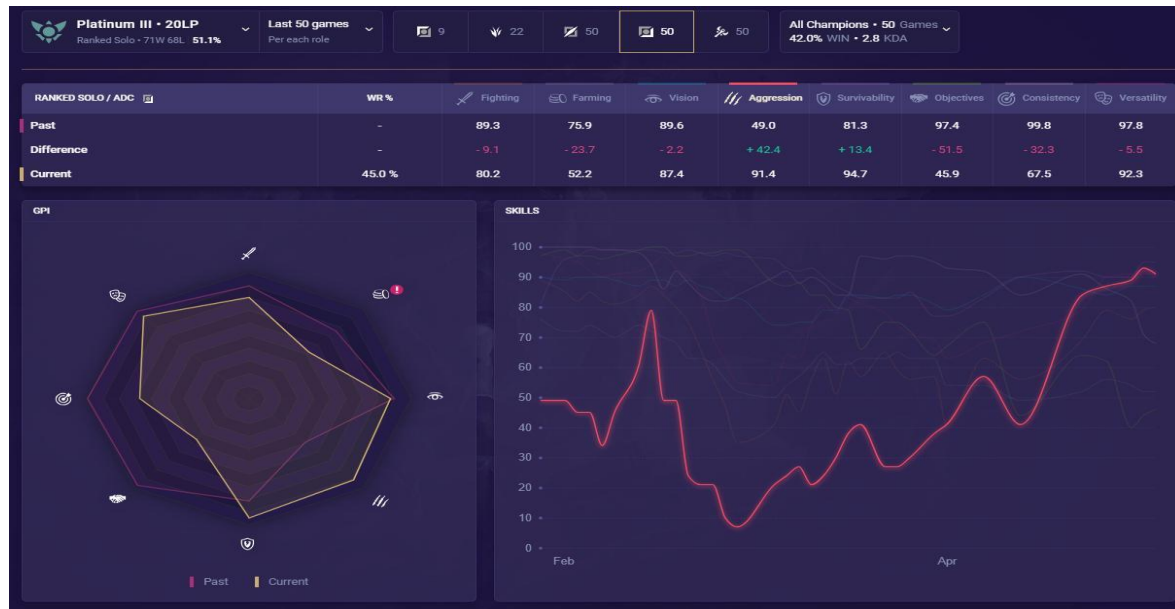


Figure 4. Performance insights from mobalytics.com, which uses the API provided from the game League of Legends, to get raw data of a player's session, analyze it and give feedback to the user in a meaningful way.

At first sight, it might seem that capturing and providing analytics metrics from VR environments is a procedure with similar level of difficulty compared to traditional 3D environments. But due to the increased level of freedom for physically interacting with the environments, a plethora of different events can be generated, rendering the process of capturing and analyzing these VR raw data challenging. Startups such as cognitive3D³ and VADR⁴ aim to provide a solution by creating SDK's integrated in game engines that log raw data from the VR hardware, e.g. user gaze, movement analysis and object engagement. These solutions focus mostly on tracking and visualizing the user's performance and the assessment process is carried out by a subject matter expert (SME). Even though these SDK's are generalized and can be seamlessly applied to different kinds of simulations by not providing assessment in-game, they exhibit two flaws; a) they need the review of a person which can lead to inconsistencies in the evaluation process when

² <https://mobalytics.gg/>

³ <https://cognitive3d.com/>

⁴ <https://vadr.io/>

multiple reviewers are involved and b) they do not provide immediate contextual feedback to the performer, thus not providing him the necessary knowledge to correct them straight away.

As mentioned in section 2.1, VR applications intend to provide high-realism environments, combined with its adaptation in fields such as medicine, where mistakes have substantial impact, research for scoring complex tasks with valid and context specific multi-metric scoring methods has emerged. The VR Heidelberg score [31] is an example of such method, where a VR scoring system was implemented for the sole purpose of evaluating a laparoscopic operation task. Another similar example is [32] where they used similar assessment methods for a hysteroscopy operation task. It is important to mention that in both of these scenarios high-priced equipment was used, adding to the difficulty of generalizing such problems. So, if the target of a VR evaluation platform is to be able to track and assess all the possible actions done by the performer in the virtual world, how can we design and implement it? Our answer is, by monitoring general VR metrics, common in every VR environment, and providing the SME or developer with tools to contextualize them depending on the simulation.

2.3 Authoring tools in training and learning

Authoring tools that are tailored for educational purposes have been used for some time now. Platforms such as elucidate⁵, adobe captivate⁶, gomo⁷ and others provide teachers and trainers tools for creating digital content scoping to transfer knowledge. Two principal features have attracted the attention of researchers and software developers: reducing development cost, and allowing the subject matter experts to be involved in the creation process. Combined with their potential interoperability, reusability and portability they prove to be essential components of any type of content creation platform.

Although, authoring tools for 2D content creation are well established in academia, development frameworks for 3D serious games are still undergoing research and innovation. 3D Game Engines such as Unity and UE4 focus mostly on providing tools for game developers targeting entertainment, also these engines have a steep learning curve and require some programming background proving difficult for teachers to create content using them [29]. Besides these limitations, existing tools that mostly focus on 3D entertainment games lack suitable analytics logging, which is an essential component both for the performer but also the trainee and it can also be used for evaluating research on the actual effects of the application. In recent years, toolkits and design methodologies for developing such learning content creation platforms have emerged. Emergo [35] is both a methodology, shown in figure 5, and a toolkit that focuses mostly on cognitive skills acquired from observing and taking notes while answering questions and deciding the

⁵ <https://www.elucidat.com/>

⁶ <https://www.adobe.com/products/captivate.html>

⁷ <https://www.gomolearning.com/>

narrative of the serious game. Multiple evaluations have been made on this platform which have marked it at least “adequate” [35], [36]. SCORM and xAPI (experience API) are two generic analytics gathering tools. The SCORM API provides straightforward functions for tracking variables and storing the results online. The xAPI is a more complex solution which provides tools for recording events in a scenario and storing the results in JSON files which associate actors (mails, social media accounts) with activities (RESTFUL API functions that return a result). Even though, these analytics platforms do not provide an immediate solution to our problem studying them and applying part of their solutions in virtual reality simulations could bring forth positive results.

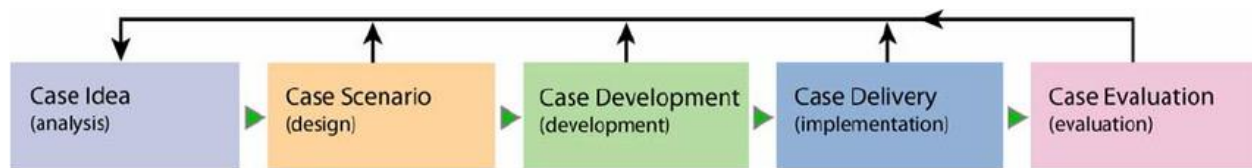


Figure 5. The Emergo methodology for serious game case development. Taken from [35].

An important tool of 3D content creation, arguably the most important, are the 3D game engines. Even though, programming knowledge is a valuable asset for game development, engines such as Unity 3D⁸ and Unreal Engine 4⁹ have made it possible, although hard, for non-programmers to contribute to the development process. By providing a highly configurable environment that software engineers can augment by developing add-ons plugins, tools that target non-programmers have been developed [40]. Additionally, they provide visual scripting capabilities, (UE4 Blueprints) which has a much easier learning curve than object oriented programming through C++. Even though, these visual scripting tools can be used without any programming knowledge, they are designed for a generic usage, revolving around rendering, 3D object management, 3D movement and many more; which leads to developers (programmers and non-programmers) having to implement their own features that scope in assessing the user’s performance in the training simulation.

Editing from inside the VR environment has been the “holy grail” of VR development for a few years now. Both, Unity3D and Unreal Engine have showcased prototypes of this feature, which are in experimental states. In an optimal VR editor, developers from a variety of fields, 3D designers, sound artists etc., can collaborate on the same project without having to grasp the complex skill of game programming. Consequently, the SME can also directly contribute on the content creation process providing his expertise,

⁸ <https://unity.com/editorxr>

⁹ <https://docs.unrealengine.com/en-US/Engine/Editor/VR/WelcomePDF/index.html>

in a straightforward manner. This collaboration, can speed up considerably the VR creation process, since all the experts can immediately correct flaws, without the tedious process of explaining them to the developers. Apart from the advantages of such a tool in collaboration across a variety of expertise, by enabling the user to author the behavioral tasks, the assistant assets (UIs, holograms, etc.) and the evaluation method from the same environment they will be eventually displayed, a coherent overview of the simulation is created, which reduces the complexity of the content procedure while also providing effortless maintenance, compared to editing through a medium that displays the outcome in different visualizations.

2.4 Supervised Machine Learning for Evaluation

In cognitive training, evaluation usually takes place in the form of multiple choice quizzes; these methods are particularly easy to assess since they restrict the possible answers the trainee can use. But how do we evaluate a scenario where freedom in the completion of the task is mandatory for learning? There is high interest in automatically predicting performance by exploiting the effectiveness of supervised machine learning in multimodal simulations or environments that the user can perform complex tasks in multiple correct ways [11], [37]. Even though the effectiveness and integrity of such systems used for automatic evaluation over an experienced human teacher is highly debatable [3], automation is particularly useful for provisioning adaptive support and immediate feedback to learners, while also reducing the cost.



Figure 6. Usage of the FOCUS system, where recommendations are presented on top of the learning material (book).

Apart from evaluating, by analyzing and learning from such data, intelligent agents can augment the training experience of such applications through assisting and guiding the trainee towards completion. Personalization of learning has been a hot-topic in academia for some years now, by meeting the learning pace and the optimal instructional approach the trainee needs, learning activities become relevant to learners and drive their interests. There is research both on how to automatically personalize content using pre-defined rules [8] but also how personalization can be achieved through novel machine learning algorithms

[23]. Multimodal machine learning solutions have not only been tested in training environments but also for evaluating the learning process through methods such as neuro-feedback. FOCUS is a personalized BCI learning assistant which improves the reading capabilities of the learner by identifying “focused” time windows for starting the reading process, while also enhancing the learner process through virtual social rewards such as, praise (text/audio) [18]. In figure 6, the individual components of the FOCUS system are presented as well as the user interacting with it.

2.5 Our publications and previous work related to this thesis

In this section past projects and publications will be presented and their relation to this work will be described.

▪ M.A.G.E.S. 3.0: Tying the knot of medical VR [28]

M.A.G.E.S. is a novel Virtual Reality authoring SDK scoping in accelerating the creation process of surgical training and assessment virtual scenarios. It is built on top of Unity 3D game engine and it provides the following features in an easy manner for developers:

- Multiplayer: collaborative networking layer that utilizes Geometric Algebra interpolation for bandwidth optimization.
- Assessment: real-time performer assessment both with supervised machine learning and predefined rule-based analytics.
- Deformations: GA-enabled deformable cutting and tearing, as well as configurable soft body simulations.
- Curriculum: Tools for defining an educational curriculum enriched with visual guidance, gamified elements and objectives to enhance transfer of knowledge and skills.
- Prototyped surgical techniques: Implementation of commonly used surgical techniques that can be customized in order to populate new content in a rapid manner.



Figure 7. In the left image the collaboration capabilities of M.A.G.E.S. are showcased in a Resuscitative Endovascular Balloon Occlusion of the Aorta VR training scenario. On the right image the nasal swab test conducted for testing people with possible COVID-19 is depicted.

▪ A True AR Authoring Tool for Interactive Virtual Museums [10]

In this publication we utilized the innovative method of spatial computing, True Augmented Reality, for cultural heritage preservation. True AR promotes high realism visualization of 3D objects that at first glance are not easily distinguishable from real objects. By exploiting True AR 3D models of various exhibits can be reconstructed and presented in a realistic and innovative way. A playground demo highlighting the components and tools for creating True AR interactive Virtual Museum applications is also presented.

Even though in this work we did not address the problem of assessment in the virtual environment, by employing similar implementations for interacting with the objects both in VR and AR, we are planning to migrate our work concerning performance assessment to the AR field.



Figure 9. The overview of the application which focuses on the restoration of Knossos (right hologram) or Sponza (left hologram).

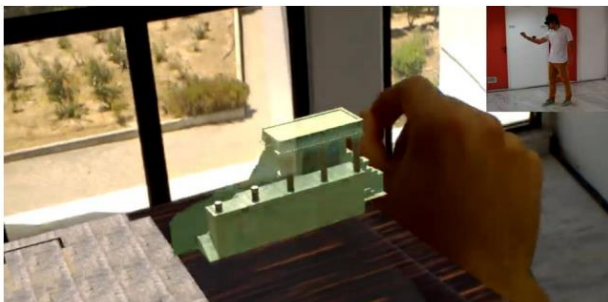


Figure 8. Interaction using the pinch gesture; the user can move a 3D object around the environment.

- **Virtual Reality Rehabilitation based on Neurologic Music Therapy: A qualitative preliminary clinical study [2]**

In this work we developed a virtual reality environment combining Presence Positive Technologies for Well-being, Neurologic Music therapy techniques and hand tracking. Our main objective was to provide a training platform capable of sensory stimulus for motor control retaining to survivors of neurological disorders. In the virtual environment a 3D avatar was added which guided the patients through the multiple hand exercises. Spatial computing rules were defined in order to evaluate the patient's performance which was used as feedback providing motivational enhancement and social reward.



Figure 10. (Left) The instructor avatar we created for guiding the patients through the hand motor exercises. (Right) A patient performing the exercises in the virtual environment.

3 Analytics System for VR training scenarios

Performance assessment is a vital part of the learning process, since it provides valuable feedback to the trainee, guiding him towards improvement. In order for this feedback to be fruitful, an accurate and well explained assessment method is needed. Apart from the final reported score, explanation on its computation is also essential. Furthermore, we opted for a method that is reusable in different kind of scenarios, an assessment method that was addressing the VR interactions of the user with the environment, for instance placing an object at a specific location. Finally, due to the rapid growth of the VR field, a systematic approach that can later be expanded and modified with ease was required.

A significant advantage that virtual training scenarios hold over non-virtual training scenarios is the convincing replay-ability options that they offer. The trainee is able to re-try again and again with no added costs, learning from his mistakes, while reproducing such scenarios in the physical lab, most of the times, is extremely costly. Immediate assessment and feedback of the user's session is vital for this cycle of continuous improvement. Enabling the learner to immediately correct his mistakes, instead of waiting feedback from his supervisor is a great feature of this project.

Breaking down the training scenario to clear, sequential, independent steps was a requisite part of our project. Apart from the added value in the training scenario, where the user is able to build a mental map of each task, both for the developer and the trainee. By utilizing a step-by-step structure, the development process becomes straightforward, rapid and the resulting simulation is scalable, since new independent steps can be added, and easily modified. Additionally, it enables the trainee to focus on the steps that he performed the poorest by redoing only those specific tasks. For the above reasons, we decided that our analytics system should be implemented focusing on augmenting a curriculum structure. Our design was heavily influenced by the flexible Scenegraph data structure which is a core module in the M.A.G.E.S. SDK [28].

3.1 Overview of the analytics system

In this section we will describe the overall pipeline of the analytics assessment platform and in the following sections we will explain each component in detail.

After programming the behavioral rules of each task, the need for developing how to assess them arises. To achieve this, the developer has to define which VR elements and interactions contribute to accurately grading each task. These settings are stored in JSON files and are loaded at the start of each session. We preferred JSON files for the ease of maintenance that they provide. However, creating and editing multiple JSON files, one for each task, is time consuming and prone to errors. These were the

reasons that pushed us in developing an authoring tool, explained in section 3.3, that speeds up the process of defining and producing these files.

After specifying the assessment elements and storing them in files a component to handle them was required. For this reason, an assessment manager was implemented which is responsible for loading the predefined scoring settings from the JSON files and storing the results of each task. Additionally, this manager is in charge of constructing, destroying and initializing the components which are responsible for scoring the trainee, the scoring factors. Finally, at the end of each scenario it calculates the user's final score from all the individual scores of each task. The overview of this workflow is depicted in figure 11.

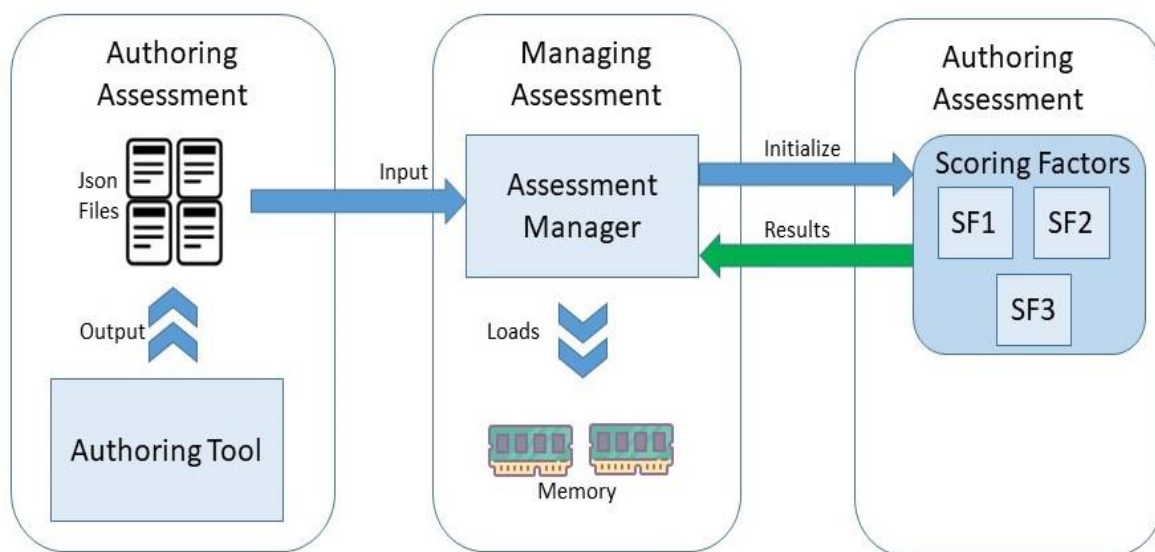


Figure 11. Workflow pipeline overview of our analytics platform

The final part of this overview is how we calculate the score of each task. As mentioned before in this chapter, our goal was to create a generic assessment solution for various VR training scenarios, thus a modular and easily scalable approach was needed. To address this issue we introduce the scoring factors components, explained in section 3.2. Each task contains a list of scoring factors and each type of scoring factor is responsible for scoring a certain behavior of the user. A task can contain multiple instances of the same type of scoring factor, each referring to different objects in the environment or having different importance.

