DESIGN OF A VIRTUAL REALITY SCENARIO AND SCENT GENERATOR FOR SENSORY TRAINING

A Thesis
Presented to
the Faculty of the Elmer R. Smith College of Business and Technology
Morehead State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Yosselin Castro Islas
November 15, 2019
Accepted by the faculty of the Elmer R. Smith College of Business and Technology, Morehead State University, in partial fulfillment of the requirements for the Master of Science degree.

____________________________________
Dr. Jorge Alberto Ortega Moody
Director of Thesis

Master’s Committee: ____________________________, Chair
Dr. Ahmad Zargari

____________________________________
Dr. Nilesh Joshi

____________________________________
Date
In the food industry, what makes the difference between a company and its competitors is the quality of the food product, and one of the components that makes quality is the flavor of a product. In order to be successful, the company will have to make sure their customers know what makes them unique and different. Trends in the food industry will continue to shift to meet consumers’ health and wellness desires, and this leads to the creation of new formulas that try to maintain the same taste but with different ingredients. The key to being competitive in the market and remaining close to consumer’s desires is keeping the same flavor of the food product or even enhancing the experience of it, this is one of the main reasons why sensory analysis in the food industry and in food science has continuously increased and has gained more terrain.

Some of the limitations for sensory analysis is the required space to have the individual booths, time consuming preparation, and material costs. Even when examining the previous limitations, one of the most important is the training of new users and retraining of existing
users. The environment created to analyze a certain food could affect the results, either positively or negatively, and having false results from skewed data would then cost the company time and money. Virtual Reality can be used to create a detailed scenario for users or employees that requires sensory training to be part of the trained panel, and as well as to mentally and physically learn the tasks required and allow companies to introduce people with different needs and learning styles. The potential of a Virtual Scenario relies on activating all senses and immersion in a holistic experience. The main objective of this research covers the development of a virtual scenario for sensory training, by visualizing a standard booth and creating a Virtual Scenario training program to analyze scents using a scent generator prototype.

The methodology is divided in steps, the first step starts with the design of the sensory training using a discrimination test and the interaction with the subject in the virtual scenario. Secondly, designing the cad parts for the virtual scenario using a 3D modeling software for sensory training with standard features including environment and materials. The third step is focused on the integration of the laboratory layout and the sensory training in the virtual scenario that will allow the subject to feel immersed. In the fourth step the programming take place to create the interface to have interactions between the environment, the human, and the scent generator used to release the scents. The fifth and final step is designing a scent generator prototype for scents released based on a 3D model printing.

Accepted by: ________________________________, Chair
Dr. Ahmad Zargari

______________________________
Dr. Jorge Alberto Ortega Moody

______________________________
Dr. Nilesh Joshi
ACKNOWLEDGEMENT

I would like to express the deepest appreciation to my committee chair, Dr. Ahmad Zargari, who had support me and giving me the best advices during my stay in the master’s degree for my professional development. Next, I would like to thank my thesis director, Dr. Jorge Alberto Ortega Moody, who believed in me during this challenge. Without his guidance, persistent help and patience this project would not have been possible. I would like also to thank my committee member, Dr. Nilesh Joshi, who provided his guidance through the courses that he teaches and support during my professional development.

In addition, a thank to those individuals who aided and supported in completing this thesis, specially Levi Howell who during his undergraduate fellowship assisted and shared his experience in this project. Also, thank the Craft Academy members, who accepted me as Resident Counselor and with whom I spent a lot of time and life experience during the last year.

I would like to thank my family, for being always next to me no matter how far I am they are always supporting me and sending me their best wishes and encourages through my life. I love them forever.

Lastly, I would like to thank Morehead State University members and friends who make easier my everyday life and gave me great memories and personal growth.
Table of Contents

Chapter 1 Introduction ............................................................................................................. 1
  1.1 Definitions......................................................................................................................... 1
  1.2 General background ......................................................................................................... 3
  1.3 Purpose of the Study ....................................................................................................... 3
  1.4 Objectives ....................................................................................................................... 5

Chapter 2 Literature Review ................................................................................................. 6
  2.1 Sensory Analysis............................................................................................................... 6
  2.2 Virtual Reality ................................................................................................................ 8
  2.3 Scent Generator ............................................................................................................. 11

Chapter 3 Methodology ........................................................................................................ 14
  3.1 Conceptual Design ......................................................................................................... 14
  3.2 Design the Training Specifications ............................................................................... 15
  3.3 Virtual Reality System .................................................................................................. 15
    3.3.1 3D Modeling ............................................................................................................. 15
    3.3.2 Mesh Simplification ................................................................................................. 16
    3.3.3 Texture .................................................................................................................... 20
    3.3.4 Lights and Sound Effects ....................................................................................... 22
    3.3.5 Colliders and Rigid Bodies ..................................................................................... 23
    3.3.6 Scripting Programming *Unity* ............................................................................. 27
  3.4 Scent Generator Design ................................................................................................ 31
    3.4.1 Scripting *Arduino* Programming ......................................................................... 32

Chapter 4 Findings and Results ............................................................................................ 34
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Scent Generator</td>
<td>34</td>
</tr>
<tr>
<td>4.2 Virtual Reality Scenario Design</td>
<td>37</td>
</tr>
<tr>
<td>Chapter 5 Summary, Conclusions and Further Work</td>
<td>42</td>
</tr>
<tr>
<td>5.1 Summary</td>
<td>42</td>
</tr>
<tr>
<td>5.2 Conclusions</td>
<td>42</td>
</tr>
<tr>
<td>5.3 Further work</td>
<td>43</td>
</tr>
<tr>
<td>References</td>
<td>44</td>
</tr>
<tr>
<td>Appendix A</td>
<td>48</td>
</tr>
<tr>
<td>Appendix B</td>
<td>62</td>
</tr>
<tr>
<td>Appendix C</td>
<td>65</td>
</tr>
<tr>
<td>Appendix D</td>
<td>85</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1. MetaCookie+: Flavor display based on cross-modal interaction ........................................ 9
Figure 2. Wearable Olfactory Display. (Carulli et al., 2016) .............................................................. 10
Figure 3. The Essence necklace. ........................................................................................................ 11
Figure 4. Odor-Presenting Unit .......................................................................................................... 12
Figure 5. Multi-Fragrance Olfactory Display .................................................................................... 13
Figure 6. Sensory Training Diagram. ................................................................................................. 14
Figure 7. Surface Mesh Simplification. (Cacciola, 2010) ................................................................. 16
Figure 8. Mesh Simplification Diagram. (Samudio, 2018) ................................................................. 17
Figure 9. Visual aspect of a 100% mesh simplification default shading and its wireframe override ................................................................. 18
Figure 10. Visual aspect of a 60% mesh default shading and its wireframe override ......................... 19
Figure 11. Visual aspect of a 10% mesh default shading and its wireframe override ......................... 19
Figure 12. Solid Colors for Texture ................................................................................................. 20
Figure 13. Metal Images for Texture ............................................................................................... 20
Figure 14. Applied texture to the floor .............................................................................................. 21
Figure 15. Applied texture to the table ............................................................................................. 21
Figure 16. Lighting specifications for the spotlight .............................................................................. 22
Figure 17. Box Colliders and Rigid bodies properties in the Sensory Booth .................................... 24
Figure 18. Gravity and kinematic properties with box colliders in the tray ...................................... 25
Figure 19. Grabbable script, box collider and rigid body properties for the cup ............................... 25
Figure 20. Oculus integration with Unity .......................................................................................... 26
Figure 21. Smell scripting shown in inspector toolbar from Unity .................................................... 28
Figure 22. Sequence scripting in inspector toolbar from *Unity*. ......................................................... 29