3.2 Assessment

Each task the user is required to perform in order to successfully complete the scenario is accompanied by its aggregate data. These data are:

- **Score:** Final score that the user achieved in this task, calculated by the scoring factors. This score is always in the range of [0, 100].
- **Time:** Time it took the user to complete the task.
- **Error messages:** List of error messages or remarks.
- **Weight:** How important/challenging is this task compared to the rest.

At the end of the training scenario the trainee's total score is calculated as the mean of all individual tasks scores. For calculating the final score, the weighted arithmetic mean was used:

$$score_{final} = \frac{\sum_{i=1}^n w_i * S_i}{\sum_{i=1}^n w_i}$$

where n equals the count of all tasks, w(i) the weight of task i and s(i) the achieved score at task i.

Algorithm 1 Task Performance Calculation

OnApplicationStart

- 1 Create Directory D
- 2 Create Directory Results
- 3 For each assessment file:F
 - 3.1 Open F
 - 3.2 Add task_name in D as key
 - 3.3 For each scoring factor:SF in F
 - 3.3.1 Create new SF
 - 3.3.1 Set parameters of SF from F
 - 3.3.2 Add SF to D(task_name)

OnTaskInitialize

- 1 Wait for task initialization
- 2 For each SF in D(task_name)
 - 2.1 Add SF to Unity Scene as component
 - 2.2 Initialize SF

OnTaskFinish

- 1 For each SF in D(task_name)
 - 2.1 S_tmp: SF.Perform()
 - 2.2 Score: Score + (S_tmp * SF.factor)
 - 3 Score: $\frac{S}{counter(D(task_name))}$
 - 4 Add to Results {task_name , Score}
-

Figure 12. Algorithm of performance score calculation for each task.

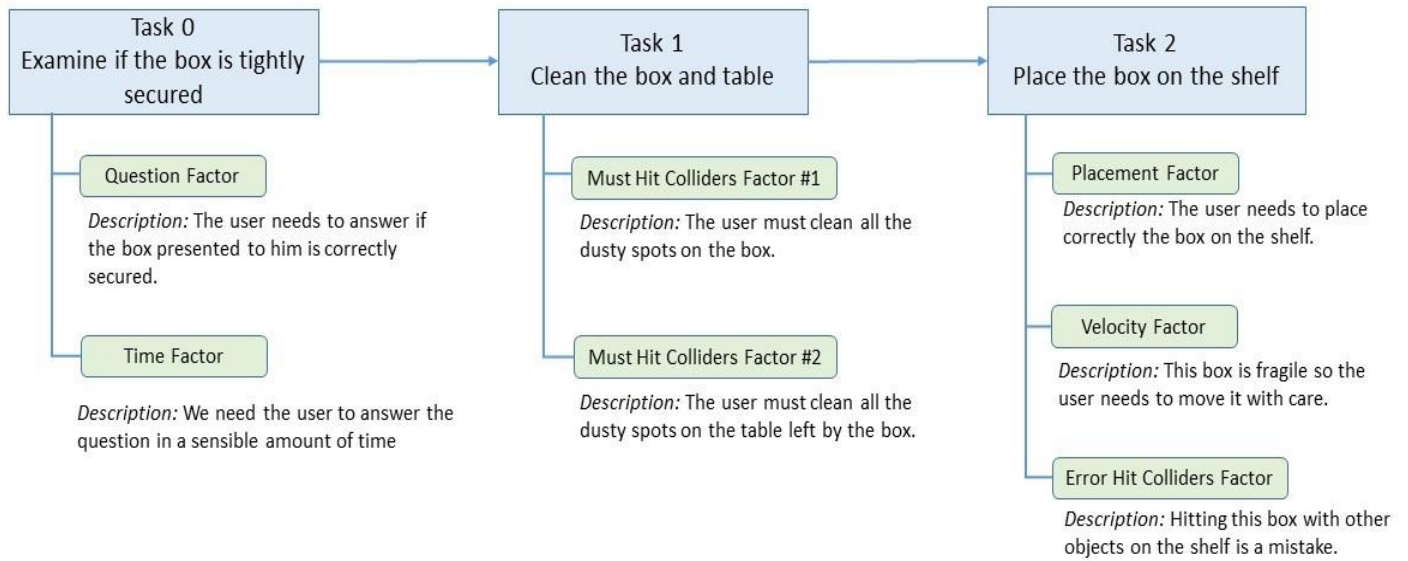


Figure 13. Scoring factor usage example. In this diagram we describe how multiple scoring factors can be combined to achieve accurate evolution of user performance in each task.

3.2.1 Assessment Manager

A training scenario is composed by sequential tasks. For supervising each individual task, we concluded that the implementation of a manager was mandatory. In detail, the assessment manager is responsible for storing and tracking the user's performance at run time while also offering the option to retry each task. For these reasons, we concluded that the implementation of a manager was necessary.

At the start of each session the assessment manager loads all the JSON files and stores their contents in memory creating instances for all scoring factors that are defined. Additionally, it also loads all resources assets that are referenced by these scoring factors in order to avoid the overhead of loading them at the start of each task. For utilizing the rich functionalities provided by the Unity Engine these scoring factors need to be added as components in the virtual scene, which also takes place at the start of each session. Finally, at the start of each task he is responsible of calls the initializing its scoring factor and at the end finalize them, clearing the virtual environment of any unnecessary assets, spawned by the scoring factors, and preparing it for the next task. Furthermore, he gets the results of each scoring factor, which consists of remark messages and the score the user achieved in that specific aspect of the task. From the multiple scoring factors in each task, multiple scores are produced, thus the assessment manager is in charge of computing the final score. For calculating the final aggregated score, the weighted arithmetic mean statistic was used, applying the importance field of each scoring factor as the weight.

As mentioned at the start of this thesis, the option of retrying holds a principal role in training and learning. For this reason, we chose to also implement the option of resetting a specific task. This provides the trainee the promptness to apply the feedback he got from his previous performance, immediately, without the hassle of closing and reopening the application. In order to implement this, the assessment manager calls the undo function of all current scoring factors and initializes the preceding ones.

Finally, when the simulation has ended the manager sends the trainee's analytics to the analytics exporter, which is tasked with saving them into tsv files for post-session review. In total, 5 types of files are created at the end of each session.

- **TotalData:** In this file the sum of all analytics from each individual task are stored. Specifically, these data are *total time of session*, *final score* and *total number of errors*
- **Time:** In this file the completion time (seconds) is stored for each task.
- **Score:** This file contains the achieved score for each task.
- **Errors:** In this file the total amount of errors and remarks performed in each task is written.
- **ScoringFactorsFile:** For each task one scoring factor file is created, named after the task description name. These files contain all the data describing the multiple scoring factors that were set for each task.

3.2.2 Scoring Factors Design

Up to this point the scoring factor component has been mentioned a few times. The main purpose of this component is to accurately assess a specific aspect attribute or interaction event from the virtual environment. They were designed around four principal goals, maintainability, reusability, modularity and accuracy. By breaking down the assessment process of a complex task to simple components, each responsible for a specific part of the assessment, we achieved modularity but also maintainability, since a methodical design enables the developer to change/add/remove a specific factor without altering any other part of this project. Additionally, since each type of scoring factor is addressing a generic aspect of the user's interaction with the VR environment, they are reusable in different fields of training scenarios. Finally, by evaluating the accuracy of each scoring factor we can procure an evaluation for the whole system, which, is a less complex procedure than evaluating the accuracy of the simulation.

Our platform provides the developers with an interface which needs to be implemented to link the scoring factor with the rest of the components. By introducing this interface, an abstraction is achieved, which enables the expansion of our system while also rendering the process of creating custom scoring

factors, contextualized to the training material, is straightforward. Finally, once this interface is understood and the usage of each function is grasped, the process of adding a scoring factor to the rest of this project can be replicated in a short amount of time. The `IScoringFactor` interface requires the developer to implement four functions.

- *Initialize*: This method is responsible for spawning all necessary assets for this scoring factor in the environment and apply the settings stored in the JSON files.
- *Perform*: This method cleans the environment of any assets that have been spawned by this scoring factor, while also computing the resulting score based on their settings.
- *Undo*: This method behaves similarly as the perform function but instead of calculating score it resets all variables to their default values.
- *GetResults*: This method returns the assessment data generated from this scoring factor in the form of an `SFData` struct (described in the appendix section).

Every scoring factor generates a common data structure where it stores its results in an explainable format. This data structure is used for exporting the scoring factors to the output files, while also providing feedback to the user, by presenting him his individual mistakes. We chose to express all scoring factors in the same type of structure, in order to automate the process of describing them.

Furthermore, we developed the `ScoringFactor` abstract base class, which is responsible for linking our Interface with the Unity Game Engine. This is achieved by extending the `MonoBehaviour` and adding each scoring factor as a different component in the virtual scene. This enables the developer to fully focus on developing the assessment mechanic and not its integration with the Unity Engine.

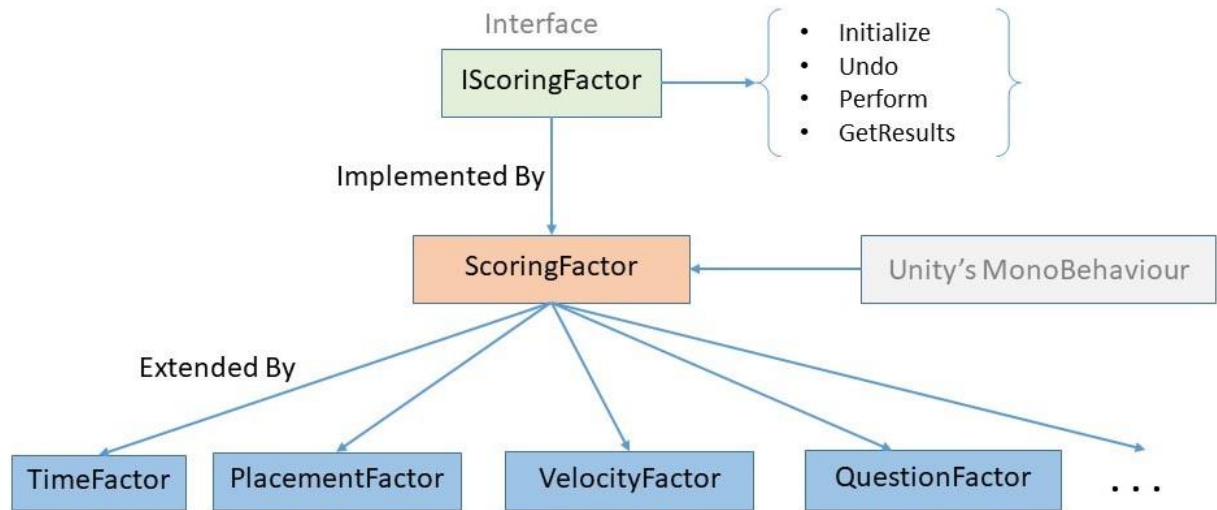


Figure 14. Architecture diagram of Scoring Factors Components

Finally, for consistency between different scenarios and convenience for the trainee we decided to that the final score needs to be mapped to the range of $[0, 100]$. This rule is enforced by the assessment manager when calculating the aggregate score of each task, but for better clarity towards the user it is advised that the output score of each factor is already mapped in that range.

3.2.3 Scoring Factors Prototypes

In the previous section the design and the method of implementing scoring factors was explained. For the completion and testing of this platform several prototypes of scoring factors were implemented. In table 1 their usage is explained.

The implementation of the **time** and **question** scoring factors proved to be straightforward. For the time scoring factor a script counting the elapsed time since the start of the task was developed and at the end of the task its value is compared with the final one. Continuing with the question scoring factor a Unity UI prefab, which can be seen in figure 15, was constructed serving as a template where the developer can optionally set images and descriptive text for each answer, while also specifying which is the correct answer.

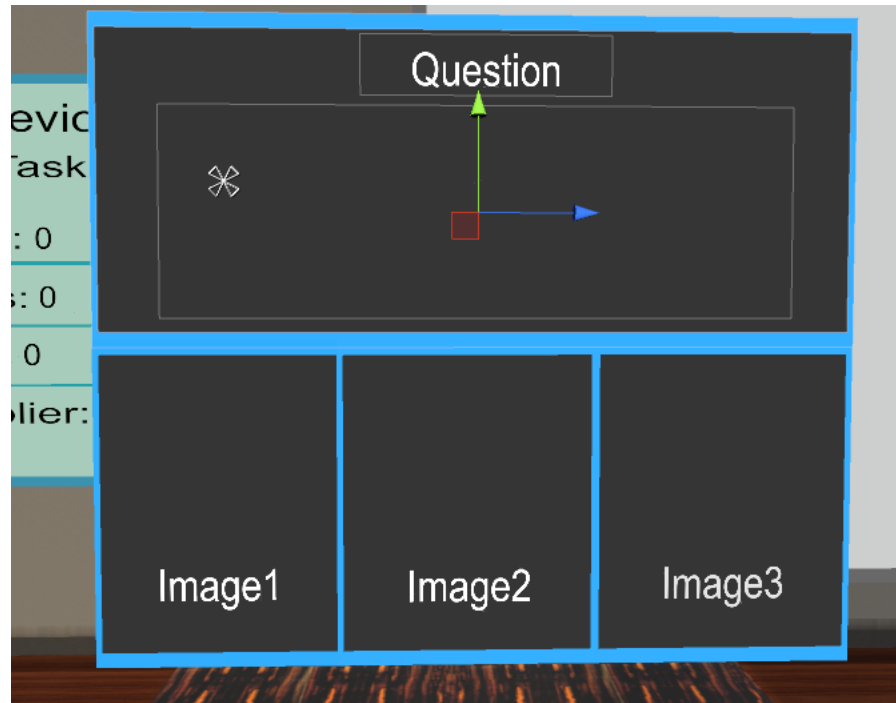


Figure 15. UI template for the question scoring factor.

The **placement** scoring factor proved to be the most challenging. Our goal was to assess the rotational placement of an object. Since the correct position is usually defined as the means for the completion of the task, we concluded that only the assessment of the orientation was contributive. For providing the perfect rotation the user needs to save it in a prefab through the unity editor, which we are calling the finalPrefab. This prefab is spawned in the scene at the initialization of the scoring factor and when it overlaps with the interactable object the rotational difference is calculated. The first obstacle that we encountered was the, always present, small offset from the perfect angle, due to being close to impossible to place an object with machine level accuracy. This led us to adding a down limit angle difference parameter that marks the start of the assessment computation. This also enables the developer to accurately assess tasks where the placement of an object is not fully restricted to a correct rotation. In addition, defining the upper limit of the angle difference was necessary, in order to be able to accurately define the angle range that was scored. As a result, this scoring factor computes the range [down_limit, upper_limit] which is mapped to the range [100 , 0] outputting the user's score. Finally, due to many objects providing symmetries in specific axis, the reset angle parameter was added. This parameter maps the user's final rotational input to the range [0, reset_angle].

For the implementation of the **avoid objects** scoring factor, the developers need to define the position of colliders in the 3D environment through the Unity Editor and save them as prefabs, while also specifying the interactable object(s) that needs to avoid them. These colliders are spawned at the

initialization stage of the corresponding task. Each collision reduces the user's score in this action based on the defined importance of this scoring factor. Finally, due to the behavior of Unity's collision system, multiple events get triggered while the two objects continuously collide. This results to the collision error getting logged multiple times, inaccurately. For this reason, we implemented a reset timer functionality, where for a couple of seconds this scoring factor stops checking for collisions.

On the other hand, the **hit objects** scoring factor behaves in the opposite way. The developer needs to define colliders in the 3D environment where the user needs to hit with an interactable object in order to achieve maximum score. The number of colliders that need to be triggered is again mapped to the [0, 100] range.

The **enforce interaction** scoring factor requires two interactable items to be referenced with the corresponding hand. The main interactable object is defined as the one the user needs to use in order to complete the task while the secondary is the one he needs to interact in order to complete it correctly. If the user interacts with the main interactable without interacting with the secondary object he loses points depending on the importance of the scoring factor. In order to punish continuous false activity, a timer functionality was implemented punishing the user every few seconds. Since these interactable objects are part of the rules for the completion of the task they are not spawned by our platform but are expected to be spawned from another module.

Finally, the **velocity** factor is used for assessing the movement speed of an object. In many occasions in VR training scenarios, objects need to be moved with care, due to being fragile or especially important. For this scoring factor the maximum velocity that this object can reach is set. At each frame the velocity of the object is calculated and if it surpasses the maximum allowed velocity it results to zero score.

Scoring Factor	Description
Time	Used in tasks where completion time is important. The user has X seconds to complete it and for each extra second that passes he has 10% penalty.
Placement	Used in tasks where the user needs to place an object at a position. An angle difference range is defined [down_limit, upper_limit] which is then mapped to [100, 0] score.
Avoid Objects	Used in tasks where the user needs to avoid touching an object with an interactable. Error colliders around the object need to be set and also the interactable item needs to be set as parameter.

Hit Objects	Used in tasks where the user needs to touch an object with an interactable. Similarly, with the above, the colliders that must be hit need to be set and also the interactable item.
Enforce Interaction	The trainee needs to interact with an object while interacting with another object. The two interactable items need to be set.
Question	A specific type of UI needs to be defined which is spawned at the initialization of the task. The user is evaluated based on his answer.
Velocity	The user needs to move an interactable object with care. The velocity of that interactable item is monitored. In case it surpasses the maximum velocity it results in zero score. The interactable and the maximum velocity need to be set.

Table 1. Description of the potential usage of each scoring factor, as well as their needed parameters.

Even though these are only seven scoring factors they cover a great deal of assessments events in a task. Due to targeting principle interactions the user has with the environment, the developer can be creative about their usage for a more accurate training scenario; e.g. for the task of “Hammering a nail on the wall” the <must stay collider> scoring factor could be used to evaluate if the user is holding the nail with his one hand while hitting the nail with the hammer and avoid the implementation of a scoring method specifically for this interaction.

3.3 Setting up Scoring Factors

Up to this point, we explained the overview of the system and the assessment method, but even though these scoring factors have been implemented the developer’s preferences need to be set and saved. To achieve this our authoring tool can be utilized.

At the start of this project we were manually creating and filling the JSON files that hold the scoring factors settings. We quickly noticed that as the complexity of the task grew so did the time it took to write the corresponding JSON file. In order to speed up this process and also provide easier maintenance for the developers we decided to create an authoring tool. In this section the features of this tool are explained as well as the structure of the JSON files.

3.3.1 JSON Files

Each JSON file describes all scoring factors settings for a task. Figure 11 illustrates an example of a JSON file containing all the parameters needed for the assessment of a task. Below there is an explanation of some important fields.

- **Multiplier:** The weight of this task's score. This is used by the assessment manager to calculate the final score of a session.
- **enforceInteractionData:** This field contains the settings of the respective, Enforce Interaction scoring factor.
- **ErrorMsg:** In case of performing poorly the error message that will be shown to the user.
- **Importance:** The weight of the specific scoring factor. This is used by the assessment manager to calculate the weighted score of each task. Predefined-weights are mapped to

We decided that in case the developer removed a scoring factor, it should be marked as disabled and not deleted from file, this way in case the developer re-enables it, the scoring factor will already have the previous settings saved. However, this small feature combined with the capability for constant expansion of scoring factors led to large and hard to read JSON files, since it required to include all the disabled settings as well. Thus, we resolved in creating an authoring tool for VR training assessment.

```
"multiplier": 1,
"errorsData": [],
"errorsStayData": [
  {
    "errorMsg": "You must hold the box while cleaning the stains on it.",
    "remainingActions": 0,
    "errorToolsColliders": [
      "Assets/Resources/Actions/Action1/InteractCollider.prefab"
    ],
    "errorGO": [
      "Assets/Resources/Actions/Action1/Cardboard_boxDirty.prefab"
    ],
    "collidersParents": [
      "Assets/Resources/Actions/Action1/LeftHand.prefab"
    ],
    "startInteractionGameObjects": [
      "Assets/Resources/Actions/Action1/Cloth.prefab"
    ],
    "errorType": 0,
    "showErrorUI": true,
    "importance": 2
  }
],
"errorsHitPerformData": [
  {
    "errorMsg": "You missed a lot of spots that needed cleaning.",
    "remainingActions": 0,
    "errorToolsColliders": [
      "Assets/Resources/Actions/Action1/Analytics/MustHitCollider.prefab",
```

```

    "Assets/Resources/Actions/Action1/Analytics/MustHitCollider2.prefab",
    "Assets/Resources/Actions/Action1/Analytics/MustHitCollider3.prefab",
    "Assets/Resources/Actions/Action1/Analytics/MustHitCollider4.prefab",
    "Assets/Resources/Actions/Action1/Analytics/MustHitCollider5.prefab",
    "Assets/Resources/Actions/Action1/Analytics/MustHitCollider6.prefab"
  ],
  "errorGO": "Assets/Resources/Actions/Action1/Cloth.prefab",
  "collidersParents": [
    "Assets/Resources/Actions/Action1/Cardboard_boxDirty.prefab",
  ],

```

Figure 16. JSON example analytics file that stores all user defined parameters.

3.3.1 Assessment Authoring Tool

Since, this project was developed on top of the Unity Game Engine developing our authoring tool by expanding the Unity Editor seemed straightforward. As a result, our authoring tool provides multiple functionalities such as referencing unity prefabs effortlessly by drag and drop, enforcing type checking for Unity's built-in classes and forthright development capabilities since it is in the same framework with the rest of the project. Additionally, by visualizing the settings of each scoring factor in an all-inclusive, easy to read panel, an overview of the assessment method is provided to the developer, rendering the creation and maintenance of this task an effortless process.

In figure 15, the panel of an example task is presented. On the left side the default template is shown, presenting the available scoring factors. The ones selected fill their needed parameters indented, which the developer must set. These parameters are identical to the ones set in the JSON files, presented in a more user friendly form. A great feature of our tool is the support for referencing unity assets directly from the editor. This allows us to specify the type of objects that will be referenced to each parameter eliminating developer errors, while also allowing the user to quickly set the assets through drag and drop and access it through double click.

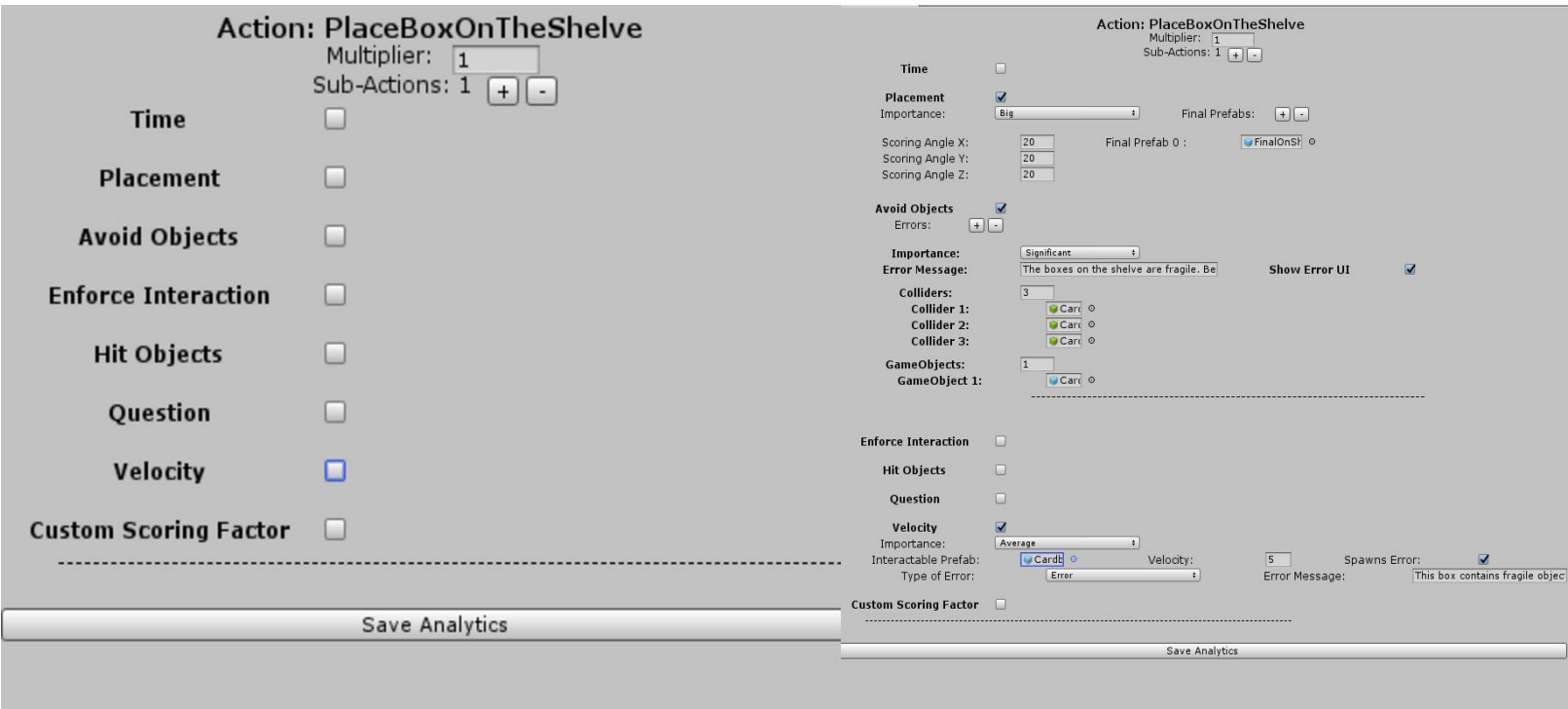


Figure 17. On the left the uninitialized analytics editor panel is shown of a specific task. On the right the filled panel is presented with the developer's set parameters and the references to the needed assets.

4 Supervised Learning based VR assessment

In chapter 3, we presented how we simplified the process of authoring the assessment method of a VR training scenario to simple pre-defined steps that a programmer can master in a short amount of time. However, by assigning the execution of this process to the developer, errors regarding the accuracy and integrity of the system may occur, due to the need for communication between the SME and the developer. Driven by the opinion that our evaluation platform has a steep learning curve for non-programmers, since using the editor of a game engine (Unity) can be a complex procedure especially for someone not familiar with computer science terminology, we opted for a solution where the SME would be able to contribute to the VR content creation process from inside the virtual environment. Our first idea was to incorporate our analytics platform in a VR environment by utilizing 3D user interfaces, but we rejected this idea in the designing phase due to two flaws that we identified: a) our assessment platform was built around the principle of continuous expandability to meet the needs of the current scenario, thus the 3D VR interface would need constant upgrade to keep up with the rest of the platform, b) there would still be a learning curve, although shorter, for the SMEs since they would have to become accustomed to our UI and possibly to VR terminology.

In addition, in order to take advantage of the embodiment capabilities [22], [33] that virtual reality provides we needed an authoring tool that enables the expert to author while performing the behavioral physical movement he is accustomed from the real world. Such a solution would also rapidly speed up the content creation process, while making it slightly cheaper since one person can fulfill two roles. Finally, as VR technology advances the expert trainers become more demanding of simulations close to reality. This demand has produced a need for, among others, constant growth of new, more complex VR interactions methods and psychomotor tasks that can be completed in multiple correct ways [20]. These two requirements force the creation of new scoring factors that can catch up to expectations of providing accurate assessment but also the increased complexity of having multiple correct answers. We concluded that a solution employing supervised machine learning algorithms could potentially solve all these problems while also providing a sensible method for scaling along the other components of VR training.

In this chapter's following sections our work on how we employed supervised machine learning to assess the user's placement of an object in VR will be explained. Moreover, a novel VR session logging algorithm and its purpose will be presented and described.

4.1 Machine Learning for VR assessment

At the beginning of designing this module, we had to research whether the available machine learning algorithms were sufficient or if a new algorithm was needed. We concluded that we could achieve our goal by employing a methodology derived from the computer vision field, in which supervised ML has robust and tested solutions such as Convolutional Neural Networks (CNN). In order to avoid reinventing the wheel we chose to use an existing ML library instead of creating our own version. Since, our project was built on top of Unity Game Engine, we needed an ML framework that could be integrated with the engine. The available options were few, unfortunately, the well-integrated Unity ML Agents does not provide functionalities for supervised ML, especially for image-based algorithms. Thus, we turned our attention to Google's TensorFlow and its integration with Unity through the TensorFlowSharp (TFSharp) plugin.

The pipeline for creating a Deep Learning agent capable of VR assessment is depicted in figure 18. In our implementation, the developer or SME generates data samples from the VR authoring unity application which are saved as images. Then we run a python script which loads these images and utilizes the ML capabilities of TensorFlow and Keras to train and export a CNN model. Finally, we load the model in Unity and when the user performs the placement task we capture new images, passing them as input in the CNN model which predicts the assessment result.

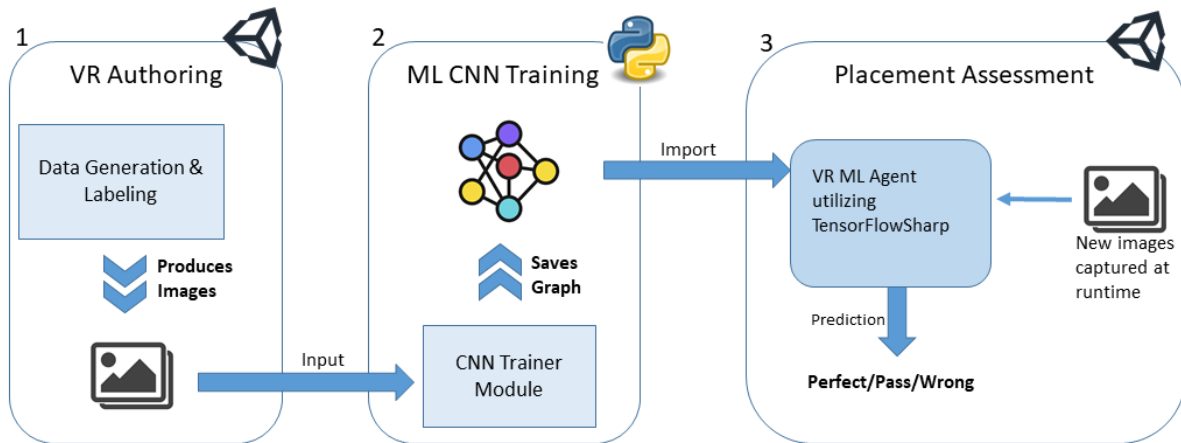


Figure 18. Diagram describing our VR ML training architecture.

4.1.1 Data Collection

The first step of every ML solution is the accumulation of data. Since, we decided to examine the problem by employing solutions derived from the computer vision field, our data would be 2D images of the VR environment. For assessing the placement of an object we did not need to capture the events that occurred in the whole environment just at specific areas of interest where the object should be placed. A unity prefab was created which was composed of 4 virtual cameras placed in specific spots in order to capture the SME's actions from different angles. In addition, an event box was added which triggered the capturing of images from all cameras, once the expert finished interacting with the object. For each area of interest, a different CNN model is trained by using the recorded image data of that specific area.

A common problem of computer vision applications is the occlusion of the objects of interest and their separation from the background. In our case, since the environment is virtual, we can meddle with what each camera records by utilizing layer masks and defining a solid color background to all cameras. Each area of interest is accompanied by a layer tag and the cameras around it only record 3D objects tagged with that specific layer.

After capturing the raw data, a way for the SME to classify them was needed. We implemented a VR environment where the SME was able to freely interact with the objects around him and at the end of each interaction he was presented with the option through a spatial UI of classifying the generated data with “perfect, pass, wrong”. Restarting the interaction with the object discards the generated data and destroys the spawned UI. In order to easily load these data on the TensorFlow module, each area of interest stores the recorded images in a separate folder, which is further split based on the classifiers (in our case “perfect, pass, wrong”).

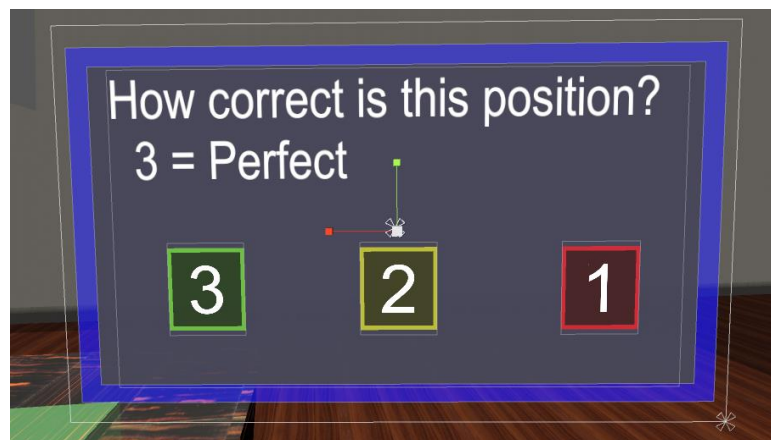


Figure 19. The UI used for classifying each generated position.

4.1.2 Training CNN models

As mentioned at the start of this chapter we chose to use Tensorflow and Keras for training our ML model, which pointed us to python. The model creation process is split into four parts, a) load dataset, b) build model, c) train model, d) export model to a file format supported by Unity.

In order to train our model the images recorded from unity need to be associated with their labels in a format that is supported from Keras. We chose the most common method of creating two data structures X, Y where X is a 4D array storing the information of images with the following shape [number_of_images, imageX_size, imageY_size, 3] and Y is a 1D array that holds the labels of each image and its size is equal to the number of images. This process is repeated for each camera. For loading and processing images the openCV python package was used. In addition, in order to avoid memory overflow during training images were downsized to 64x64 resolution; in case an input image has lower size it is discarded.

For building our models two methods were examined, *multiple CNNs* and *multi view*. In the former, we exploited the layer capabilities provided by the Keras library by building a sequential model. The model's structure is depicted in figure 18. We are using three convolution layers each accompanied by its respective pooling layer. Continuing, we are passing our input through a flatten layer, a dense layer with relu activation, a dropout layer and finally a dense layer with softmax activation. Our choice of layers and their parameters was heavily influenced by [12] and from our post-evaluation. By calculating and plotting the accuracy and loss at each epoch we concluded that our models stop learning at ~45 epochs.

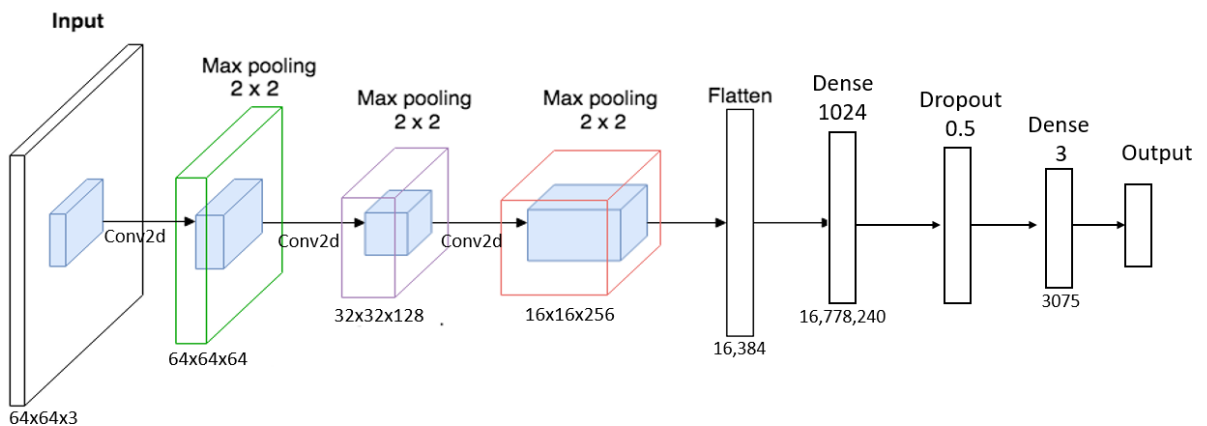


Figure 20. Our sequential convolutional model structure with the output dimensions on each layer.

For training our models the stochastic gradient descent (SGD) method was used. Since assessing the correct placement of the images is formed as a classification problem, the loss function used is the cross-entropy one. Cross-entropy is the most common loss function in artificial neural network settings. It represents the difference between the true and the predicted labels of the data. The neural network's job is to minimize that difference, and this is achieved by a stochastic gradient descent method. Additionally, the labels are provided in a one-hot representation, and therefore the categorical version of the function is used. As stated above, the loss function is minimized by a stochastic gradient descent method. The method used in this setting is the Adadelta optimization, because it adapts the learning rate by using a moving window of gradient updates, instead of accumulating all past gradients. This is crucial since as the training is close to its end the gradients become smaller and the big gradients at the beginning of the session should not be taken into consideration. By adapting the learning rate this way, the neural network manages to converge to the local minimum of the loss function, instead of bouncing up and down optimal value.

Our training dataset consists of 88 images from each 3D camera. Even though we could generate more samples through our VR Editor application, our goal was to create a rapid process that the developer or subject matter expert could easily replicate. Due to the small datasets we chose a small batch size for training as well. While testing different training configurations we quickly noticed that producing a CNN model with high accuracy was heavily influence by the random seed parameter of tensorflow. Even though, trying different seeds until a satisfactory model is produced is possible, since we had four models this process could take a large amount of time. Thus we turned our attention to the multi-view methodology in order to produce a single CNN capable of assessing all four images.