Figure 23. Scent Generator Design ............................................................................................................ 31

Figure 24. The architecture of the Scent Generator .................................................................................... 32

Figure 25. Scent Generator Prototype ....................................................................................................... 34

Figure 26. *Arduino* Uno and Relay Integration. ......................................................................................... 35

Figure 27. Multiple Scents Connection. ....................................................................................................... 35

Figure 28. Integration of Scent Generator ................................................................................................... 36

Figure 29. Virtual Reality Scenario Case 1. ................................................................................................. 37

Figure 30. Virtual Reality Scenario Case 2. ................................................................................................. 38

Figure 31. Virtual Reality Scenario Case 3. ................................................................................................. 39

Figure 32. Virtual Reality Scenario Case 4. ................................................................................................. 39

Figure 33. Virtual Reality Scenario Case 5. ................................................................................................. 40

Figure 34. User Immersed in the Training ................................................................................................. 41

Figure 35. User Immersed in the Training ................................................................................................. 41

Figure 37. Integration of Virtual Reality in Sensory Training ..................................................................... 57

Figure 38. Costs of the 3D printed prototype. ............................................................................................. 57

Figure 39. Elements of the Scent Generator ............................................................................................... 58

Figure 40. State machine for the Sensory Training program. ................................................................... 58

Figure 41. Case 1 shown on game with audio scripts ................................................................................. 59

Figure 42. Case 2 shown on game with audio scripts ................................................................................. 59

Figure 43. Case 3 shown on game with audio scripts ................................................................................. 60

Figure 44. Case 4 shown on game with audio scripts ................................................................................. 60

Figure 45. Case 5 shown on game with audio scripts ................................................................................. 61
Figure 46. Case 6 shown on game with audio scripts............................................................................. 61
Chapter 1 Introduction

1.1 Definitions

The Institute of Food Technologists (IFT) established that “sensory evaluation is a scientific discipline that is used to evoke, measure, analyze and interpret those responses to products as perceived through the sense of sight, smell, hearing, touch and taste” (Stone & Sidel, 1993; IFT, 2007).

Using sensory evaluation one can understand how humans perceive and respond to various stimuli in food. Understanding the complex nature of sensory perception and consumer behavior is applied as a quality tool in the product development process. The information is used to recognize the main consumer drivers that will influence the decision making process.

The odor of a product is detected when its aromatic compounds enter the nasal passage and are perceived by the olfactometry system. Odor refers to when the aromatic compounds are sniffed through the nose, either voluntarily or otherwise. Aroma is the odor of a food product, and fragrance is the odor of a perfume or cosmetic (Meilgaard, Vance, & Thomas, 1999). In this research we will refer to the aromatic compounds used as scents.

The sense of smell is employed daily and is concerned with making sensory memory, identifying chemical signals from one’s surroundings, such as danger triggers, favorite food, as well as correlating smells to a feeling or an experience. Regarding the olfaction process, scent detection involves many regions of the brain in which both conscious and unconscious activities are carried out (Alaoui-Ismaili, Vernet-Maury, Dittmar, Delhomme & Chanel, 1997). In addition, scents can enhance learning activities, increase the level of attention, and are deeply evocative (Narumi, Nishizaka, Kajinami, Tanikawa & Hirose, 2011).
Virtual environments or virtual scenarios are custom-made 3D virtual worlds created out of 3D models, which allows users to interact and immerse themselves in “worlds” that are distant, expensive, hazardous, or inaccessible. Virtual scenarios are also known as, “artificial reality”, “virtual worlds” and “synthetic environments” (Alaraj et al., 2011).

Research by Zaho (2002) defined Virtual Reality as “a closed computer system that consists of a virtual environment, a physical environment, as well as a software and hardware interface, which allows interaction between a human and a computer.”

In their book, Sherman and Craig (2003), related to the virtual reality as a medium composed of interactive computer simulations that sense the participant’s position and actions and replace or augment the feedback to one or more senses, giving the feeling of being mentally immersed or present in the simulation. This definition comes from four key elements that the subject experiences in virtual reality, which are: a virtual world, immersion, sensory feedback, and interactivity.

The engine Unity is self-described as “a flexible and powerful development platform for creating multi-platform 3D and 2D games and interactive experiences”. The engine allows for quick testing and has many well documented instructional resources and tutorials in C#. It uses a component-based methodology, which involves the use of renders, scripts, and controllers to create game objects (Blum, et al., 2016).

Arduino is an open-source microcontroller that is known for being easy to program and is used of easy interaction between hardware and software. These factors create a friendly interface in which any person or student, even those without a background in electronics and programming, can utilize based on inputs and outputs (Hughes, 2016).
1.2 General background

Sensory analysis is based on average data obtained by either trained users or consumers for evaluating food products under controlled conditions in a sensory laboratory. Traditionally individual booths with specifically controlled features such as; colors, light, and other aspects of the environment are used. Due to the increasing market and competitors, the need for sensory analysis is becoming increased in food science for developing new products and in industry for developing higher quality products while also reducing costs. Sensory analysis is being used as a powerful tool in quality control assuring customer satisfaction.

Advances in technology have evolved to include interface systems which allow for the introduction of virtual reality and virtual scenarios into food science. Research on this field is becoming vaster as a rich context for marketing consumer evaluations in virtual reality environments are being used, such as coffee shops, bars, and laboratories.

1.3 Purpose of the Study

The application of this technology in the food science will continue to grow. In turn, the sensory science field will have a revolution of their basic understandings of multisensory experiences with immersive controlled environments that can be customized to meet demands using CAD.

Sensory analysis has been developed in a traditional laboratory using an individual booth, with controlled conditions to evaluate the consumer test preference through conscious responses, but the results have not always been positive. In the research by De Wijk, Kooijman, Verhoeven, Holthuysen and de Graaf (2012), they proposed that consumer preference will be conducted based on unconscious decision making which involved behavior responses to the product.
For consumption preference tests, creating different environments such as; controlled laboratory environments, bars, and restaurants is highly impactful on the research to analyze responses and gather sensory feedback when evaluating the same product.

As Crofton, Botinestean, Fenelon and Gallagher (2019) mentioned in their article, recreating an immersive environment has an ecological advantage, reducing the relatively large amount of space required, when reduces’ expenses and time-consumption.

The emergence of virtual reality technology has created many opportunities to improve in the sensory discipline. The purpose of this project is to create a laboratory in a Virtual Scenario with individual booths to have the same criteria for training, in an immersive environment, that will allow the subject to feel the experience of the traditional method. To achieve this, a small scent generator was made to give the advantage of portable sensory training. Utilizing a virtual scenario results in reducing time-consuming tasks, standardizing the scenario to repeat the training as needed, reducing the cost of installation, and making an ecological test by reducing waste material.
1.4 Objectives

The main objectives of this research are as follows:

➢ Design the sensory training based on sensory discrimination tests.
➢ Design the interaction for sensory training that will be recreated in the Virtual Scenario.
➢ Design the CAD parts of the Virtual Scenario using 3D modeling software.
➢ Integrate the laboratory layout and the sensory training in the Virtual Scenario using *Unity* when allows the user to feel immersed.
➢ Design the programming for sensory training using C# through *Unity* software to have interaction between the environment and the user.
➢ Program the scent generator prototype using an *Arduino* to release the scents using C# programming.
➢ Design the scent generator prototype for scent release using *SolidWorks* 3D model printing.
Chapter 2 Literature Review

2.1 Sensory Analysis

Sensory testing normally is conducted in a traditional laboratory using individual booths to keep focus on the food product and reduce interaction with people and surroundings. When individuals consume the product, it could be in their home, a restaurant, a cafeteria or an airplane, which can cause a difference in the results by not doing in the traditional way.