For our multi-view model Keras Functional API was utilized. The procedure for building the four CNN models is similar with the *multiple CNNs* methodology with the exception that a label needs to be defined in order to access them, when fitting our training dataset. The main difference is that before adding the flatten layer we concatenate them into one model. Its structure is shown in figure X. Even though now we only need to tune our parameters around one agent's accuracy, our DL learning capabilities are still dependent to the random seed of Tensorflow.

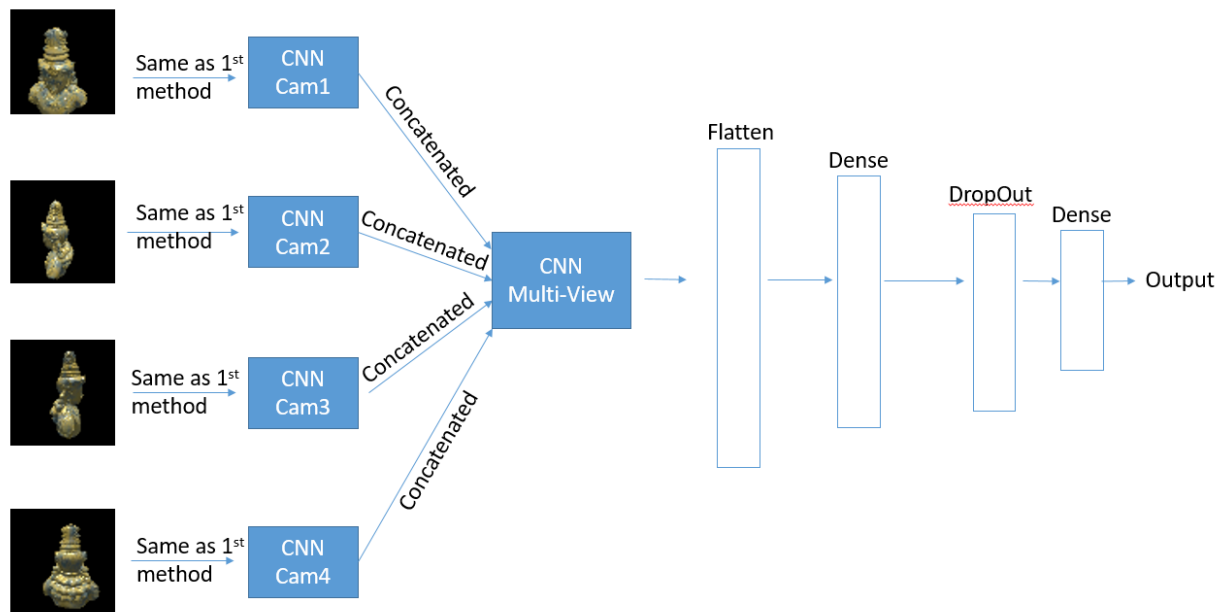


Figure 21. Structure of multi-view model. The four CNNs are created similarly as shown in figure 17.

How our models learn with different configurations and optimization algorithms is described in chapter 6. Finally, our python script exports the trained model. Unity’s plugin TensorFlowSharp is compatible only with specific binary files which was the reason we needed to convert our saved graph of “.chkp” format to “.bytes”. For this conversion the Tensorflow freeze-graph library was used.

4.1.3 Predicting at runtime

As mentioned before the Unity package TensorFlowSharp was used for importing the saved graph in Unity. Importing a model is straightforward and the detailed code can be found in the appendix, two important details need to be taken into account: a) the first(input) and last(output) nodes of the graph need to be defined, b) the input images, that our model will assess, needs to be the same size as the images used in training.

For capturing images during an assessment session, we are using the same Unity prefabs used in authoring containing the areas of interest and the cameras. Once, the user ends interacting with the 3D object all cameras capture an image and input them to their respective CNN model in case the *multiple-cnn* method is selected or to the *multi-view* CNN model, which predicts the result. Since this process can be

computational heavy, Unity's coroutines are utilized to avoid a drop in rendered frames per second, although this leads to a small delay (approximately one second) in the computation of the trainee's results, it was regarded as trivial compared to an occasional "freeze" that could destroy the user's immersion. Finally, each graph outputs their prediction class "perfect", "pass" "good". In the case of multiple CNNs the output labels are mapped to 2,1,0 accordingly. The final assessment of the user is then computed as the average of the four outputs.

Algorithm 3 Predicting at Runtime

OnApplicationStart

- 1 Load all DL files
- 2 Create list of Graphs: L_g
- 3 Create list of Cameras L_c
- 4 foreach DL saved file
 - 4.1 Create new Graph: G
 - 4.2 Import DL to G
 - 4.3 Insert G to L_g
 - 4.4 Insert Camera: C to L_c

OnEndInteraction (Interactable: I)

- 1 mean score: $M_s = 0$
 - 2 if I is colliding with AOI
 - 3 Start Thread:
 - 3.1 foreach C in L_c
 - 3.1.1 Capture ScreenShot Texture: S_t
 - 3.1.2 Convert S_t to array image: A_m
 - 3.1.3 Pass A_m as input on G
 - 3.1.4 Get output of G, list of confidence on all labels L_l
 - 3.1.5 Max_l : label of max confidence in L_l
 - 3.1.6 $M_s : M_s + Max_l$
 - 3.2 $M_s : \frac{M_s}{cameras}$
 - 3.3 show label corresponding to M_s to user
-

Figure 22. Algorithm of prediction at runtime

Even though this solution worked great in tasks where the required data could be captured in one frame, for example placing an object, it does not cover tasks that entail continuous movement or added parameters such as force and eye movement. In addition, the lack of high amount of samples for training and low resolution of images, forced us to search for data of higher accuracy. Instead of continuing using methodologies derived from the computer vision field we believed that by training our models with data logged directly from the virtual environment a general and automated process for training ML with small amounts of datasets is feasible.

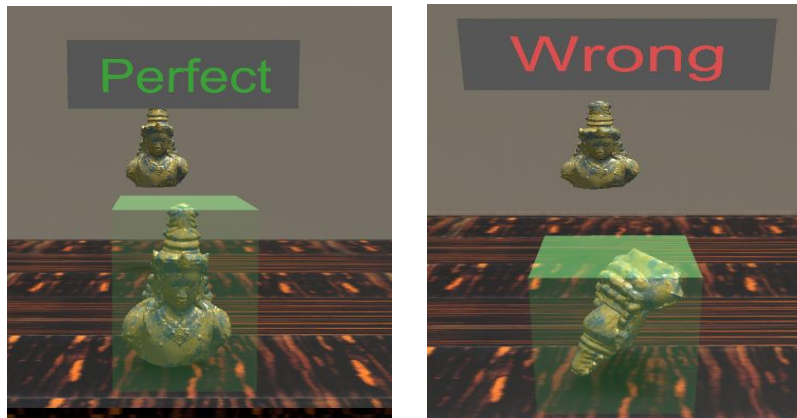


Figure 23. On the left showing the result of a correct orientation and on the right a wrong one.

4.2 VR Session Logging

To the best of our knowledge, accurate recording of a VR session can be achieved through two methods, either logging all the inputs of the user and on second stage replicate all the induced events in order to compute the desired information, or logging the effects that these inputs had. As VR advances, the “reality” of the environments expands as well; trivial interactions such as ripping a plastic case to grab a tool or removing the cap from a bottle, have turned to mandatory tasks that greatly add up to the immersive experience. This exponential content expansion is hindering the development of the latter logging method; therefore, we conclude that a VR logger that records raw user input suits the current growth direction of virtual reality environments.

4.2.1 Logging Data

User input in VR applications happens through the VR controllers and the HMD. Currently, there is a variety of HMD types and each company offers its own proprietary feature such as eye tracking, bio monitoring, haptics etc., but for our solution we only examined the common input capabilities that can be found in all VR headsets that support six degrees of freedom. These data are composed of:

- Left/Right Hand Translation (*Vector3*): Consists of 6 floats (3 for each hand) describing the position of each hand in the virtual environments
- Left/Right Hand Rotation (*Vector3*): Consists of 6 floats (3 for each hand) describing the rotation of each hand in the virtual environments
- Camera Translation (*Vector3*): Consists of 3 floats describing the position of the user's head in the VR environment
- Camera Rotation (*Vector3*): Consists of 3 floats describing the rotation of the user's head in the VR environment
- Start Interaction (*Event*): The user has started interacting with an object. Consists of the name of the interacted object (*string*), and the hand the user used (*string*).
- End Interaction (*Event*): The user has ended interacting with an object. Consists of the name of the interacted object (*string*), the hand the user used (*string*) and the amount of time the user was interacting with the object (*float*).
- Special Use (*float*): Some 3D objects provide an extra functionality while holding them, such as pressing the button for activating a drill. This input is stored as a float which represents the strength that the use button is pressed; from 0 to 1.
- Time (*float*): The time that has passed since the start of the session, in seconds.

For each session we are creating three files corresponding to the input devices the user has, right controller, left controller and HMD. Each of these files logs the transformation data per frame while also recoding events in the corresponding frame that they occur. As a result, each line corresponds to a frame and we also have indirectly access to information about the time that each event occurred. These files can get up to 10MB for a 30-minute session.

4.2.2 Non-Deterministic Physics

We believed, that by recording accurately the user's movement then we could reproduce all the consecutive events, such as grabbing an object and placing it in a container. Unfortunately, we did not take into account the non-determinism physic engine of the Unity platform.

Nowadays, interaction based on physical rules is considered essential in every high fidelity VR environment. In the case of VR training scenarios, gamified non-natural interactions impact negatively on psychomotor learning, but even worse semi-natural interaction techniques can lead to unusable and non-immersive applications [26]. Game engines, enrich their physics modules with randomization functionalities in order to provide a more gamified experience. Additionally, for optimization reasons, they do not store the 3D objects in the same specified order in each "run", which results to, physic based events (e.g. collision) that occur in the same frame, being calculated in different order between different playthroughs even if the user's input was exactly the same. These variables can lead to inconsistencies when attempting to accurately record data from physical VR interactions, such as dropping a tool on the table and

re-grabbing it later on. This issue was obstructing our advancement with the VR Supervised Machine Learning solutions, since we could not apply a generic implementation to accurately log different VR sessions and consequently extrapolate any kinds of event data needed for training a VR assessment agent.

4.2.3 Propagated Logging

Logging the movements of all interactable VR objects using a brute force algorithm is currently close to impossible in most environments due to their sheer number. A premature design was to log all the objects that showed some kind of movement, which could be identified from their change in transformation. But that also meant that we would log unnecessary information such as predefined animations and moving UIs. An implementation was needed that was able to identify and log only the moving objects which the user initiated. For the above reasons we implemented our own innovative method that logs multiple objects at the same time.

We designed our solution around the user's input, the virtual hands. All VR interactions start from the user's hands. Through the virtual non-physical hands, the user has the options to grab or trigger a 3D object. In case of virtual physical hands, he can also push them. Our idea was to create an algorithm that could "follow" the propagating consequences cause by the user's interactions.

First of all, the virtual hands and the camera, mapping the input from user's headset, are logged per frame through a Unity script we called InteractionLogger. Once the user grabs an object we record the interaction item, the hand-side that added the interaction and we add a component script called PropagatedLogging on it which logs the transformation of that object. In addition, the triggering of an object's functionality is also recorded by capturing the user's press on the trigger button. On each collision between an object that is logged and an interactable item the PropagatedLogging component is added to the latter. Finally, when the object stops moving a Unity Coroutine is started from the same script component which after a short amount of time destroys the component. On each frame all logging objects send their transformation to the LoggerManager entity responsible for writing to the respective output file. We decided that instead of having multiple files for each object that were currently tracked we needed only three files one for the camera and two for each hand. In each frame the LoggerManager stored the transformation of all the logging objects to the corresponding file. We chose this method for two reasons, firstly we don't need to open and close files for writing each time an object starts being tracked, secondly by writing all changes in transformations on the same file the frame that we recorded corresponds to the respective line.

Algorithm 2 Propagated Logging

OnApplicationStart

- 1 Add Logger script on leftHand : LH, rightHand : RH, head : H
- 2 Add listener OnBeginInteraction, on LH,RH
- 3 Add listener OnEndInteraction, on LH,RH

OnBeginInteraction (Interactable: I)

- 1 Add PropagatedLogging: PL script on I
- 2 Add HandSide: HS on PL
- 3 Set event : On collision between I and I_2
 - 3.1 Add PL on I_2
 - 3.2 Add HS on PL
 - 3.3 For each movableChild: MC in I_2
 - 3.3.1 Add PL to MC
 - 3.3.2 Add HS to PL

OnEndInteraction (Interactable: I)

- 1 Start Thread:
 - 1.1 While I is moving:
 - 1.1.1 continue
 - 1.2 Destroy PL

OnEachFrame

- 1 Write to file position:P, rotation:R, of LH, RH, H
- 2 For each Interactable: I with PL
 - 2.1 Write to file P,R of I
 - 2.2 if I is not moving
 - 2.3 Start Thread:
 - 2.3.1 timer : 0
 - 2.3.2 while (timer < 1s)
 - 2.3.2.1 timer = timer + frame_time
 - 2.3.2.2 if I is moving
 - 2.3.2.2.1 return
 - 2.3.3 Destroy PL

Figure 24. Algorithmic explanation of propagated logging.

Frame	Hand Pos XYZ	Hand Rot XYZ	Time	O1 Pos XYZ	O1 Rot XYZ	O2 Pos XYZ	...
X	(3.2 , 4.1 , 3.6)	(38 , 70 , 186)	3.1784	(2.1, 6.4, 7.2)	(41 , 89 , 95)	(4.1 , 8.9 , 9.5)	...
X+1	(3.1 , 4.8 , 3.7)	(24 , 36 , 122)	3.1852	(2.2, 6.5, 7.8)	(45 , 88 , 96)	(4.3 , 7.9 , 8.5)	...

Table 2. Example of the format logged data are stored in file for a specific virtual hand.

For testing our implementation, we created a visualization effect. Each time the `PropagatedLogging` script is added to an object it changes the material color to green and on deletion it reverts it to the original. Even though we implemented this small feature for testing and debugging purposes it can also be used for showcasing how our implementation works.

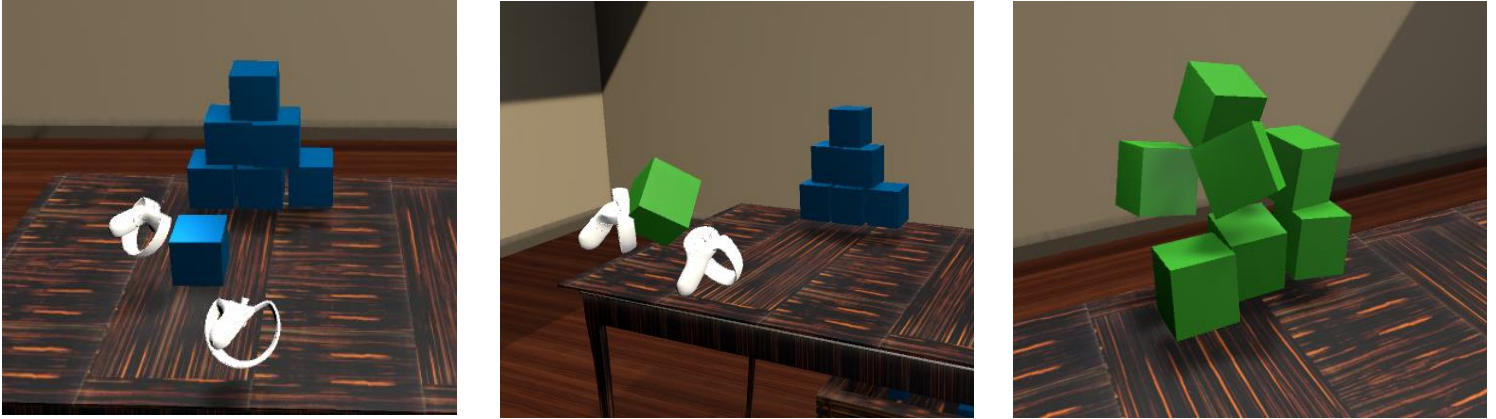


Figure 25. On the left only the transformations of the user's hands and camera are logged. On the middle image the user has grabbed the object adding to it the component responsible for logging. On the right image the object is thrown propagating the logging component to the rest of the cubes.

5 Demos

To showcase how this work can be applied, three VR demo scenarios were developed, using the Unity Engine, each focused on a different part of this thesis. The first demo consists of a small VR training scenario that assesses the user's actions by utilizing the scoring factors, described in chapter 3; the second demo is focused on ML based assessment, and the third one on logging the user's data in a VR session and recreating the session from them. For tracking the VR hardware as well as handling its input the SteamVR SDK was used and for interacting with the virtual environment we chose the NewtonVR Unity Asset.

5.1 VR Analytics based Assessment

Before employing the analytics system to assess the user, a method to define and control the flow of the tasks was needed. For this purpose, we created two entities the ActionBase interface and the ActionsController. The controller is responsible for managing the flow of the scenario by keeping track of the current task and calling the necessary functions for initializing it and cleaning it. The ActionBase interface's purpose is to provide an easy way for initializing a task by spawning the 3D objects with the needed functionality, as well as cleaning each task by deleting the objects and reset any non-permanent changes done to the environment. The ActionBase interface provides similar functions with the scoring factor interface (the initialize, perform, undo functions); this is a design choice for easier integration with the analytics platform and not a requirement for utilizing our analytics.

For the training scenario a simple theme was chosen, cleaning and preserving a cardboard box. This scenario is composed by three tasks:

- Deciding over the most appropriate tool for cleaning the box.
- Cleaning the dust and dirt spots from the box.
- Carefully placing the box on the shelf.

For each task defining its scoring factors was needed. Below you can find the scoring factor that were used to accurately assess the user:

1. **Question Factor and Time factor.** The decision for which tool to use is done in the form of a multiple choice question. We want the user to answer based on his existing knowledge and not by researching a website; thus we used the time factor.
2. **Hit Colliders Factor and Interact Object Factor.** Since the user needs to clean all the spots from the box, we set up all the dirt spots colliders in the hit colliders factor and we defined

the interactable object to be the cloth. Additionally, the user needs to hold the box while cleaning it, for this we are using the interact object factor.

3. **Placement Factor, Velocity Factor and Error Hit Factor.** Since the user needs to carefully place the box on the shelf we are using the velocity factor to make sure he is moving it slow and steady. For measuring the accuracy of placement we are using the placement factor and finally, since there are other boxes on the shelf the error hit factor is used in case the user hits the fragile box with them.



Figure 26.

*First task (left) the user needs to answer the question.
Second task (middle) the user needs to clean the dust spots on the cube.
Third task (right) the user needs to place the box on the shelf.*

In addition, two user interfaces were created the options panel and the details panel. The options panel contains two buttons *perform* and *undo*. The trainee uses these to traverse through the training simulation. The *perform* button completes the current task, calculates the user's score and initializes the next task. The *undo* button clears all objects of the current task, resets the user's score in the previous task and initializes it. The details panel presents the results of the previous completed task and their explanation. The user has the option of clicking on the *task_name* field, which opens a new UI panel containing the details of each scoring factor. These panels output the following values to the user:

- *Score*: the final score of the previous task.
- *Errors*: the amount of errors the user conducted in his previous task.
- *Time*: the time it took the user to finish the previous task.
- *Multiplier*: the weight corresponding to the importance of the previous task.
- *Task_Name*: the name of the previous task.
- *ScoringFactor/Name*: the descriptive name of this scoring factor.

- *ScoringFactor/Score*: the resulting score of this scoring factor.
- *ScoringFactor/Importance*: the weight of this scoring factor.
- *ScoringFactor/Performance*: the specific output of this scoring factor, e.g. in the time factor, the completion time of the user for this task.
- *ScoringFactor/MaxTarget (situational)*: the maximum output of this scoring factor, e.g. in the time factor, the maximum completion time the user has in his disposal.
- *ScoringFactor/ErrorMessage*: the error message shown to the user in case he performs poorly.

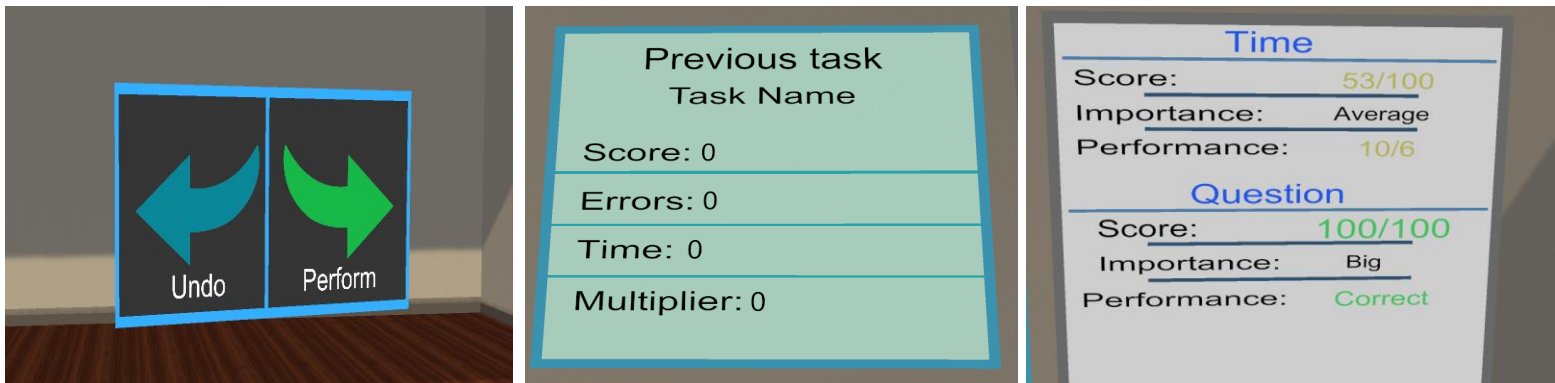


Figure 27.
 On the left the buttons for performing and undoing a task are shown.
 On the middle the panel showing the results of the previous task.
 On the right the panel for presenting the results of each scoring factor is shown.

Finally, for this demo two types of notification UIs were created, one for errors and one for information. Error UIs are spawned when the user performs poorly in a specific task and contain the corresponding message that has been defined by the developer on the corresponding scoring factor. Notification UIs are constructed at the start of each task and contain guidelines on how to perform it correctly.

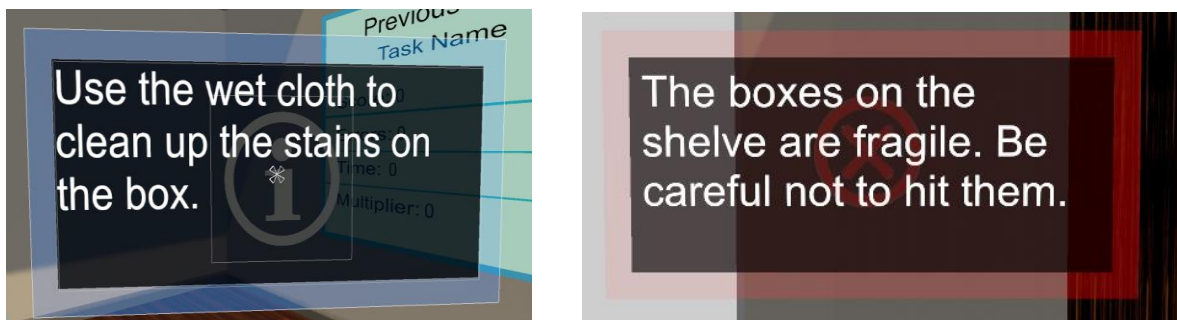


Figure 28. Right: The UI used for feeding information to the user.
 Left: The UI spawned when user performs an error.

5.2 Machine Learning Assessment

For the second demo, the user was tasked to place a 3D object on a specific location, with a specific orientation. Instead of using the placement scoring factor for assessing the user's actions; our methodology involving convolutional neural networks was utilized in this demo. The first step of this demo was asserting what kind of 3D objects to use. We chose a Buddha statue that was characterized by its asymmetrical features.

For our object an area of interest was defined and the image data needed for training our models were generated by following the method explained in chapter 4.2.1. These data were used for training one CNN model, utilizing our *multi-view* methodology. At runtime, the placement area, a static copy of each object with the correct orientation and an interactable copy are spawned for the user. Once the user places the statue inside the placement area each virtual camera records an image, passes it through its layer masks, removing any unnecessary artists. The four images are then fed to our CNN model which predicts the assessment output, out of the three predefined labels, *wrong*, *pass*, *perfect*. Finally, the prediction is presented to the user.



Figure 29. Our demo showcasing our ML implementation.
The green cube serves as the area of interest with the four cameras around it for generating images of the statue's position.

5.3 Logging VR Sessions

For showcasing our session logger, a VR playground environment was created. The demo allows the user to record his own session and then watch it through an NPC replicating his movements. We position the user in a room filled with interactable items. Our goal was to create a small playground where the user could create a variety type of events, in order to showcase the reusability and interoperability of our logger in different kind of scenarios and different physics interactions. The following items are placed in the scene:

- Tower of cubes. This serves as the most representative example of our module. The user can hit the tower with an interactable breaking it, thus triggering multiple components of the physics system

(collisions, gravity, friction). Since, our logger accurately records their transform, when the user decides to replay this session the cubes will fall at the exact same location.

- Lever. For implementing the physical interaction of a lever a Unity Hinge Joint was used. The user can pull the lever, and after a short amount of time it will return to its original position. We chose this object to showcase that our logger works robustly across different physical restrictions.
- Drawer. For implementing the physical interaction of a drawer a Unity Configurable Joint was used. The user can open the drawer, place objects inside them and after a short amount of time the drawer will automatically close. Since our logger also records objects that are not directly rendered on the camera, in the playback the object will result in the exact same position.
- Buddha Statue. A simple example of an interactable. The user can grab it, move it and place it.



Figure 30. VR playground for showcasing the capabilities of our VR Logger.

Although, there is still a long way to go in order to be able to accurately map input from VR users of diverse height and arm length to any type of realistic rigged human or humanoid character, we can map our input to playful avatars with similar body parts as the user. For directly mapping the user's movements to the NPC the same 3D model for visualizing the virtual hands was used, which was the Oculus Controllers while for the head a 3D model was designed promoting gamified robot-like features. While the NPC is replaying a session, it is not able to adapt to any kind of change; the smallest of modification in the environment will make it impossible to complete. For the purpose of avoiding this the user is placed in the VR room with the role of an observer, being able to move and inspect the virtual space but being unable to interact with any virtual items.

Even though, our goal for creating this module was to generate accurate VR data for machine learning while developing this demo new ideas rose for how to utilize it. Since, we can replace the user with an avatar, by expanding our module to log microphone input, a tool can be developed which enabled the teacher to create educational VR “videos” on how to correctly complete the training scenario. Considering the recording is not just a video, the user can move around the virtual room and observe the teacher’s actions from any point of view. In addition, by creating a tool that visualizes on unity’s editor and provides functionality for editing these sessions data, we can create small VR playbacks providing guidance as he completes his tasks.



Figure 31. Image from our playback demo, where the instructor is shown opening a drawer.

6 Evaluation

6.1 Self-assessment

6.1.1 Analytics Based Assessment

Our analytics system was designed around structuring and consolidating the development process of the assessment method of a VR training scenario. By requiring all members of a team to follow an organized methodology for creating such systems leads to reduced errors, straightforward communication and easier maintenance, since it will be simple for any developer to understand and alter previous work done by colleagues. In addition, by utilizing this structure, visualization and explanation of the assessment results, can be generalized providing a consistent report among tasks and training scenarios. Furthermore, by introducing the scoring factor entity we componentize this system, allowing for reusability and variety of combinations that the developer can choose from to achieve the desired result. Finally, arranging a formulated scoring methodology through different training scenarios can contribute to easier comprehension from users on how they were assessed, since they need to understand one scoring method for all possible simulations.

The analytics authoring tool serves as an all-inclusive editor for the developer to define the assessment method of each task. It was designed around the policy that this process will take place exclusively through its user interface. This approach was chosen in order to have all the assessment material and assets concentrated in one development environment, eliminating the tedious process of searching through all the scripts and assets that are needed for other components of the simulation, thus providing easier maintenance and debugging while also controlling the integrity of the resulting score. The downside of this practice is that it obliges the developer to constantly upgrade the editor for every new assessment method, even though sometimes scoring factors are purposed for a one-time use thus developing using C# code being a faster approach. Additionally, it was built on top of Unity Editor's framework that grants direct integration with all its assets and file formats while also allowing the user to work in a single framework, eliminating the need of switching between different editors.

In order to use our authoring tool, familiarity with terminology of interactive 3D graphics is needed, such as colliders, interactable objects and spawning, as well as basic understanding of the functionalities provided by the Unity Editor. Even though, this knowledge is easily learned from an unexperienced VR programmer or 3D artist, it is not simplified to the extent that an SME from another field can easily grasp it and develop with it.

6.1.2 Machine Learning Assessment

By employing ML techniques, we improved the capabilities of this project in accurately assessing the performance of tasks that are time-consuming to define when using traditional coding techniques. Our implementation is capable of learning the variables needed to assess the virtual placement of an object from a generic type of input, images. This procedure can prove a challenge to author it precisely from a 3D editor environment partially due to symmetrical objects that lead to multiple correct answers but also due to the degree of freedom in completing arbitrary tasks, such as *placing an object on the table*. Even though the desired result can be achieved by both technologies, the simplification of authoring the evaluation of complex tasks leads to decrease of development time and errors. Additionally, by utilizing our methodology we simplify the usage of a VR editor tailored for editing the assessment method. The process of defining all the parameters needed for the evaluation of a task is altered into labeling the users actions and automatically learning the needed variables from the generated data. As a result, the user is able to author the scenario with the minimal negative impact on the immersion provided from the VR simulation.

By simplifying the procedure of constructing the evaluation to one that requires knowledge only on the field of the training scenario, the simulation can be authored directly from an SME. This eliminates potential errors risen from the communication of the SME with the developer, while also ensuring the integrity of the evaluation. The main problem that occurred in this project is the lack of real-time feedback on the progress of the accuracy of the CNN model. Since, the SME lacks the information of how many sample data are needed he might finish the authoring process prematurely. This can lead to the tedious cycle of restarting the VR simulation to generate extra data and checking the accuracy of the ML model on each iteration. Therefore, to reduce this time overhead a VR ML training module will be developed providing the SME to train ML algorithms by pressing a specific button on the VR controller.

6.1.3 VR Logger

In this project, the VR logger module was introduced. This module allows us to log the user's sessions within the virtual environment in the form of positions, rotations and interactions resulting to improved accuracy without compromising generalization. Our main goal for developing it was to produce more descriptive data than images, that can express the location of an object in the 3D environment with precise accuracy, resulting to a reduce of the amount of poses required to train supervised ML agents and accordingly, further reduce the time spent by the SME or the developer authoring the simulation.

In addition, the accurate output recorded from our VR logger allowed us to develop the playback feature. Our primary objective for building this feature was to test and prove the accuracy of our logged data but we also showcased how it can be applied for creating high fidelity VR recordings that guide the trainee through his tasks. Documenting the user's session in this form leads to a variety of features that can

enhance VR applications and their development. Replaying the session of a user can prove especially useful for visualization purposes, where humans can directly annotate these data for training ML agents but also evaluate the trainee's performance. Trainees can also benefit from the playback feature personally, observing and studying their previous actions can serve as a form of feedback. A drawback of this implementation is its inability to adapt to new updates of the training scenario. Updating the starting position of a 3D object or its interaction method invalidates all previous recorded sessions. Logging spawn events accompanied by the initial position and rotation of the object and altering the VR Logger to record relative data from the initial transform might serve as a solution.

6.2 Qualitative Evaluation

We conducted a qualitative evaluation in order to provide quantified metrics on our VR assessment platform. In more detail, we wanted to examine if our analytics system was capable of producing the assessment metrics of a VR training scenario, in a short amount of time, without compromising accuracy, fairness or the overall VR experience. Additionally, our machine learning implementation was compared with the above system, in order to determine if there was a positive impact by employing ML in the assessment pipeline of a training scenario. In this experiment 10 users participated, which were asked prior to this study their familiarity with VR applications on the scale of one to seven, one for not having any experience with the technology seven for having VR development experience for more than 3 years. We classified these participants into two categories; users that reported above 5 were branded as *VR Experts* (6) while below, as *VR Beginners* (4). Six were branded as *Experts* while four as *Beginners*. We believe this distinction was necessary due to the excitement that VR brings forth to new users, thus implicating their impartiality on the answers.

This evaluation was split into two parts. In the first study users were tasked with completing the *Cardboard Box Preservation* training scenario and fill a survey on the accuracy of the assessment. Continuing, they were tasked with comparing the assessment accuracy of placement scoring factors and our ML implementation. A small demo was created that put both of these methods side by side and the users reported on the accuracy of each implementation. The purpose of this session was to determine if our DL agent was as accurate as our rule-based implementation or better.

Prior to the start of the first study users were introduced to the VR controls, what to expect from each UI panel, as well as the type of training scenario they will need to complete, without explaining any of the underlying technologies. The *Cardboard Box Preservation* training scenario was chosen due to its simple content; no particular expertise is needed to accurately complete it. At the end of each task the users studied their results from the UI panels and filled a survey concerning the accuracy of their assessment and whether the feedback was well-explained and satisfactory; both of these answers were in the range 1 to 7.

In addition, we asked for their comments regarding the metrics used and if in their opinion there are other contributive factors that we should include to further improve the accuracy of our system.

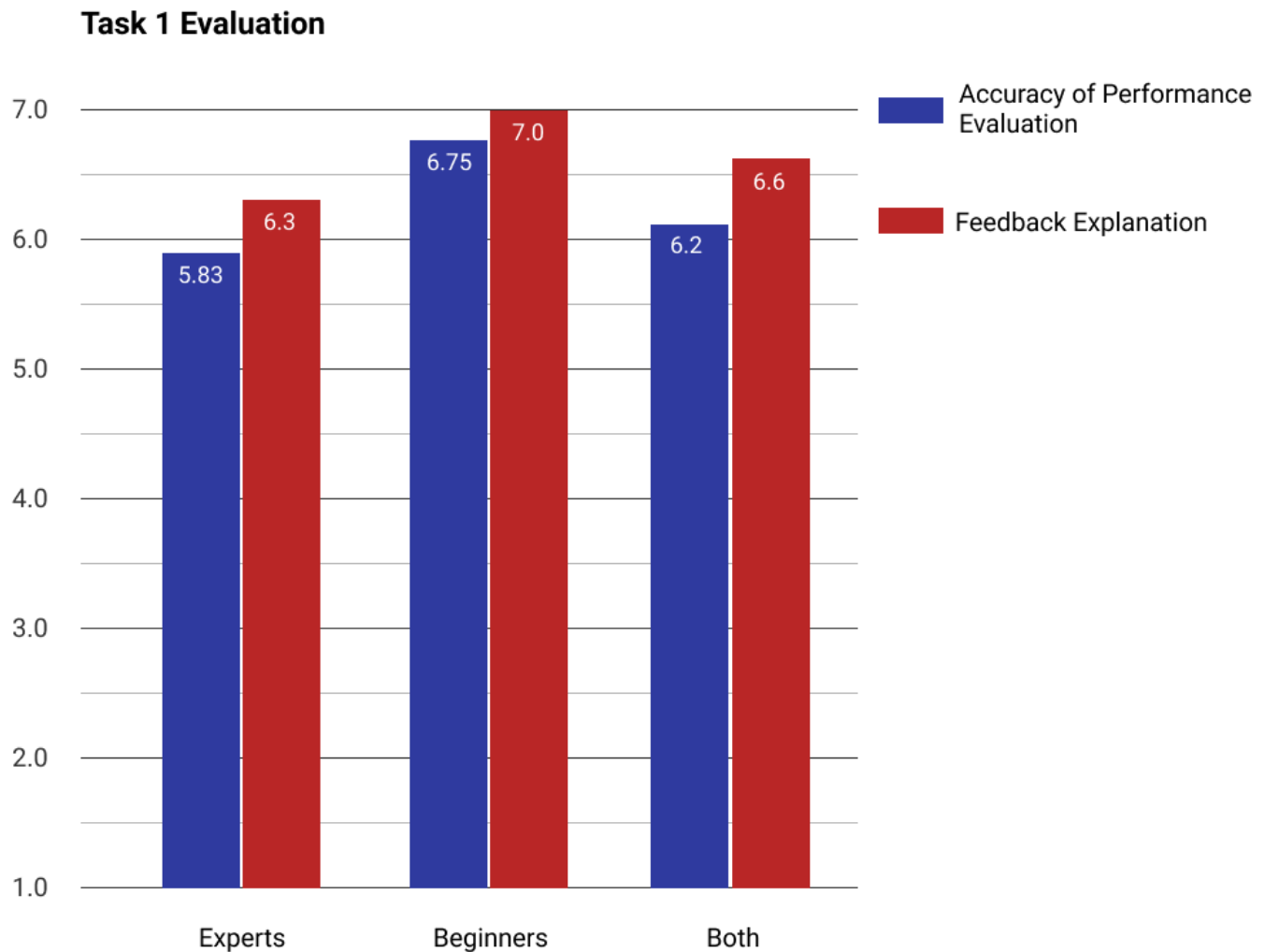


Figure 32 Evaluation of task 1 of the Cardboard Box Preservation scenario.