There is some research about how eating in different environments influences the consumer’s the perception and acceptability of the products. Research by Xu, Hamid, Shepherd, Kantono and Spence (2019), demonstrated the influence of the environment in affecting the consumers’ hedonic, emotional, and electrophysiological responses while consuming chocolate ice cream in a laboratory, cafeteria, university area, and a city bus stop. In another study, it was found that consuming food in a traditional laboratory while using individual booths, showed that the food presented had less hedonic quality compared to the same food consumed in a different environment (Meiselman, Johnson, Reeve, and Crouch 2000). A study conducted by Garcia-Segovia, Harrington and Seo (2015) used two contrasting environments consuming identical food in a restaurant. One environment is surrounded by a gourmet table against to a booth environment with a plastic tray. Yielded results favorable responses to the appearance of food consumed in a gourmet table versus in a sensory testing booth.

Sensory analysis has multiple applications not only on the scientific field, but also in marketing and industry, the principal applications are quality control, product development, environment odor detection, evaluation on food and beverages, pharmaceutical field, and cosmetics.
The sense of smell is being used for humans to interpret aromas from the environment to identify danger, food or pleasant experiences and in the past has helped enhance survival instincts. Scientific research has been done involving the connection between emotions in response to aromas, like Alaoui-Ismaili et al. (1997), where they study the relationship between emotional generation in response to aromatic compounds using the sense of smell.

There is some research in the marketing area that describes the link between sensations and consumer perception to customer preference in the market. Aradhna Krishna (2011), defined “sensory market” as marketing that engages the consumers’ senses and affects their perception, judgment and behavior. Humans can correlate an experience to a smell. From a marketing point of view, aromas can be employed to evoke positive experiences with products used by consumers to gain company’s preference in future purchases.

Sensory analysis requires certain sensitivity in humans to be able to detect and to identify compounds, normally research is conducted with a trained panel or expertise panel. The results of untrained panels will require increased concentrations to enhance signal detection. Research was conducted by Dalton, Doolittle and Breslin (2002) on enhanced sensitivity to aromatic compounds developing the ability after repeated exposures of a volatile compound. These results suggest that untrained people can acquire olfactory sensitivity by repeated exposure to a compound or smell.

In sensory analysis discrimination tests are one of the main tests. Discrimination tests have two scopes, they are used in sensory analysis to identify that there is a difference between samples to recognize that there is no difference between the samples. One type of discrimination test is the Triangle test, where the main scope is to recognize if a difference exists between two samples. The larger number of subjects used in a Triangle test yields better results, although a
small number of subjects can give important results when the difference is easier to detect. The procedure consists of presenting three samples, where two of them are the same sample and one is different. The subject is asked to identify which of the three samples is the odd sample. The samples are coded in a three-digit number arrangement and presented in a random order: AAB, ABA, BAA, BBA, BAB, and ABB. The sample that is shown once in each random order is the different sample.

Food perception is a multisensory experience that involves the five senses and the feedback from the different surrounding environments plays a principal role in generating stimuli that a normal laboratory cannot create in front of a booth. Although, there is more research needed in the sensory field between the perception of a product and the interaction in an immersive environment, there are some results obtained in real life experiments changing the conditions and the influence that it has when perception and liking products are of concern.

Research by Kennedy, Stewart-Knox, Mitchell and Thutnham (2004) showed that testing conditions had an effective response even with an untrained panel, when testing in a laboratory or testing at home. The results showed that testing under home conditions can enhance the perception of the same product, even though the composition of the product did not change.

2.2 Virtual Reality

Introducing a multisensory experience in a virtual reality environment could evoke and increase the feeling of realism while working in an immersive scenario. These scenarios seem to be working or learning in a real-life experience, with the advantages of having controlled variables and more widely it available for more people.

In their book Sherman and Craig (2003), mentioned one of the four key elements is the sensory feedback as an ingredient essential to virtual reality. In most virtual scenarios the visual
sense is the only sense that receives feedback, though virtual environments also exist that display exclusively haptic (tough) experiences. Researchers have demonstrated that the sense of smell has an important presence in the feedback when being immersed in a virtual medium or environment, and this is possibly due to the new technology and development.

Scientific research has been done involving the sensory approach through the five senses and their reaction in front of stimuli in an immersive environment. Research by Narumi, Nishizaka, Kajinami, Tanikawa and Hirose (2011), they built a “meta cookie+” based on an illusion (Figure 1) and changed the appearance and scent of the cookie using virtual reality without doing any change to the chemical composition. They proposed that changing the appearance and scent of the food product can impact on the perception. They define MetaCookie as a “flavor augmentation system that enables changes that alter the perceived taste of a cookie by overlapping visual and olfactory information onto a real cookie."

Figure 1. MetaCookie+: Flavor display based on cross-modal interaction among vision, olfaction and gustation. (Narumi et al., 2011)
Research by Carulli, Bordegoni and Cugini (2016) Figure 2 proposed that introducing scents simulation in virtual reality environments can enhance the user’s sense of presence and concede analysis of the impact through scents while judging products.

![Wearable Olfactory Display](image)

Figure 2. Wearable Olfactory Display. (Carulli et al., 2016)

The wearable olfactory display seen in Figure 2 consists of air cannons that are placed on commercial earphones, which used the ultrasonic atomization method to generate the scents. This method contain a small cylinder of porous cotton as a cartridge and, under the effect of ultrasonic device, a scent mist is produced and released to the user’s nose. This application works through a tablet sends inputs to an Arduino board and sends the command to the air cannons.

Virtual Reality has endless applications depending on the needs that arise, Virtual Reality proposes a method to solve or approach situations that initially seemed economically unattainable and even proposes to carry out activities with greater security and practicality. In neurosurgical medicine, the use of virtual reality helped to train surgeons providing an environment as close to reality to feel immersed in a real operating room without the high cost and reducing the legal and ethical concerns by practicing with patients (Alaraj et al., 2011).
2.3 Scent Generator

In the scientific field the term olfactometers or dynamic sniffer refers to a device designed to generate odors controlled by computer with a determined flow, concentration, frequency and standardized parameters, normally used in the laboratory for research. Generally, because of the size and weight are not comfortable to use it and are not portable.

One example of a transportable and wearable device is the Essence necklace (Figure 3) design by Amores and Maes (2017), which is controlled remotely by a smartphone to release scent using a piezoelectric transducer that can be used in everyday life situations. The release of the scent will be based on contextual data from the user. Its limitations was that it can release one scent although changing scents depends on the user, but it needs to be done one at a time.

Figure 3. The Essence necklace. (Amores and Maes, 2017)
The Essence necklace (Figure 3) consists of a 3D printer cover (A) that holds the piezoelectric on top of the cotton stick filter, soaking up the fragrance from the container. Once the fragrance is depleted, the user can easily unscrew the cap and refill the container. The back part of the necklace (B) contains the micro-controller and the rest of the electronics are to control the release of scent (Amores and Maes, 2017).

Another wearable device studied by Yamada, Yokoyama, Tanikawa, Koichi and Hirose (2006), was an odor unit that was housed and carried in a backpack with smells being conveyed through tubes to the user’s nose. The prototype is shown in the next page with the details of the elements.

![Figure 4. Odor-Presenting Unit. (Yamada et al., 2006)](image)

The wearable olfactory display (Figure 4) proposed by Yamada, Yokoyama, Tanikawa, Koichi and Hirose, (2006) consists of an odor-generating unit, an odor-controlling unit, and an odor-presenting unit. The odor-generating unit consists of air pumps and odor filters. The odor-controlling system consists of a PC and a device controller. The odor-presenting unit consists of a system to release the odor as close as possible to the user’s nose.
Previous work has been studied using a computer-controlled device that release scents in a virtual environment, this device is named olfactory display (Nakamoto, 2013). Unfortunately, the prototypes trying to release a great number of scents makes it cumbersome and not a very pleasant experience while being used as a headset.