From the survey results of task 1 a substantial drop on both metrics between experts and beginners is noticed. As they commented, this was due to the short amount of the amount of time they had to answer the question, as well as the lack of explanation that time was a factor on this particular task.

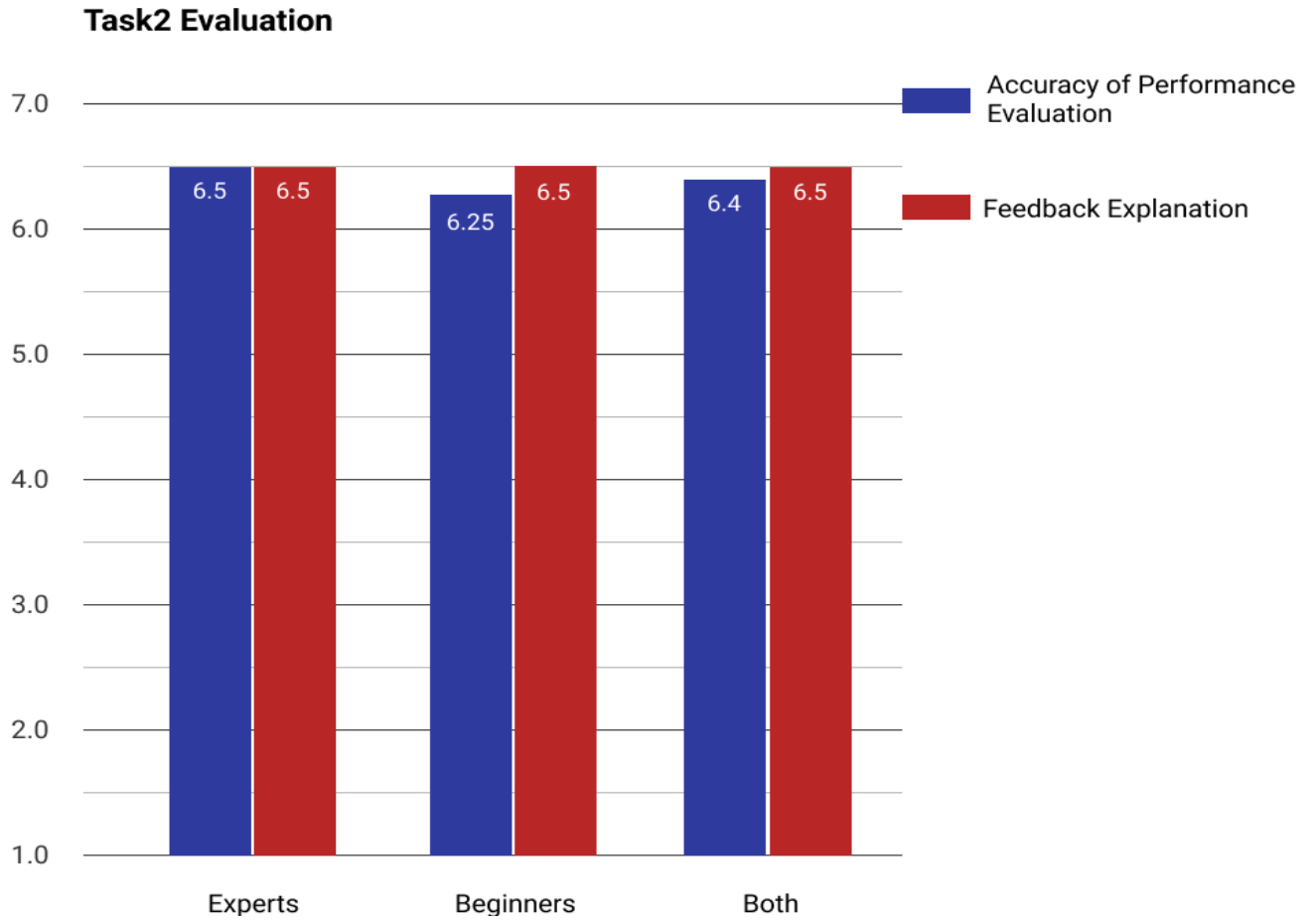


Figure 33. Evaluation of task 2 of the Cardboard Box Preservation scenario.

The second task of the scenario consists of psychomotor skills, since users have to hold the box with one virtual hand while cleaning it with the other. A few beginners had trouble performing accurately due to their inexperience with using the VR controllers. As a result, they commented that the assessment of their performance was harsh.

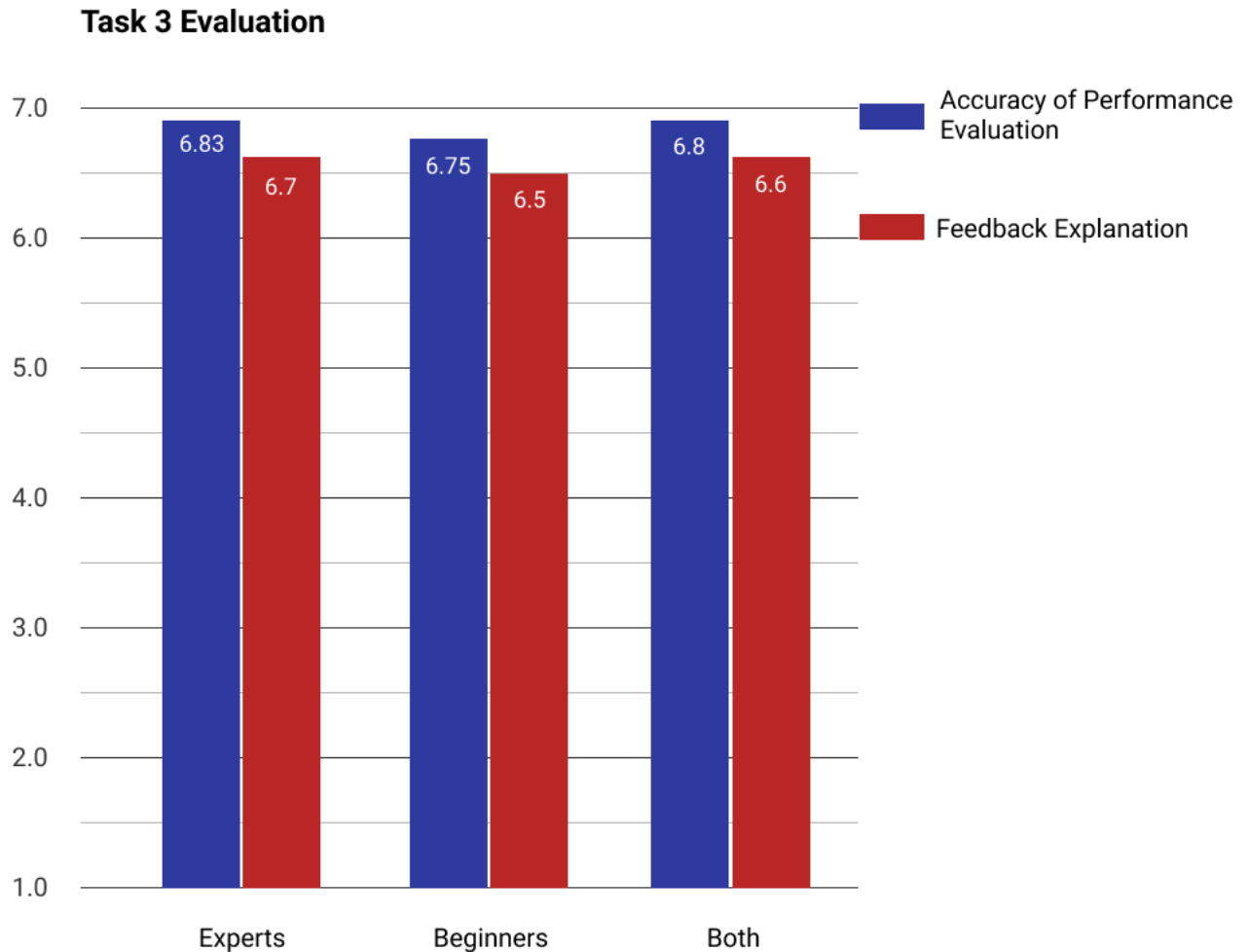


Figure 34. Evaluation of task 3 of the Cardboard Box Preservation scenario.

In the third task, we do not notice a substantial difference between beginners and experts and they did not have particular comments for this task.

Overall, most of users criticized the lack of content specific explanation of why each scoring factor was important for the current task. Even though, they did not have a hard time understanding *how* each scoring factor was grading them, they also expected an explanation on *why* this was important. Since, this feature is currently not supported, we plan on adding it in the future. In addition, they stated that time should be a contributive factor in all tasks, even if trivial, since it showcases the familiarity of the trainee with the environment.

In our second evaluation demo users were asked to insert the Buddha statues in their respective placement boxes. Static objects representing the perfect orientation were added in the scene. In order to accurately compare our ML implementation with the placement factor we mapped the output score of the

placement factor to wrong, pass, perfect. In detail, users with score above 80 got the perfect mark, users with score above 50 got pass, and the rest wrong. In addition, to see how much our evaluation methods matched the innate understanding of the users, they were tasked to achieve all three results with both performance assessment implementations with the least amount of tries. Furthermore, they graded the performance assessment for each example similarly as the first part of this study. Finally, to avoid any kind of bias we split our evaluators into two groups. The one group started with the scoring factor implementation and then continued to evaluate our AI, while the rest evaluated them in the opposite order.



Figure 35. The evaluation demo comparing the neural network with the scoring factor. The left statue and placement box uses our ML implementation while the right use the placement factor.

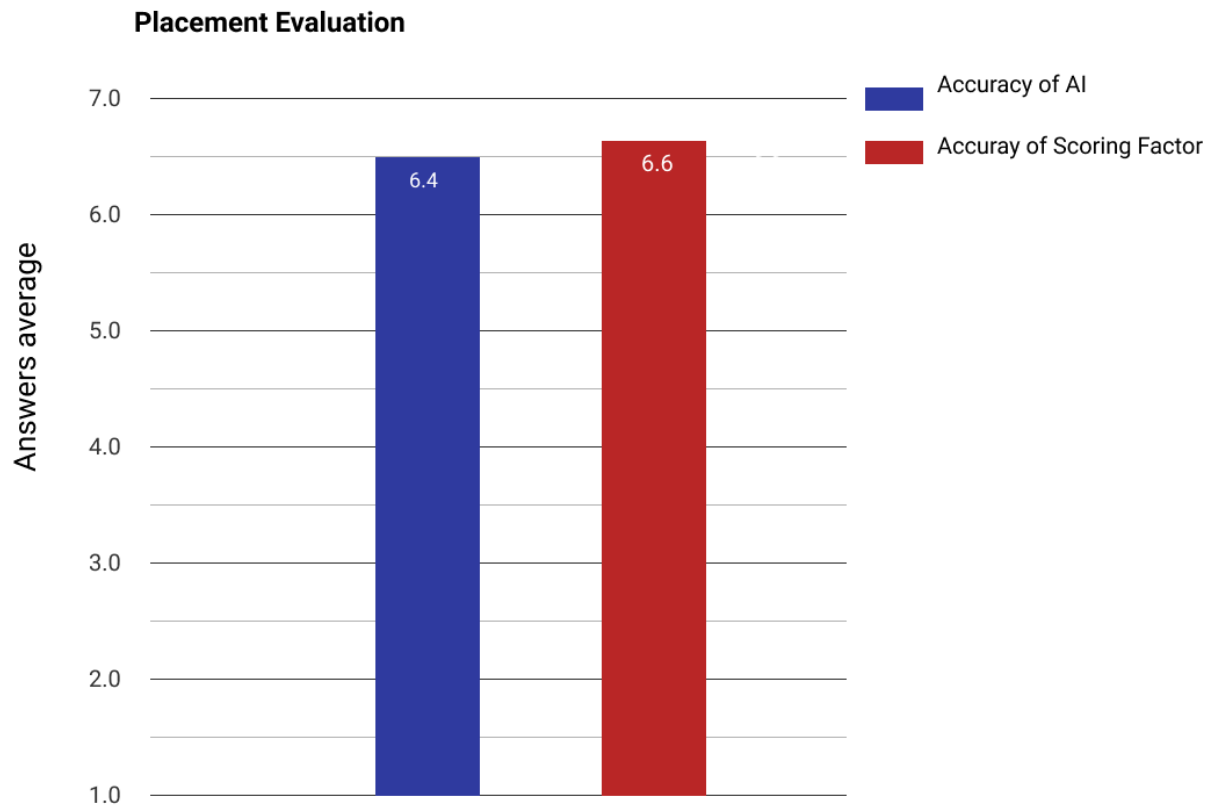


Figure 36. Users evaluation on the assessment accuracy of each implementation.

As we deduce from figures 30 and 31, both implementations provide similar results. From the users' feedback as well as our observation during the demo we noticed that they could not easily achieve the *perfect* result in the ML placement example. We believe this was due to labeling only the ideal orientation images as perfect, during the ML training, thus making it difficult for the users to achieve this result.

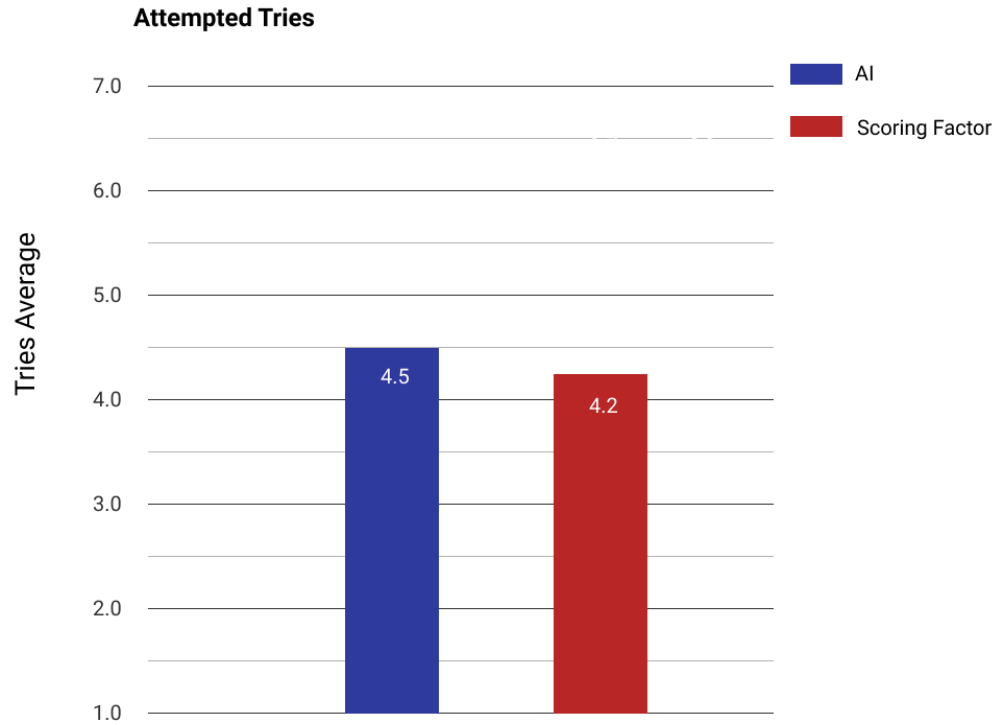


Figure 37. Average attempted tries for each placement.

6.3 Metrics

In this chapter we present the metrics and graphs that influenced us in the implementation and optimization of our VR assessment agent. In addition, performance metrics on our VR logger module are presented both in usual scenarios as well as under heavy stress testing.

6.3.1 Machine Learning Optimization

Throughout the development process we noticed that three hyper-parameters had a noticeable impact on the accuracy of our ML agent, a) batch size during training, b) number of epochs, c) different optimizers. The number of sample images also had a heavy influence in the training of our ML for VR but due to our goal of creating a rapid and repeatable process for training different models, a small dataset was enforced. For the following graphs 88 poses were used of the Buddha 3D model. They were split at 70% (61) for training, 15% (13) for validation and 15% (13) for testing.

We started with choosing the right amount of epochs for training our CNN models. Since, multiple runs are needed to find a random seed that produces appropriate results, training our models with excessive

epochs can lead to big computational times. In addition, in the future we aim to trigger the process of training directly through the virtual environment, giving feedback to the developer about its results in a sensible amount of time. Thus, minimizing the epochs needed for training was important. Through our testing we concluded that the optimal number of epochs is 50. From the graphs shown in figure 18 and 19 we can see that our ML stops increasing its accuracy at the range of [30, 45], thus an epoch of 20 might lead to inadequate learning while 70 to unnecessary learning and overfitting.

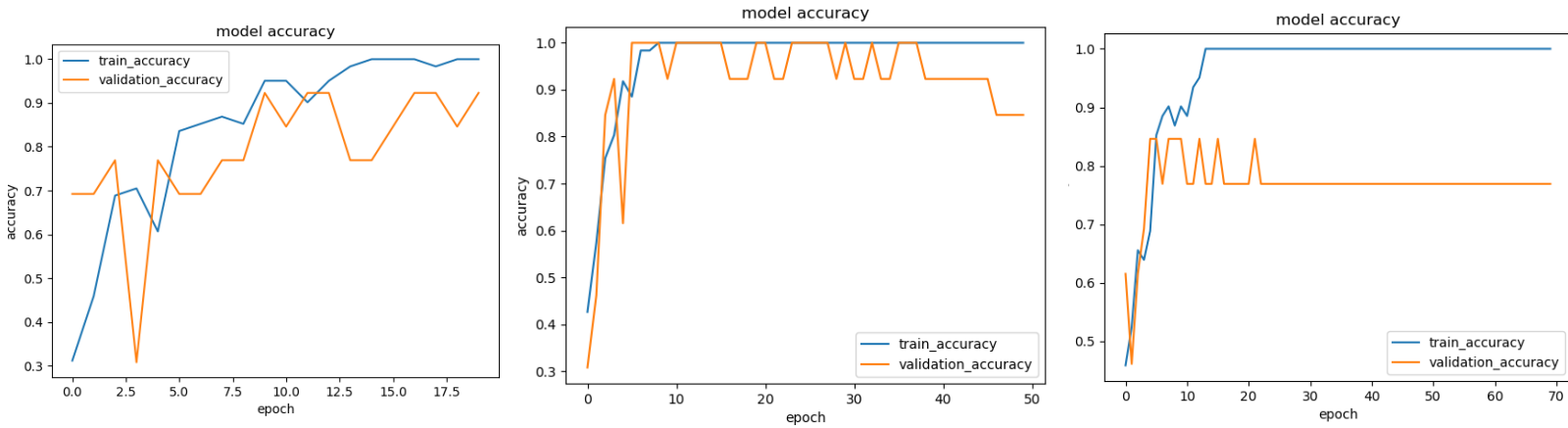


Figure 38. On the left the accuracy graph for 20 epochs is depicted, on the middle for 50 and on the right for 70. These graphs depict the training process with different random seeds.