A research project performed by Risso, Covarrubias, Bordegoni and Gallace (2018), worked with a portable olfactory device called MFOD (Multi-Fragrance Olfactory Display) that released eight fragrances in a controlled manner through airflow from solid fragrance releases. They evaluated how impactful odors generated by the MFOD were on user’s taste perception.

![Concept of the MFOD](image)

Figure 5. Multi-Fragrance Olfactory Display.
(Risso et al., 2018)

The MFDO (Figure 5) consists of a dispenser able to display eight fragrances that includes a centrifugal fan (1) to control the air flow, a servo-motor (2), a fragrance or cylindrical repository (3), the eight solid fragrances and a pipe (5) where the airflow goes and passes through and delivers the fragrance close to the user’s nose.
Chapter 3 Methodology

3.1 Conceptual Design

The evolution of the sensory training application requires the following sequential steps, each step acts of important implementations for the final design (Figure 6).

Figure 6. Sensory Training Diagram.
3.2 Design the Training Specifications

The first step of the development process requires the design specifications of the training area. These cases consist of a tutorial and a test evaluation for the subject displaying on a screen of the sensory booth where the evaluation will take place. Figure 6 it shows the flow that will take place starting with the welcoming, then the virtual scenario for sensory training, followed by an introduction of the environment, and lastly the subject will be asked to choose between the tutorial or the test option. In the tutorial, an introduction will be displayed explaining the nature of the logistics with the five scents that will be released through the scent generator. The user will continue the procedure until the identification of the five scents is complete. If the user needs to retry a scent in order to be able to completely identify them, they will be allowed to retake the tutorial. When the user finishes with the identification process, they will be referred to the test evaluation. In the test evaluation, an introduction to the evaluation is performed, and the user will have to identify the odd sample. The result will display on the screen when the test is done.

3.3 Virtual Reality System

3.3.1 3D Modeling

In order to create the immersive environment, this researcher utilized SolidWorks, a solid modeling computer aided design (CAD) and computer aided engineering (CAE) software. In the development of the 3D environment the first step was designing the CAD drawings using SolidWorks. Several of the 3D models that were used throughout this research were downloaded from different free cloud-based websites. The 3D models that were not available through online cloud-based websites were created by the researcher.
3.3.2 Mesh Simplification

The simplification process starts with a 3D design in SolidWorks and then is imported to the Autodesk 3D Studio Max software from a file type ending with the extension SLDprt, SLDasm, STL or STP and from this point the mesh simplification seen in Figure 7 needs to be done several times until the optimal visual aspects are found. The optimal visual aspect can be detected when altering the percentage of optimization until a detailed enough image with no loss of important features from the original design is displayed. During this process you can lose some unimportant details for the final object. After the mesh simplification is finished, the model needs to be exported as a file type FBX which is the type of file that Unity utilizes. After importing the file to Unity, the performance needs to be satisfactory in order to be successful, shown in Figure 8.

Figure 7. Surface Mesh Simplification. (Cacciola, 2010)
Figure 8. Mesh Simplification Diagram. (Samudio, 2018)
Once the models were completed, the next software *Autodesk 3D Studio Max*, was used as an extension for the 3D CADs files to simplify the mesh of the objects. A mesh consists of triangles arranged in a 3D space to create the impression of a solid object. The triangles are enough to define the basic shape (*Unity*, 2019). The mesh is what makes a file complicated with a structure hard to render in a virtual environment. In *Autodesk 3D Studio Max*, there are some options that are very useful for simplifying the mesh of the objects. This is done by reducing the number of faces and vertices in an object by simplifying the geometry, which speeds up the rendering while maintaining an acceptable image (*Autodesk*, 2019). The more complicated the mesh, the more detailed the object and the better the visual aspects of the objects in the virtual environment. Figure 9 illustrates an example of an object with 100% mesh in three different views from the 3D Studio Max through default shading view and two wireframes override to clarify the visual aspect of the mesh simplification is one the objects used in this project.

Another important consideration about the meshes are that they allow objects to have correct lighting, color, and textures for creating a more realistic environment.

Figure 9. Visual aspect of a 100% mesh simplification default shading and its wireframe override.
Figure 10. Visual aspect of a 60% mesh default shading and its wireframe override.

Figure 10 shows a 60% mesh simplification where a decrease in the number of vertices of the CAD file, results decrease in the quality of the design enough to maintain the details of the booth. This is the better option and the one that is used in the environment.

Figure 11. Visual aspect of a 10% mesh default shading and its wireframe override.

Figure 11 shows how working with a 10% mesh simplification decrease the quality of the object, deletes important details of the object, is not recommended to use in the environment.
3.3.3 Texture

The next step after is the mesh simplification, adding texture. One of the fundamental elements that makes up a virtual scenario is how close to real life the surroundings can be. Adding as many details as possible gives the user a better immersive feeling and understanding of the environment. The texture is added through the mesh geometry surface that provides the fine details of the objects. The textures were imported from open sources or through Unity databases and are assigned to a material in Unity. Once the texture is embedded as a material in the system it can be applied to an object.

In the virtual scenario two different textures were used, solid colors (Figure 12), and images that gave the object a material like finish (Figure 13). Textures are images that wrap or work as a stamp around the object to provide the final visual effect.

![Figure 12. Solid Colors for Texture.](image1)

![Figure 13. Metal Images for Texture.](image2)
Figure 14. Applied texture to the floor.

In Figure 14 and Figure 15 show the different options under the shader properties that have to be adjusted including Tiling, Offset, Emission, and Secondary Maps in a trial and error method until the best visual effect is reached. All objects contain textures either solid colors or images.

Figure 15. Applied texture to the table.
3.3.4 Lights and Sound Effects

Some other important aspects of designing the environment are the lights and the sound effects. In Unity there are different types of lights and selecting the correct light will define the color and the mood of the environment.

During a sensory evaluation it is required to have the correct light that provides the original color and brightness of the product being evaluated and most importantly, avoiding shadows that contribute to altering the integrity of the product. Due to these reasons, a spotlight was added and the “no shadows” option is selected right at the top of the sensory booth (Figure 16).

![Figure 16. Lighting specifications for the spotlight.](image)

The sound effects used in the scenario are “sniff” effects, performed while the user was doing the scent identification; a “sliding” effect, displayed when a new tray was shown in the scenario; and finally a “placing glass on the table” effect, illustrate the effect of the cups while they were placed on the tray.
3.3.5 Colliders and Rigid Bodies

After designing the environment and adding the realistic textures to the objects, physics laws are added to the game. To add physical behavior to the object some characteristics are required to work as they do in real-life such as gravity, mass, weight, drag, collisions, and make the objects gradable applied through the software.

*Unity (2019)* explains physics as: “The convincing physical behavior, an object has in a game must accelerate correctly and be affected by collisions, gravity and other forces. *Unity’s* built-in physics engines provide components that handle the physical simulation for you. With just a few parameter settings, you can create objects that behave passively in a realistic way (i.e., they will be moved by collisions and falls but will not start moving by themselves). By controlling physics from scripts, you can give an object the dynamics of a vehicle, a machine, or even a piece of fabric.”

To add physics to an object the creation of a rigid body is the main component that is attached to the object that will react when using gravity. Another main component is the collider, which are geometrical shapes or primitive types that are invisible to the view of the user but respond to collisions, the main colliders used in the scenario were box colliders.

A collider can be manipulated by scripting or manually altering the setting. When manipulating the setting using scripting the collisions can be detected when adding another collider and, in this case, the original collider needs to be configured as a trigger and that will allow the other collider to pass through.
This trigger property has been used for the training during the sniffing performance, a collider was created close to the OVR Player controller generating the idea of a nose that provoke the sniff action when the collider is triggered by any of the cups that contain the scents, and at the same time generate the pumping of the corresponding scent using the script of the Arduino.

Figure 17. Box Colliders and Rigid bodies properties in the Sensory Booth.

Another useful property is kinematics, which are attached to the rigid body. It is used when an object needs to act as a static collider and needs to move when some force is applied to the object or disabled when it is not required. The difference between static collider and kinematic property is that with kinematic is applied the object still has friction. Another property that can be adjusted is the gravity, it can be disable like in the case of the booth in Figure 17 where the rigid body was kinematic but no gravity force was applied. The colliders prevent either the user or the other objects to pass through the surface of the booth.
Figure 18. Gravity and kinematic properties with box colliders in the tray.

Figure 18 shows the properties attached to the tray. The tray is an object that can interact with the user, so it is required to have physical properties such as mass, gravity and kinematics to have friction and enable movement when required. The colliders were useful to avoid passing through the surface when the user was interacting with the cups.

Figure 19. Grabbable script, box collider and rigid body properties for the cup.
In Figure 19 the properties attached to the cups are shown. The cup is an object that can interact with the user, so it is required to have physics such as mass, gravity and kinematics to have friction and enable movement when required. For the cups the mass added was bigger than the mass added to the tray because when the grabbable script is added to the cups the interaction is increased and with it the possibilities of hitting another cup and falling around the tray or even all over the booth surface. Are also increasing the mass of the cups facilitates the handling. The colliders were added to avoid passing through the surface when the user was interacting with the cups.

Figure 20. Oculus integration with Unity.

Oculus offers SDK (Software Development Kit) to integrate with Unity program which includes scripts for the controllers and headsets named as OVR (Oculus Virtual Reality). The kit consists of an interface for controlling the VR camera behavior, a first-person control, object grabbing.
Figure 20 is shows the character controller, here the OVR player is controlled with the purpose of creating the interaction in the scenario.

### 3.3.6 Scripting Programming *Unity*

The last part of the process is creating the virtual scenario including scripting the program to interact with all the physical properties of the objects and their reactions while the user is interacting in the immersive environment.

*Unity* (2019) explained scripting as: “An essential ingredient in all applications you make in all games. Most applications need scripts to respond to inputs from the player and to arrange for events in the gameplay to happen when they should. Beyond that, scripts can be used to create graphical effects, control the physical behavior of objects or even implement a custom system for characters in the game.”

Using scripts enables the possibility of modifying properties of the object and allows the user to respond the way that is expected in real life. Through *Unity* the programming can be compiled by several languages, in this project C# was the language used for programming. The scripts created the connection between the software and the user.

The way a script works is by modifying the properties available in *Unity* that are visible through the inspector toolbar of the software. Basically, all properties that are shown in the inspector panel can be triggered using commands in C# programming.

To create the communication in the Scenario for Sensory Training there were two scripts developed that allowed the user to interact with the environment:

1. The interaction between the user that allowed to grab the cups with the five scents (Figure 21 and Appendix B).
2. The states machines, named in this project as cases, which include a total of six cases and the seven for ending the game to complete the sensory training (Figure 22, Figure 39 and Appendix C).

![Figure 21. Smell scripting shown in inspector toolbar from Unity.](image)

Figure 21 shows how the creation of variables for the Smell scripting through C# were visible in the inspector toolbar inside of Unity software. Two types of variables were displaying: the string variables as an array of characters, and Boolean variables that have two pre-defined possible values True and False. The value of False is equivalent to zero (0) and any nonzero value is considered True.
Appendix B detailed scripting is presented for the Smell programming which create the interaction between the scent generator, which releases each scent, and the user when grabbing the cups. In the script the variable public string “uno” means the cup number one, “dos” means cup two, “tres” means cup three, “cuatro” means cup four, “cinco” means cup five, “seis” means cup six, which is linked to the pumps is called “public Boolean B_Uno”, which means button one, so on and so forth. Through the previous details, it is explained that the pumps being Boolean variables when having a value of zero or false means the pump is turned off and when the pump has a value of one or true that pump is triggered and released the scent.

The command that triggers the pumps from one to five is called “OnTriggerEnter”, what it means is that when one of the cups enter the collider created to represent the nose, the pump that corresponds to that cup will trigger and that scent is released. In the opposite way, the command that turn off the previous pump and triggers just the pump number six is the command called “OnTriggerExit”. This script creates the communication between the Arduino 1 that has embedded its own programming with the user through the Unity software.

![Figure 22. Sequence scripting in inspector toolbar from Unity.](image)
Figure 22 shows how the creation of variables for Sequence scripting through C# are visible in the inspector toolbar inside of Unity software. The scripting integrated the concept design of the Sensory Training presented as states or called later as cases.

Another important concept is the buttons that displayed on the screen as the user progresses in the training. Depending on which state the user was in the training, certain objects interacts and shown important information for the user.

The same happened with the audio instructions, those audios were specific for each state considering the state in which the user was performing, certain instruction was given to continue to the next state of the training.

In the Appendix C the entire sequence programming is shown for details of how the script runs in the actual training. For creating the interface on the screen there were buttons created using canvas feature from Unity that allowed the user to interact while being immerse on the game. Six game objects were created for each button called G_B1 to G_B6.

The audio scripts were added as audio clips that depending on the case, those were triggered or not. All Boolean values were used for flag purposes that helped during the creation of the program. The creation of state machine for randomized the position of the cups were developed on each case.

The scripting is done to show just the objects required as a true expression or disable those objects not required as a false expression.
3.4 Scent Generator Design

In this project, a Scent Generator is designed using *SolidWorks*, to make it able to release multiple scents in the virtual environment for the training it was created by a 3D printer at Morehead State University (MSU).

![Scent Generator Design](image)

**Figure 23. Scent Generator Design.**

The Scent Generator Figure 23 consists of an *Arduino Uno* (A) connected by a relay (B) to six peristaltic pumps (C) using 12 volts and from the pump using hoses to create the connection of the scents (D) localized in their tray (E) to the pumps and using a splitter (F) for release those scents to the user. The case of the Scent Generator consists of a base (G) containing the *Arduino Uno* and the relay, the middle case (H) that holds the six peristaltic pumps, the top case (I) hold
the scents, the trays and the splitter, the cover of the case (J) and finally the hole for the hose (K) that release the scent next to the user’s nose.

Figure 24. The architecture of the Scent Generator.

Figure 24 shows how the architecture of the Scent Generator is connected using the Oculus headset which is connected to the computer, and the computer attached to the scent generator. The Scent Generator holds the Arduino that is joined to the peristaltic pumps and from here they are hooked to up the scents by hoses until reach the user’s nose.

3.4.1 Scripting Arduino Programming

One of the main components that holds the Scent Generator is the Arduino Uno, which works with C++ language for programming. In the project the Arduino is required to create the script for controlling the six peristaltic pumps and then connect to the scenario.

Appendix D shows the scripting for the Arduino which represents the way the pumps are triggered by commands. The six pumps are represented as outputs starting from the number two to seven. In the loop the script is telling the Arduino that the conditional command is that if a
sign is received then pump one is triggered and is represented as “High” and when it is not required the pump script is represented as “Low” when it means is that the pump is turned off.

This script receives a sign from the scenario when the user encounters the collider of the nose with one of the cups, depending on which cup it is, the sign will trigger the command first in the scripting used in *Unity* and then in the *Arduino* scripting.