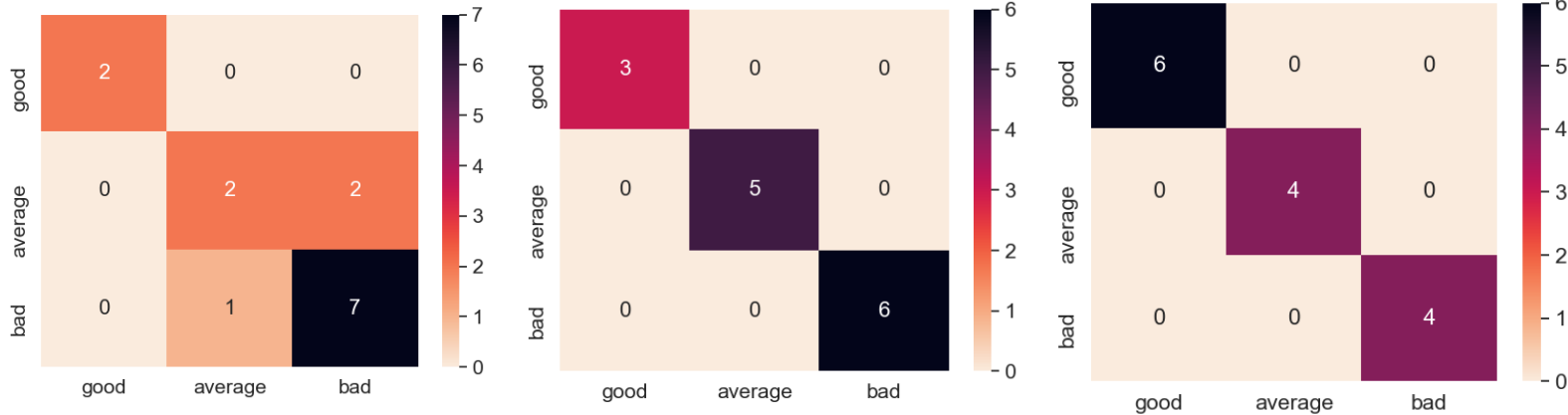


Figure 39. The confusion matrices for the above training runs, showcasing how each model behaves to new data (testing dataset). On the left the confusion matrix for 20 epochs is depicted, on the middle for 50 and on the right for 70.

Batch size is the number of training examples used in each forward/backward pass. Due to our small dataset we concluded to use small batch sizes, specifically 2, 6 and 10. From our optimization runs we concluded that the results for batch size of 2 and 6 were similar. Due to smaller batch sizes producing more passes at each epoch thus more computational time, the latter option was more preferable. Even though batch sizes equal to 10 behaved similarly on most runs, we noticed that for specific random seeds produced less accurate result than when choosing a smaller batch size. Thus we concluded that the optimal batch size for training our CNN model was 6.

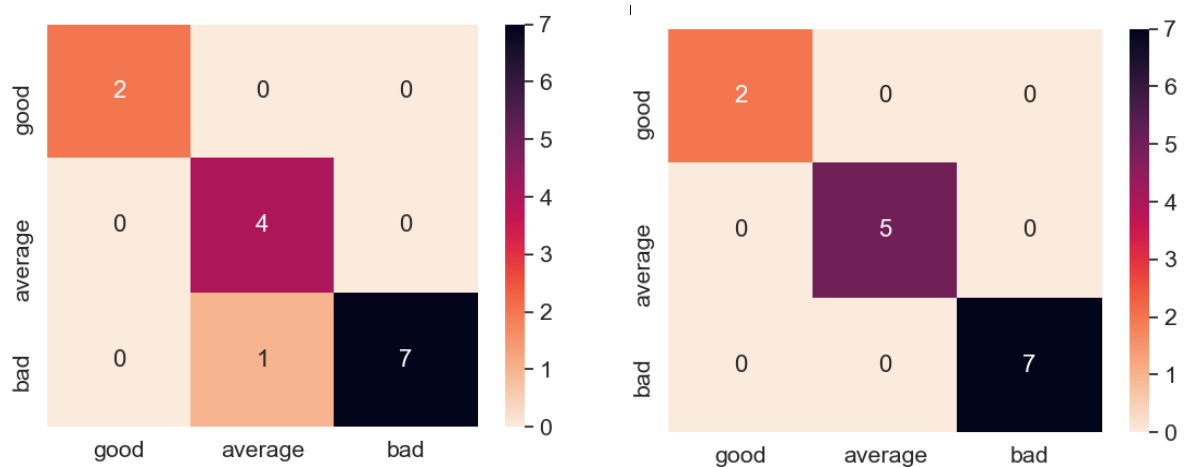


Figure 40 Confusion matrices graphs for Batch_size=10 on left and Batch_size=6 on the right. Both produced from the same random seed and dataset.

Finally, we wanted to examine how the neural network was learning when different optimizers of keras were utilized. We compared keras Adadelta optimizer with the SGD optimizer. The most noteworthy difference was the increase in epochs needed from the SGD optimizer to achieve similar results as the Adadelta optimizer.

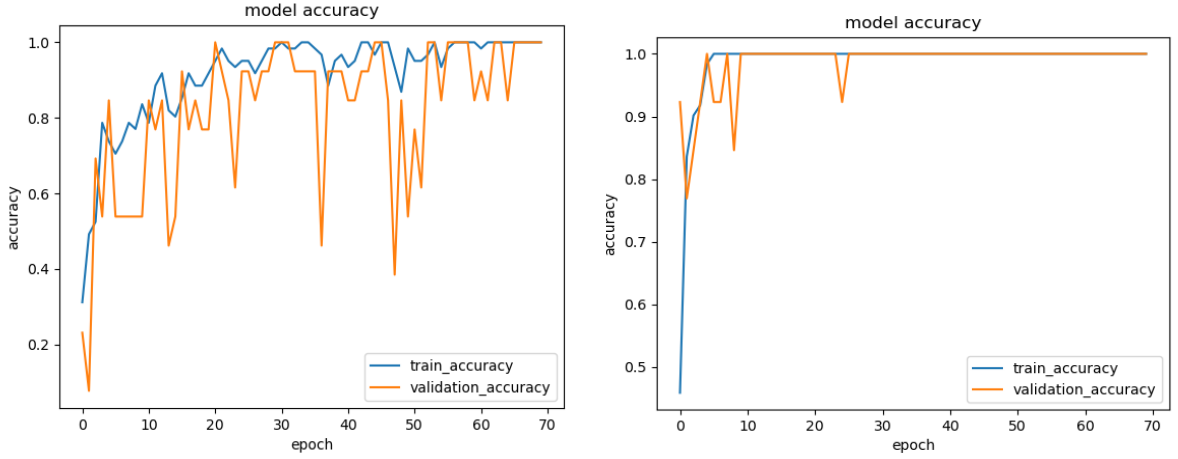


Figure 41. CNN accuracy on each epoch.
The SGD optimizer was used for the left graph while the Adadelta optimizer was used for the right.

As mentioned above the computation time for training was crucial to this project. Thus, the Adadelta optimizer was preferred over SGD for our implementation.

6.3.2 VR Logger Performance Metrics

Our VR logger module was designed around its usability in different virtual environments. This led us to examining its impact on the computed frames per second (FPS) in different scenarios. We conducted two evaluations, one in our virtual playground described in section 5.3 and one in a new environment designed for stress testing our implementation. For each evaluation we recorded the framerate in two runs, once with logging enabled and once with logging disabled. In order to provide accurate results in both tests that we run in our playground, similar events were triggered. Our results are reported in table 3.

Metric	Logging Disabled (FPS)	Logging Enabled (FPS)	Difference
Average	89.56	85.13	4.43
Minimum	76.56	68.78	7.78
Maximum	93.29	92.57	0.72

Table 3. Performance metrics for VR Logger Module in our VR playground environment.

From table 3 we can discern that our VR logger is not affecting negatively the maximum framerate, which was expected due to this framerate being achieved when not a lot of physical events are computed, and accordingly not being logged. Continuing, with the average difference we can identify that our module has a noticeable impact on the application but not to the extent where it greatly impacts the user's experience.

Finally, the drop in the minimum metric was expected, since it is produced when the most physical events are calculated, which in turn need to be logged.

For our second evaluation VR logger module we added 500 3D cubes in the scene, and once a keyboard button was pressed, indicating the commencing of the simulation, a significant amount of force was added to them. We modified our algorithm to log the objects transformation data once they started moving. This was necessary due to the force originating from the press of a keyboard button and not from the user's hands. We wanted to examine if the computational tax from our logger had a significant impact on the framerate compared to other computational heavy processes such as lighting, physics, rendering.

As reported in table 4, the average framerate had a significant difference. Due to logging all cubes in most frames the difference between the two framerates was close to the one reported on the minimum metric. An important observation is the minimum and average framerates in both runs are significantly low for a VR application which targets 70-90 fps. This leads us to the conclusion that even though our solution has a noteworthy impact on framerate, it will not directly affect the development process, due to the effectual scenarios being optimized or discarded for other technological reasons, such as many physical events or rendering a plethora of objects.

Metric	Logging Disabled (FPS)	Logging Enabled (FPS)	Difference
Average	66.1	55.74	10.36
Minimum	23.88	13.59	10.29
Maximum	78.31	77.60	0.71

Table 4. Performance metrics for VR Logger Module in our VR stress test scenario.

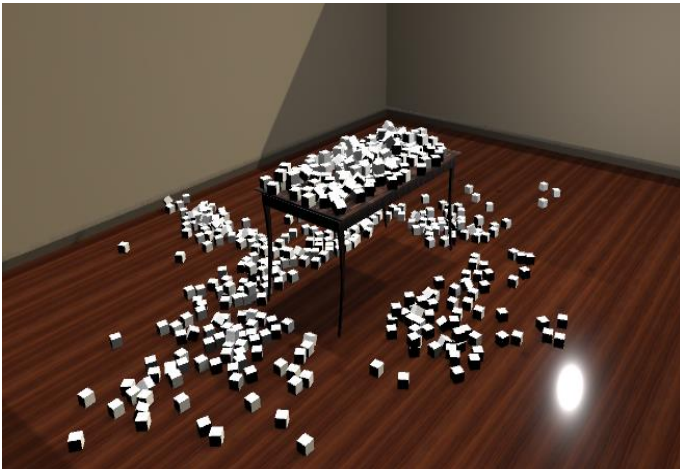


Figure 42. Images of the stress test scenario. On the left the cubes are in idle. On the right force was added to them.

7 Conclusion

In this chapter, we will explain how this project achieves its primary goal and also shortcomings and deviations from the original plan. Following, we will explain what is our planned future direction regarding augmenting this project with new features but also improving and optimizing the current implementation, in order to provide a universal process for VR training assessment.

7.1 Achievement of Goal

The initial goal of this Master's thesis was to provide a generic, cost-effective process to author the performance assessment part of a VR training scenario. Currently, the only available authoring tools and systems for assessment in 3D environments offer this functionality targeting Serious Games. Even though these could potentially be applied to VR applications, by specifying the problem only to the VR training space we are able to produce an accurate and friendly to use solution.

The analytics based assessment coupled with the VR scoring factors, provide a generic and systematic tool that can easily generate an evaluation process for a training scenario. By, employing a structured procedure, expanding our system becomes a trivial process, by defining new scoring factors, while also allowing us to create an authoring tool that can simplify and speed up the process of applying the SME's guidelines. Furthermore, by utilizing invariant data for training machine learning agents from the virtual environment we are creating a new possibility of generalization for our system, since we eliminate the need for the development of new scoring factors. At the same time, we translate the authoring process to one that the SME is familiar with, performing the scenario, attaining forthrightness and accuracy integrity.

The VR Logger component enables us to capture the most common actions the user can perform in the virtual environment in a way that is constant among different training scenarios and fields. Unfortunately, we did not achieve our initial goal of training assessment ML agents that utilize these types of data to create a comprehensive assessment platform.

7.2 Future Work

Machine Learning tailored around VR training is still in its infancy. By utilizing our system, a plethora of new technologies can be born that can rapidly advance the growth of this field. First of all, the use of raw VR data for training, both with supervised and reinforced learning, can produce agents that guide and assess the user's actions. We aim to integrate the VR Logger component in our VR editor tool to record the raw data of a task performed by the SME and not only in predefined areas in order to create ML models that can assess the trainee by comparing his actions to the expert. Another ML feature that can enhance VR training is the smart automatic completion of tasks. Currently, either the user has to complete predefined

events to move to the next task or explicitly inform the system that he has finished. The first option does not allow the user to neglect a step, which is one of the most common mistakes, while the latter diminishes the user's experience and immersion by constantly distracting him from his training. We believe an agent can be trained to predict when the user has completed his task based on his movement and interactions, e.g. in some tasks averting his field of view from the area of interest and leaving the tool on the table can be recognized as the event for moving to the next task. Additionally, by employing supervised or reinforcement machine learning in VR an adaptive training mode can be created, personalizing the training experience to each user. By utilizing the raw data and the assessment results in previous sessions the agent can learn which guiding entities (UIs, holograms, aid-lines, etc.) prove helpful to the user, and generate an environment tailored around the user's strengths and weaknesses. Through challenging him on every step, while simultaneously supplying sufficient guidance the simulation can deliver a gradual learning curve. Finally, we plan to incorporate this thesis together with the aforementioned features and the work done in [40], to create an intelligent tool for authoring VR training scenarios as well as assisting the user in completing them.

Currently, when using our VR ML module for training ML agents, the user needs to exit the virtual environment and manually run the python script for training the CNN models. In order to avoid this tedious process, we plan to offer the user the option to trigger the training script directly through the VR environment, optimally giving feedback for its progress. This allows him to continue with the development of other tasks while the ML agent is being trained.

Even though our VR Logger can log the most common and descriptive types of VR data, there are still some data that can augment the agents learning, such as force applied to an interactable object or collisions with the static environment. Even though in most cases the information these types of data provide can be excavated from the resulting changes in transform, in specific cases of heavy objects and static objects with heavy friction this information is vital for producing an accurate ML model, and it can greatly reduce the amount of training data needed.

Continuing with the analytics platform, we plan to merge our scoring factors implementation with the VR Logger module in order to create a module that produces coherent insights for each session, such as preferred hand for each tool, most used interaction points on an object and metrics on the least interacted tools. Subsequently, these insights can be to the supervisor or the developer to effortlessly understand how the users interact with the virtual environment and adapt the simulation to fit their needs.

Finally, in our opinion each approach for VR assessment presented in this thesis is best suited for evaluating different types of tasks. Based on our intuition we suspect that the assessment of cognitive tasks, such as multiple choice questions, are established easier by exploiting the analytics based methodology,

while psychomotor tasks by the ML implementation. A formal study proving or invalidating this assumption would prove particularly interesting since it could prove vital in pointing the future development of this work towards the correct direction.

References

- [1] G, Babu. (2017). Enhancement of Learning Through Collaborative Learning. newmanpublication. 4. 29-38.
- [2] Baka, Eva & Kentros, Mike & Papagiannakis, George & Thalmann, Nadia. (2018). Virtual Reality Rehabilitation Based on Neurologic Music Therapy: A Qualitative Preliminary Clinical Study. *Learning and Collaboration Technologies. Learning and Teaching* (pp.113-127). 10.1007/978-3-319-91152-6_9.
- [3] Baker, R.S. Stupid Tutoring Systems, Intelligent Humans. *Int J Artif Intell Educ* 26, 600–614 (2016).
- [4] Joseph Bates. 1992. Virtual reality, art, and entertainment. *Presence: Teleoper. Virtual Environ.* 1, 1 (Winter 1992), 133–138.
- [5] Francesco Bellotti, Bill Kapralos, Kiju Lee, Pablo Moreno-Ger, Riccardo Berta, "Assessment in and of Serious Games: An Overview", *Advances in Human-Computer Interaction*, vol. 2013, Article ID 136864, 11 pages, 2013. <https://doi.org/10.1155/2013/136864>
- [6] Bohil, Corey & Owen, Charles & Jeong, Eui & Alicea, Bradly & Biocca, Frank. (2009). Virtual Reality and Presence. *21st Century Communication: a reference handbook* (pp.22) 10.4135/9781412964005.
- [7] A. Calvo, D. C. Rotaru, M. Freire and B. Fernandez-Manjon, "Tools and approaches for simplifying serious games development in educational settings," 2016 IEEE Global Engineering Education Conference (EDUCON), Abu Dhabi, 2016, pp. 1188-1197, doi: 10.1109/EDUCON.2016.7474707
- [8] Dorça, Fabiano & Araújo, Rafael & Carvalho, Vitor & Resende, Daniel & Cattelan, Renan. (2016). An Automatic and Dynamic Approach for Personalized Recommendation of Learning Objects Considering Students Learning Styles: An Experimental Analysis. *Informatics in Education.* 15. 45-62. 10.15388/infedu.2016.03.
- [9] Gallagher AG, Ritter EM, Champion H, Higgins G, Fried MP, Moses G, Smith CD, Satava RM. Virtual reality simulation for the operating room: proficiency-based training as a paradigm shift in surgical skills training. *Ann Surg.* 2005 Feb;241(2):364-72. doi: 10.1097/01.sla.0000151982.85062.80. PMID: 15650649; PMCID: PMC1356924.
- [10] Geronikolakis, E., Zikas, P., Kateros, S., Lydatakis, N., Georgiou, S.D., Kentros, M., & Papagiannakis, G. (2020). A True AR Authoring Tool for Interactive Virtual Museums. *Visual Computing for Cultural Heritage*.
- [11] Giannakos, Michail & Sharma, Kshitij & Pappas, Ilias & Kostakos, Vassilis & Velloso, Eduardo. (2019). Multimodal data as a means to understand the learning experience. *International Journal of Information Management.* 10.1016/j.ijinfomgt.2019.02.003.
- [12] K. Jarrett, K. Kavukcuoglu, M. Ranzato and Y. LeCun, "What is the best multi-stage architecture for object recognition?" 2009 IEEE 12th International Conference on Computer Vision, Kyoto,
- [13] Jenny, Seth & Keiper, Margaret & Taylor, Blake & Williams, Dylan & Gawrysiak, Joey & Manning, R. & Tutka, Patrick. (2018). eSports Venues: A New Sport Business Opportunity. *Journal of Applied Sport Management.* 10. 34-49. 10.18666/JASM-2018-V10-II-8469.
- [14] Hainey, Thomas & Connolly, Thomas & Chaudy, Yaelle & Boyle, Elizabeth & Beeby, Richard & Soflano, Mario. (2013). Assessment Integration in Serious Games. IGI Global. 10.4018/978-1-4666-4773-2.ch015.