With this programming the system is ready to work, and the immersive environment reacts like real life with all the components of physics, colors, textures and lights.
Chapter 4 Findings and Results

4.1 Scent Generator

The final prototype of the Scent Generator is shown in Figure 25, contained in the base (G) of the integration systems for the Arduino and a relay, in the middle case (H) holds the six peristaltic pumps (C), three on each side of the case. In this figure is shown the pumps one, two and three. The top case (I) holds the five scents (D) and the coffee is contained in vials and the splitter. In this figure is shown the vials four, five and six. In the cover of the case (J) is the hole (K) for the final hose. The tray for the scents (E) is finally out of the case for is part of the prototype to hold the scents during transportation or for storage. The labels correspond to the same letter that is used in Figure 23 for the Scent Generator design.

Figure 25. Scent Generator Prototype.
Figure 26. *Arduino* Uno and Relay Integration.

Figure 26 shows the integration of the *Arduino* Uno (A) and the relay (B) that connects each pump to the power supply, the controller and the pumps.

Figure 27. Multiple Scents Connection.

From each pump (C), a hose connects from one end the scent (D) and the other end to a splitter (F) creating the connection of the multiple scents, this is shown in Figure 27.
Finally, the integration of the scent generator to the Oculus headset and the computer is shown in Figure 28 which is used during the Sensory Training.

One finding of the prototype design was that the time required to release the scent was ten seconds due to the hose’s distance from the splitter to the headset, so the user needs to wait ten seconds to start detecting any scent. Another finding is that because of the previous reason, the system requires ten seconds to cleanse the lines. In order to clean the lines, pump six is designed for this action and was connected to the splitter in the opposite way as the other five pumps, this is because it is required to suck the air from the system and this cleans the line to prevent possible cross contamination between different scents. The pump six is the one that contains coffee in the vial to absorb the aromas and neutralize the smell for the user.
4.2 Virtual Reality Scenario Design

The training was divided into six cases to give the correct sequential flow, each case is explained and shown in the next figures along with the audio scripts that the user hears while being immersed in the environment.

Figure 29. Virtual Reality Scenario Case 1.

The first screen, Case 1, shown in Figure 29, the user hears and sees: “Welcome to the virtual scenario for sensory training. Today you will be participating in a test to determine the differences between smells.” Next, the user hears: “If this is not your first time playing, go ahead and click to the test. If it is your first time playing, click on the tutorial where you’ll get acquainted with the different smells.” This explains the reason of having two buttons on the screen, one for the “Tutorial” and the second one for the “Test”, in this case the user just wants to take the test.
The second screen, Case 2, show in Figure 30, referees to the “Tutorial” where the user can read and heard: “Before you there are five different smells going from left to right, they are: bora bora, aqua, bamboo, midnight storm, and a blank.” Next, the user heard: “The cups are picked up with a smooth grabbing motion and should be brought up to the box with a steady motion. Hold it there for ten seconds till you identify the scent. Make sure to place the cup back in its designated place after use. After setting down the cup, the sixth pump will kick on, wait at least ten seconds to clear the line and your nose”. When the user finished identifying the scents, they clicked the next button.
In the third screen, Case 3, shown in Figure 31, the user finished with the tutorial identified the five scent. The user read and heard: “If you’re now familiar with the smells, go ahead and click to the test. If you need more time keep going through the tutorials.”
In the fourth screen, Case 4, shown in Figure 32, the user chose the “Test” evaluation, on the screen the user read and heard: “Before you are three different cups, two are the same, and one is different. They all have random numbers on them, but your job is to discern which cup smells different. Once you’re done smelling each cup, click the button that corresponds with the different smell. If you need to smell one again then do so before you make a choice”. Next, the user heard: “Utilizing a smooth grabbing motion pick up the cup and bring it to the box, holding it there for ten seconds till you can discern a scent. Make sure to place the cup back in its designated place after use. After setting down the cup, the sixth pump will kick on, wait at least ten seconds to clear the line and your nose.”

![Image](image.png)

**Figure 33. Virtual Reality Scenario Case 5.**

In the fifth screen, Case 5, shown in Figure 33, the user finished the test evaluation and heard: “Here’s your results. If you didn’t get a perfect score, you will need to wait in order to retake the test. This is so your nose can reset. If this is your second take, you need to go back to the tutorial and refamiliarize yourself with the smells.”
Figure 34 shows the user in the immersive environment selecting the option of tutorial in the scenario and in the Figure 35 it shows the user grabbing a cup and bringing up to the box with a steady motion, holding it there for ten seconds till the user identified the scent.
Chapter 5 Summary, Conclusions and Further Work

5.1 Summary

The overall objective of this research was to develop a Virtual Reality Scenario for Sensory Training with a Scent Generator which helps release five different scents to the users’ nose and with this, recreate the traditional scenario used for sensory evaluation. All these objectives were successfully achieved.

The first step was to design the CAD files to implement the final layout of the laboratory for sensory training. A mesh simplification was required for all the CAD files. The sensory training was designed to help the user get involved with the five scents and at the end prove their skills by doing the final test.

The environment required the use of different textures, lights, audio scripts and adding physics to the game objects to make the scenario realistic. The behavior of the virtual scenario was achieved using C# programming through two scripts, one for the smell connecting the scent generator to the software and the second script was the sequence of the sensory training to interact with the user.

Finally, the development of the Scent Generator using a 3D printed design that contained the five scents that were released was designed, created and tested.

5.2 Conclusions

The results of the present study suggest that the integration of the virtual environment, the training, and the Scent Generator have successful interactions and create an immersive environment where the user can have a multisensory experience during the sensory training.
The present study also suggests that the Scent Generator can release five unique scents with its pumps, and with the sixth pump sucking is able to cleanse the system.

5.3 Further work

For future work it is suggested to implement the Virtual Reality Scenario to train a sensory panel based on the design created for the present study. It can be also used in other applications in which Virtual Reality needs a multisensory experience such as videogames or trainings for fields like medical or machine.
References


Autodesk Inc. 2019. 3ds Max Optimization Modifiers.  


https://docs.unity3d.com/Manual/AnatomyofaMesh.html


Appendix A

Experiments and Results:

Figure 14. Applied texture to the floor.
Figure 16. Lighting specifications for the spotlight.
Figure 17. Box Colliders and Rigid bodies properties in the Sensory Booth.
Figure 18. Gravity and kinematic properties with box colliders in the tray.
Figure 19. Grabbable script, box collider and rigid body properties for the cup
Figure 21. Smell scripting shown in inspector toolbar from Unity.
Figure 23. Sequence scripting in inspector toolbar from Unity.
Figure 36. Integration of Virtual Reality in Sensory Training.

Figure 37. Costs of the 3D printed prototype.
Figure 38. Elements of the Scent Generator.

Figure 39. State machine for the Sensory Training program.
Case 1

The user heard and read:

“Welcome to the virtual scenario for sensory training. Today you will be participating in a test to determine the differences between smells.”

Next, the user heard:

“If this is not your first time playing, go ahead and click to the test. If it is your first time playing, click on the tutorial where you’ll get acquainted with the different smells.”

This explains the reason of having two buttons on the screen, one for the “Tutorial” and the second one for the “Test”. In case if the user just wants to take the test.

Figure 40. Case 1 shown on game with audio scripts.

Case 2

In the “Tutorial” the user read and heard:

“Before you there are five different smells going from left to right, they are: bora bora, aqua, bamboo, midnight storm, and a blank.”

Next, the user heard:

“The cups are picked up with a smooth grabbing motion and should be brought up to the box with a steady motion. Hold it there for ten seconds till you identify the scent. Make sure to place the cup back in its designated place after use. After setting down the cup, the sixth pump will kick on, wait at least ten seconds to clear the line and your nose”. 

When the user finished identifying the scents they clicked the next button.

Figure 41. Case 2 shown on game with audio scripts.
Case 3

The user finished with the tutorial identifying the five scents.

The user read and heard:

“If you’re now familiar with the scents, go ahead and click to the test. If you need more time keep going with the tutorials.”

Figure 42. Case 3 shown on game with audio scripts

Case 4

In the “Test” evaluation, the user read and heard:

“Before you are three different cups, two are the same, and one is different. They all have random numbers on them, but your job is to discern which cup smells different. Once you’re done smelling each cup, click the button that corresponds with the different smell. If you need to smell one again then do so before you make a choice.”

Next, the user heard:

“Utilizing a smooth grabbing motion pick up the cup and bring it to the box, holding it there for ten seconds till you can discern a scent. Make sure to place the cup back in its designated place after use. After setting down the cup, the sixth pump will kick on, wait at least ten seconds to clear the line and your nose.”

Figure 43. Case 4 shown on game with audio scripts
Figure 44. Case 5 shown on game with audio scripts

Case 5

- If the user finished the test evaluation and select the correct smell will heard: “Well done!”
- Next, the user click the quit button.

Figure 45. Case 6 shown on game with audio scripts

Case 6

- If the user finished the test evaluation and select the wrong smell will heard: “Wrong answer! You will need to wait in order to retake the test. This is so your nose can reset. If this is your second take, you need to go back to the tutorial and refamiliarize yourself with the smells.”
- Next, the user click the quit button.
Appendix B

Smell scripting in *Unity*

```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;

public class Smell : MonoBehaviour
{
    public string uno;
    public string dos;
    public string tres;
    public string cuatro;
    public string cinco;
    public string seis;
    public bool B_Uno;
    public bool B_Dos;
    public bool B_Tres;
    public bool B_Cuatro;
    public bool B_Cinco;
    public bool bandera;

    public GameObject G_Uno;
    public GameObject G_Dos;
    public GameObject G_Tres;
    public GameObject G_Cuatro;
    public GameObject G_Cinco;

    SerialPort stream = new SerialPort("COM3", 9600);
    public char[] send;
    void Start()
    {
        uno = "Uno";
        dos = "Dos";
        tres = "Tres";
        cuatro = "Cuatro";
        cinco = "Cinco";
        stream.Open();
        stream.WriteLine("0");
    }

    // Update is called once per frame
    void Update()
    {
    }

    void OnTriggerEnter(Collider other)
    {
    }
```
if (((other.gameObject.name == "Uno" || other.gameObject.name == "Uno (Clone)" ) /& G_Uno.gameObject.GetComponent<Rigidbody>().isKinematic == true) || ((other.gameObject.name == "Dos" || other.gameObject.name == "Dos(Clone)" ) /& G_Dos.gameObject.GetComponent<Rigidbody>().isKinematic == true) || (other.gameObject.name == "Tres" || other.gameObject.name == "Tres(Clone)" ) /& G_Tres.gameObject.GetComponent<Rigidbody>().isKinematic == true) || ((other.gameObject.name == "Cuatro" || other.gameObject.name == "Cuatro(Clone)" ) /& G_Cuatro.gameObject.GetComponent<Rigidbody>().isKinematic == true) || ((other.gameObject.name == "Cinco" || other.gameObject.name == "Cinco(Clone)" ) /& G_Cinco.gameObject.GetComponent<Rigidbody>().isKinematic == true)
{
    B_Uno = false;
    B_Dos = false;
    B_Tres = false;
    B_Cuatro = false;
    B_Cinco = false;
    stream.WriteLine("6");
    bandera = true;
}

void OnTriggerEnter(Collider other)
{
    if (((other.gameObject.name == "Uno" || other.gameObject.name == "Uno (Clone)" ) /& G_Uno.gameObject.GetComponent<Rigidbody>().isKinematic == true) )
    {
        B_Uno = true;
        B_Dos = false;
        B_Tres = false;
        B_Cuatro = false;
        B_Cinco = false;
        stream.WriteLine("1");
    }
    else if (((other.gameObject.name == "Dos" || other.gameObject.name == "Dos(Clone)" ) /& G_Dos.gameObject.GetComponent<Rigidbody>().isKinematic == true) )
    {
        B_Uno = false;
        B_Dos = true;
        B_Tres = false;
        B_Cuatro = false;
        B_Cinco = false;
        stream.WriteLine("2");
    }
83  else if ((other.gameObject.name == "Tres" || other.gameObject.name == "Tres(Clone)"") /*& G_Tres.gameObject.GetComponent<Rigidbody>() */
84  ().isKinematic == true*/
85  {
86      B_Uno = false;
87      B_Dos = false;
88      B_Tres = true;
89      B_Cuatro = false;
90      B_Cinco = false;
91      stream.WriteLine("3");
92  }
93  else if ((other.gameObject.name == "Cuatro" || other.gameObject.name == "Cuatro(Clone)"") /*& G_Cuatro.gameObject.GetComponent<Rigidbody>() */
94  ().isKinematic == true*/
95  {
96      B_Uno = false;
97      B_Dos = false;
98      B_Tres = false;
99      B_Cuatro = true;
100     B_Cinco = false;
101     stream.WriteLine("4");
102   }
103  else if ((other.gameObject.name == "Cinco" || other.gameObject.name == "Cinco(Clone)"") /*& G_Cinco.gameObject.GetComponent<Rigidbody>() */
104  ().isKinematic == true*/
105  {
106      B_Uno = false;
107      B_Dos = false;
108      B_Tres = false;
109      B_Cuatro = false;
110     B_Cinco = true;
111     stream.WriteLine("5");
112  }
Appendix C

Sequence scripting in *Unity*

```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Sequence : MonoBehaviour
{
    public int state;
    public GameObject Display_Text;
    public GameObject Countdown;
    public GameObject Results;
    public GameObject Button_One;
    public GameObject Button_Two;
    public GameObject Button_Three;
    public GameObject Button_Four;
    public GameObject Button_Five;
    public GameObject Button_Six;
    public GameObject Escenario;

    public Canvas Botones;
    public Button Button_Text_One;
    public Button Button_Text_Two;
    public Button Button_Text_Three;
    public Button Button_Text_Four;
    public Button Button_Text_Five;
    public Button Button_Text_Six;

    public GameObject Uno;
    public GameObject Dos;
    public GameObject Tres;
    public GameObject Cuatro;
    public GameObject Cinco;

    public GameObject Tapa_1;
    public GameObject Tapa_2;
    public GameObject Tapa_3;
    public GameObject Tapa_4;
    public GameObject Tapa_5;

    AudioSource AudioSequence;

    public bool Play_1;
    public bool Play_1_2;
    public bool Play_2;
    public bool Play_2_2;
    public bool Play_3;
    public bool Play_4;
    public bool Play_4_2;
    public bool Play_5;
}
50   public bool Play_6;
51
52   public bool Cup_1;
53   public bool Cup_2;
54   public bool Cup_3;
55   public bool Cup_4;
56   public bool Cup_5;
57
58   public AudioSource Audio_Source;
59   public AudioClip Audio_Step1;
60   public AudioClip Audio_Step1_2;
61   public AudioClip Audio_Step2;
62   public AudioClip Audio_Step2_2;
63   public AudioClip Audio_Step3;
64   public AudioClip Audio_Step4;
65   public AudioClip Audio_Step4_2;
66   public AudioClip Audio_Step5;
67   public AudioClip Audio_Step6;
68
69   public bool B1;
70
71   public GameObject G_B1;
72   public GameObject G_B2;
73   public GameObject G_B3;
74   public GameObject G_B4;
75   public GameObject G_B5;
76   public GameObject G_B6;
77
78   public int First_Cup;
79   public int Second_Cup;
80   public bool aux_Cup;
81
82   public int pos_1;
83   public int pos_2;
84   public int pos_3;
85
86   public Vector3 V_Pos_1;
87   public Vector3 V_Pos_2;
88   public Vector3 V_Pos_3;
89
90   void Start()
91   {
92       state = 1;
93       B1 = false;
94
95       Play_1 = false;
96       Play_1_2 = false;
97       Play_2 = false;
98       Play_2_2 = false;
```csharp
D:\Unity_Projects\yoss - 111419 - V5\Assets\Sequence.cs