- [15] Hallmann, Kirstin & Giel, Thomas. (2017). ESports - Competitive sports or recreational activity? Sport Management Review. 10.1016/j.smr.2017.07.011.
- [16] N. W. Hartman and G. R. Bertoline, "Spatial abilities and virtual technologies: examining the computer graphics learning environment," Ninth International Conference on Information Visualisation (IV'05), London, UK, 2005, pp. 992-997, doi: 10.1109/IV.2005.120.
- [17] Hooper, Jessica & Tsiridis, Eleftherios & Feng, James & Schwarzkopf, Ran & Waren, Daniel & Long, William & Poultsides, Lazaros & Macaulay, William & Papagiannakis, George & Kenanidis, Eustathios & Rodriguez, Eduardo & Slover, James & Egol, Kenneth & Phillips, Donna & Friedlander, Scott & Collins, Michael. (2019). Virtual Reality Simulation Facilitates Resident Training in Total Hip Arthroplasty: A Randomized Controlled Trial. The Journal of Arthroplasty. 34. 10.1016/j.arth.2019.04.002.
- [18] Huang, J., Yu, C., Wang, Y., Zhao, Y., Liu, S., Mo, C., Liu, J., Zhang, L., & Shi, Y. (2014). FOCUS: enhancing children's engagement in reading by using contextual BCI training sessions. CHI '14.
- [19] Hung, I-Chun & Lin, Lung-I & Fang, Wei-Chieh & Chen, Nian-Shing. (2014). Learning with the Body: An Embodiment-Based Learning Strategy Enhances Performance of Comprehending Fundamental Optics. Interacting with Computers. 26. 360-371. 10.1093/iwc/iwu011.
- [20] Kamarianakis, M.N., & Papagiannakis, G. (2020). Deform, Cut and Tear a skinned model using Conformal Geometric Algebra. ArXiv, abs/2007.04464.
- [21] Mik Kersten and Gail C. Murphy. 2006. Using task context to improve programmer productivity. In Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering (SIGSOFT '06/FSE-14). Association for Computing Machinery, New York, NY, USA, 1–11.
- [22] Konstantina Kiltani, Raphaela Groten, and Mel Slater. 2012. The sense of embodiment in virtual reality. Presence: Teleoper. Virtual Environ. 21, 4 (December 2012), 373–387. DOI: https://doi.org/10.1162/PRES_a_00124
- [23] Kurilovas, Eugenijus. (2018). Advanced machine learning approaches to personalise learning: learning analytics and decision making. Behaviour & Information Technology. 1-12. 10.1080/0144929X.2018.1539517.
- [24] Loh, Christian & Yanyan, Sheng & Ifenthaler, Dirk. (2015). Serious Games Analytics: Theoretical Framework. Springer. 10.1007/978-3-319-05834-4_1.
- [25] Lumsden, Jim & Skinner, Andy & Woods, Andy & Lawrence, Natalia & Munafò, Marcus. (2016). The effects of gamelike features and test location on cognitive test performance and participant enjoyment. PeerJ. 4. e2184. 10.7717/peerj.2184.
- [26] McMahan, Ryan & Lai, Chengyuan & Pal, Swaroop. (2016). Interaction Fidelity: The Uncanny Valley of Virtual Reality Interactions. 9740. 59-70. 10.1007/978-3-319-39907-2_6.
- [27] Monahan, Teresa & McArdle, Gavin & Bertolotto, Michela. (2008). Virtual reality for collaborative e-learning. Computers & Education. 50. 1339-1353. 10.1016/j.compedu.2006.12.008.
- [28] George Papagiannakis, Paul Zikas, Nick Lydatakis, Steve Kateros, Mike Kentros, Efstratios Geronikolakis, Manos Kamarianakis, Ioanna Kartsonaki, and Giannis Evangelou. 2020. MAGES 3.0: Tying the knot of medical VR. In ACM

SIGGRAPH 2020 Immersive Pavilion (SIGGRAPH '20). Association for Computing Machinery, New York, NY, USA, Article 6, 1–2. DOI: <https://doi.org/10.1145/3388536.3407888>

- [29] Petridis, P., Dunwell, I., Panzoli, D., Arnab, S., ... Protopsaltis, A. (2012). Game Engines Selection Framework for High-Fidelity Serious Applications. *International Journal of Interactive Worlds*, 1–19. doi:10.5171/2012.418638
- [30] Rizzo A', Shilling R. Clinical Virtual Reality tools to advance the prevention, assessment, and treatment of PTSD. *Eur J Psychotraumatol*. 2017;8(sup5):1414560. Published 2017 Jan 16. doi:10.1080/20008198.2017.1414560
- [31] Schmidt, Mona & Kowalewski, Karl-Friedrich & Schmidt, Marc & Wennberg, Erica & Garrow, Carly & Paik, Sang & Benner, Laura & Schijven, Marlies & Müller, Beat & Nickel, Felix. (2019). The Heidelberg VR Score: development and validation of a composite score for laparoscopic virtual reality training. *Surgical Endoscopy*. 33. 10.1007/s00464-018-6480-x.
- [32] Schmidt, M.W., Kowalewski, K., Trent, S.M. et al. Self-directed training with e-learning using the first-person perspective for laparoscopic suturing and knot tying: a randomised controlled trial. *Surg Endosc* 34, 869–879 (2020). <https://doi.org/10.1007/s00464-019-06842-7>
- [33] Schultze, U. Embodiment and presence in virtual worlds: a review. *J Inf Technol* 25, 434–449 (2010).
- [34] Shen, Jianqiang & Li, Lida & Dietterich, Thomas & Herlocker, Jon. (2006). A hybrid learning system for recognizing user tasks from desktop activities and email messages. 86-92. 10.1145/1111449.1111473.
- [35] Sloodmaker, Aad & Kurvers, Hub & Hummel, Hans & Koper, Rob. (2014). Developing Scenario-based Serious Games for Complex Cognitive Skills Acquisition: Design, Development and Evaluation of the EMERGO Platform. *JOURNAL OF UNIVERSAL COMPUTER SCIENCE*. 20. 561-582.
- [36] Sloodmaker, A., Nadolski, R., Kurvers, H. J., Hummel, H. G. K., & Koper, E. J. R. (2018). Usability of the EMERGO player environment for scenario-based serious games. Manuscript submitted for publication.
- [37] Spikol, Daniel & Ruffaldi, Emanuele & Dabisias, Giacomo & Cukurova, Mutlu. (2018). Supervised machine learning in multimodal learning analytics for estimating success in project-based learning. *Journal of Computer Assisted Learning*. 34. 10.1111/jcal.12263.
- [38] Tölzer, Christine & Gupta, Kapil & Yadav, Sathish & Borucu, Ufuk & Garzoni, Frederic & Staufer, Oskar & Capin, Julien & Spatz, Joachim & Fitzgerald, Daniel & Berger, Imre & Schaffitzel, Christiane. (2020). Unexpected free fatty acid binding pocket in the cryo-EM structure of SARS-CoV-2 spike protein. 10.1101/2020.06.18.158584.
- [39] Tsiridis E. *The Adult Hip – Master Case Series and Techniques*. Cham: Springer, 2018.
- [40] Zikas, P., Papagiannakis, G., Lydatakis, N., Kateros, S., Ntoa, S., Adami, I., & Stephanidis, C. (2020). Scenior: An Immersive Visual Scripting system based on VR Software Design Patterns for Experiential Training. arXiv: Graphics.

Appendix 1 – Analytics Assessment

Below the AnalyticsManager code is presented. This manager is responsible for initializing and finalizing the scoring factors, as well as storing their results. In addition, the feedback panels are managed by this class.

```
public class AnalyticsManager : MonoBehaviour
{
    //Singleton fo easier access from other scripts.
    private static AnalyticsManager _instance;
    public static AnalyticsManager Get{...}

    //Dictionary storing the score mapped to each task
    public Dictionary<string,float> results; <# Serializable

    //Tmp variables
    private float _currentScore;
    private float _timer;

    //Initialize Manager
    public void Start()
    {
        results = new Dictionary<string,float>();
        _timer = 0;
    }

    //Function called on every frame calculating time.
    public void Update()
    {
        _timer += Time.deltaTime;
    }

    //Initialize Action
    public IEnumerator InitializeAction()
    {
        //Wait one frame for the task to be initialized.
        yield return new WaitForEndOfFrame();

        //Get scoringFactors for this task.
        ActionData currentScoringFactors = ActionsController.Get.GetCurrentScoringFactors();
        //Initiliza them
        foreach(ScoringFactor sF in currentScoringFactors.ScoringFactors)
        {
            sF.Initialize();
        }
        _timer = 0;
    }

    //Finish Action
    public void Perform(bool skipped = false)
    {
        //Init Variables
        float result = 100;
        string actionName = ActionsController.Get.GetCurrentActionName();
        ActionData currentScoringFactors = ActionsController.Get.GetCurrentScoringFactors();
```

```

//Reset Canvas Showing all scores.
var detailsCanvas :GameObject = GameObject.Find("DetailsCanvas");
detailsCanvas.GetComponent<HandleFactors>().ResetUI();

//For each scoring factor
for (int i=0;i<currentScoringFactors.ScoringFactors.Count;i++)
{
    float factor = GetWeightFromEnum(currentScoringFactors.ScoringFactors[i].sFactorImportance);
    //Calculate deducted points multiplied by factor.
    float deductedPoints = (100-currentScoringFactors.ScoringFactors[i].Perform())*factor;
    result -= deductedPoints;
    //Calculate deducted points multiplied by factor.
    currentScoringFactors.numErrors += currentScoringFactors.ScoringFactors[i].numErrors;

    //Show all factors explanations.
    detailsCanvas.GetComponent<HandleFactors>().UpdateUI((ScoringFactorData) currentScoringFactors.ScoringFactors[i].GetReadableData(),i);
}

result = Mathf.Clamp( value: result, min: 0f, max: 100f);
//Add results to action's data panel.
results.Add(actionName, result);

//Show all factors explanations.
var scoreCanvas :GameObject = GameObject.Find("ScoreCanvas");
scoreCanvas.transform.Find("ErrorsValue").GetComponent<Text>().text = currentScoringFactors.numErrors.ToString();
scoreCanvas.transform.Find("MultiplierValue").GetComponent<Text>().text = currentScoringFactors.multiplier.ToString();
scoreCanvas.transform.Find("ScoreValue").GetComponent<Text>().text = result.ToString(CultureInfo.InvariantCulture);
scoreCanvas.transform.Find("TimeValue").GetComponent<Text>().text = _timer.ToString( format: "F1");
}

//Function returning multiplier from enum.
private float GetWeightFromEnum(FactorImportance enumWeight){
    switch (enumWeight)
    {
        case FactorImportance.Trivial:
            return 0.15f;
        case FactorImportance.Small:
            return 0.3f;
        case FactorImportance.Average:
            return 0.5f;
        case FactorImportance.Big:
            return 0.8f;
        case FactorImportance.Significant:
            return 1f;
        default:
            throw new ArgumentOutOfRangeException(nameof(enumWeight), enumWeight, message: null);
    }
}

```

Below the class base class of scoring factors is presented as well as the ScoringFactorData struct which serves as a generic descriptive struct for all scoring factors.

```
public enum FactorImportance{Trivial, Small,Average, Big, Significant }

public class ScoringFactor : IScoringFactor
{
    //Error Message shown to users
    public string ErrorMessage;
    //Importance Weight of this Scoring Factor
    public FactorImportance SFactorImportance;
    //Score of this Scoring Factor.
    protected int Score;
    //Variables used for the calculation of the score.
    public int FactorSpecific;
    protected int FactorTarget;
    //Descriptive name of the scoring factor.
    protected string Type;
    //If enabled an error UI will be shown when the event is triggered.
    public bool ShowErrorUi;
    //Times the error of this scoring factor was triggered.
    public int NumErrors;

    public void LogError()
    {
        NumErrors++;
        if (ShowErrorUi)
        {
            UISpawner.Get.SpawnErrorUI(ErrorMessage);
        }
    }
    public object GetReadableData()
    {
        ScoringFactorData data;

        data.scoreSpecific = FactorSpecific;
        data.outOF = FactorTarget;
        data.errorMessage = ErrorMessage;
        data.type = Type;
        data.score = Score;
        data.importance = SFactorImportance;
        return data;
    }

    public void Initialize(){...}
    public float Perform(){...}
    public void Undo(){...}
}
```

Appendix 2 – VR Session Logging

Below is the code of the PropagateLogging script is shown. This script is attached to each object that is currently logged except the virtual hands and the camera.

```
//Variable used only on the stress test demo.
public bool doNotDestroy;

//The depth level of this script. How many other objects did it pass before being
//attached to this one?
private int _level;
//Previous position and angle used for detecting if the object has stopped moving.
private Vector3 _prevPosition;
private Vector3 _prevAngles;
//Transformation difference tolerance for ending logging.
private const float EndTolerance = 0.01f;
//Used for the color change effect.
private Color oldColor;
//The originating hand.
private LoggerManager.TrackingTarget _target;
//Time counter used for computing when to stop logging the transformation of this object.
private float _destroyCounter;

//Init function for setting the depth level of this script as well as the original
//material.
public void SetLevel(int level, LoggerManager.TrackingTarget target)
{
    _level = level;
    var MR = GetComponent<MeshRenderer>();
    if (!MR) return;

    var material = MR.material;
    oldColor = material.color;
    material.color = Color.green;
    _target = target;
}

//Collision Detection function which adds the propagating script to the collided object.
public void OnCollisionStay(Collision other)
{
    //Skip static objects.
    if (other.gameObject.isStatic)
    {
        return;
    }

    LoggerManager.Instance.IncreaseLoggingCount();
    //Add it only if it doesn't already have it.
    var target = other.transform.root.gameObject;
    if (!target.GetComponent<PropagateLogging>())
        target.AddComponent<PropagateLogging>().SetLevel(_level+1, _target);
}

//Function called on every frame responsible for deleting the object.
public void Update()
{
    //Objects which are directly held are logged through the InteractionLoggerImproved
    //script.
    if (_level == 0) return;
```

```

//Check transformation difference from last frame in case we need to delete this.
var transform1 = transform;
if (CompareVector3(_prevAngles,transform1.eulerAngles,EndTolerance) &&
    CompareVector3(_prevPosition,transform1.position,EndTolerance))
{
    _destroyCounter += Time.deltaTime;
    if (_destroyCounter > 1 && doNotDestroy==false)
    {
        Destroy(this);
    }
}
else
{
    _destroyCounter = 0;
}
_prevPosition = transform1.position;
_prevAngles= transform1.eulerAngles;
}

//Function called on every frame AFTER the physics engine has finished its calculations.
//It's responsible for logging the object's transformation.
public void LateUpdate()
{
    var transformRef = transform;
    var trans = transformRef.position;
    var rot = transformRef.rotation.eulerAngles;
    var content = " "+gameObject.name + " " + trans.x + " " + trans.y + " " + trans.z + " " +
rot.x + " " + rot.y + " " + rot.z;

    LogManager.Instance.WriteToFile(_target,content);
}

//Function comparing two vectors and returns true when the script must be destroyed.
private bool CompareVector3(Vector3 a, Vector3 b,float tolerance)
{
    var dx = a.x - b.x;
    var dy = a.y - b.y;
    var dz = a.z - b.z;

    if (Mathf.Abs(dx) > tolerance) return false;
    if (Mathf.Abs(dy) > tolerance) return false;
    if (Mathf.Abs(dz) > tolerance) return false;

    return true;
}

//Function called when this script is destroyed. It restores the original material.
public void OnDestroy()
{
    var MR = GetComponent<MeshRenderer>();
    if (!MR) return;
    var material = MR.material;
    material.color = oldColor;

    MR.material = material;
}
}

```


Appendix 3 – Supervised ML Functions

Below our functions for building our model and fitting it to our training data are presented. Only the *multi-view* implementation is shown here.

```
def submodel(x):
    #Each camera model has 3 convolution layers each with its corresponding max pooling
    #layer.
    l1 = Conv2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu')(x)
    l2 = MaxPooling2D(pool_size=(2, 2), strides=2, padding='same')(l1)
    l3 = Conv2D(filters=128, kernel_size=(3, 3), padding='same', activation='relu')(l2)
    l4 = MaxPooling2D(pool_size=(2, 2), strides=2, padding='same')(l3)
    l5 = Conv2D(filters=256, kernel_size=(3, 3), padding='same', activation='relu')(l4)
    l6 = MaxPooling2D(pool_size=(2, 2), strides=2, padding='same')(l5)

    return l2
```

```
def build_model():
    #Defining input shape of model for each camera.
    x0 = keras.Input(shape=(basewidth, basewidth, 3), name='cam0')
    x1 = keras.Input(shape=(basewidth, basewidth, 3), name='cam1')
    x2 = keras.Input(shape=(basewidth, basewidth, 3), name='cam2')
    x3 = keras.Input(shape=(basewidth, basewidth, 3), name='cam3')

    #Create models with all 3 convolutional layers for each camera.
    m0 = submodel(x0)
    m1 = submodel(x1)
    m2 = submodel(x2)
    m3 = submodel(x3)

    #Concatenate the models to the final CNN.
    x = keras.layers.concatenate([m0, m1, m2, m3])

    #Add sequentially flatten and dense layers.
    l1 = Flatten()(x)
    l2 = Dense(1024, activation='relu')(l1)
    l3 = Dropout(0.5)(l2)
    l4 = Dense(3, activation='softmax')(l3)

    #Create the final model specifying inputs and outputs
    model = keras.Model(inputs=[x0, x1, x2, x3],
                        outputs=[l4])

    return model
```

```
def train(model, x_train, y_train, x_test, y_test, x_val, y_val, i):
    #Parameters epoch, batch_size and optimizer are global.
    #Optimize our model using crossentropy and the corresponding optimizer.
    if (optimizer == "Adadelta"):
```

```

model.compile(    loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adadelta(),
                  metrics=['accuracy'])

if (optimizer == "SGD"):
    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.sgd(),
                  metrics=['accuracy'])

#Fiting the model to our training and validation data.
history = model.fit({'cam0': x0_train, 'cam1': x1_train,
                    'cam2': x2_train, 'cam3': x3_train},
                    y_train,
                    batch_size=batch_size,
                    epochs=epochs, verbose=True,
                    validation_split=0.1)

#Testing our model on the test dataset.
result = model.evaluate(x_test, y_test)

```