99     Play_3 = false;
100    Play_4 = false;
101    Play_4_2 = false;
102    Play_5 = false;
103    Play_6 = false;
104
105    Cup_1 = false;
106    Audio_Source = GetComponent<AudioSource>();
107
108    aux_Cup = false;
109
110 }
111 // Update is called once per frame
112
116 void Update()
117 {
118    switch (state)
119    {
120
121        case 1:
122            Display_Text.gameObject.GetComponent<TextMesh>().text = "Welcome to the virtual scenario for sensory training" +
123                                                                       System.Environment.NewLine +
124                                                                       "you will be participating in a test to determine" +
125                                                                       System.Environment.NewLine +
126                                                                       "the differences between smells";
127            Button_One.gameObject.active = true;
128            Button_Two.gameObject.active = true;
129            Button_Three.gameObject.active = false;
130            Button_Four.gameObject.active = false;
131            Button_Five.gameObject.active = false;
132            Button_Six.gameObject.active = false;
133
134            G_B1.gameObject.active = true;
135            G_B2.gameObject.active = true;
136            G_B3.gameObject.active = false;
137            G_B4.gameObject.active = false;
138            G_B5.gameObject.active = false;
139            G_B6.gameObject.active = false;
140
141            Button_Text_One.GetComponentInChildren<Text>().text = "Tutorial";
142            Button_Text_Two.GetComponentInChildren<Text>().text = "Test";
143            Button_Text_Three.GetComponentInChildren<Text>().text = "Next";
```
Uno.gameObject.active = false;
Dos.gameObject.active = false;
Tres.gameObject.active = false;
Cuatro.gameObject.active = false;
Cinco.gameObject.active = false;

// Tapa_1.gameObject.active = false;
// Tapa_2.gameObject.active = false;
// Tapa_3.gameObject.active = false;
// Tapa_4.gameObject.active = false;
// Tapa_5.gameObject.active = false;

if (Play_1 == false && AudioSource.isPlaying == false)
{
    AudioSource.PlayOneShot(Audio_Step1);
    Play_1 = true;
}

else if (Play_1 == true && Play_1_2==false && AudioSource.isPlaying == false)
{
    AudioSource.PlayOneShot(Audio_Step1_2);
    Play_1_2 = true;
}

Countdown.gameObject.GetComponent<TextMesh>().text = "10 seconds";
Countdown.gameObject.active = false;
Results.gameObject.GetComponent<TextMesh>().text = "Well done!";
Results.gameObject.active = false;

if (G_B1.gameObject.GetComponent<Button>().On_Off == true && Play_1==true && Play_1_2==true && AudioSource.isPlaying == false) {
    state = 2;
} else if (G_B2.gameObject.GetComponent<Button>().On_Off == true && Play_1 == true && Play_1_2 == true && AudioSource.isPlaying == false) {
    state = 4;

} //else if bu
// AudioSource.Audio_Step1 = GetComponent<AudioSource>();
// Audio_Step1.Play(0);
// Debug.Log("started");

// AudioSource = GetComponent<AudioSource>();
// AudioSource.clip = Resources.Load<AudioClip> ("Audio_Step2");
// AudioSource.Play();

// Audio_Source.PlayOneShot(Audio_Step1, 1);
// this would be welcome audio
// Audio_Source.PlayOneShot(Audio_Step1_2, 1);
// this would be the question of whether or not it is the first time playing, we don't have to include the text

break;

case 2: //tutorial
    Display_Text.gameObject.GetComponent<TextMesh>().text = "Before you there are five different smells going from left to right," + System.Environment.NewLine + "they are: bora bora, aqua," + System.Environment.NewLine + "bamboo, midnight storm, and a blank.";
    Button_One.gameObject.active = false;
    Button_Two.gameObject.active = false;
    Button_Three.gameObject.active = true;
    Button_Four.gameObject.active = false;
    Button_Five.gameObject.active = false;
    Button_Six.gameObject.active = false;
    G_B1.gameObject.active = false;
    G_B2.gameObject.active = false;
    G_B3.gameObject.active = true;
    G_B4.gameObject.active = false;
    G_B5.gameObject.active = false;
    G_B6.gameObject.active = false;

    if (Cup_1 == false) {
        Uno.gameObject.active = true;
        Uno.gameObject.transform.localPosition = new Vector3 (-0.008899689f, -0.2876806f, -0.4890001f);
        Uno.gameObject.transform.localEulerAngles = new Vector3(0f, 0f, 0f);
        Cup_1 = true;
    }

    if (Cup_2 == false)
Dos.gameObject.active = true;
Dos.gameObject.transform.localPosition = new Vector3 (-0.01279831f, -0.2876806f, -0.4070001f);
Dos.gameObject.transform.localEulerAngles = new Vector3(0f, 0f, 0f);
Cup_2 = true;

if (Cup_3 == false)
{
    Tres.gameObject.active = true;
    Tres.gameObject.transform.localPosition = new Vector3 (-0.01279831f, -0.2876806f, -0.3197001f);
    Tres.gameObject.transform.localEulerAngles = new Vector3(0f, 0f, 0f);
    Cup_3 = true;
}

if (Cup_4 == false)
{
    Cuatro.gameObject.active = true;
    Cuatro.gameObject.transform.localPosition = new Vector3 (-0.008899689f, -0.2797906f, -0.2338001f);
    Cuatro.gameObject.transform.localEulerAngles = new Vector3(0f, 0f, 0f);
    Cup_4 = true;
}

if (Cup_5 == false)
{
    Cinco.gameObject.active = true;
    Cinco.gameObject.transform.localPosition = new Vector3 (-0.01279831f, -0.2797906f, -0.1365201f);
    Cinco.gameObject.transform.localEulerAngles = new Vector3(0f, 0f, 0f);
    Cup_5 = true;
}

// Tapa_1.gameObject.active = true;
// Tapa_2.gameObject.active = true;
// Tapa_3.gameObject.active = true;
// Tapa_4.gameObject.active = true;
// Tapa_5.gameObject.active = true;

Countdown.gameObject.GetComponent<TextMesh>().text = "10 seconds";
Countdown.gameObject.active = true;
Results.gameObject.GetComponent<TextMesh>().text = "Well done!";
Results.gameObject.active = false;

Button_Text_One.GetComponentInChildren<Text>().text = "Tutorial";
Button_Text_Two.GetComponentInChildren<Text>().text = "Test";
Button_Text_Three.GetComponentInChildren<Text>().text = "Next";

if (Play_2 == false && AudioSource.isPlaying == false)
{
    AudioSource.PlayOneShot(Audio_Step2);
    Play_2 = true;
}

else if (Play_2 == true && Play_2_2 == false && AudioSource.isPlaying == false)
{
    AudioSource.PlayOneShot(Audio_Step2_2);
    Play_2_2 = true;
}

if (G_B3.gameObject.GetComponent<Button>().On_Off == true && Play_2 == true && Play_2_2 == true && AudioSource.isPlaying == false)
{
    state = 3;
    G_B3.gameObject.GetComponent<Button>().On_Off = false;
}

//need to have public gameobjects for cups as to turn them on/ off
break;
case 3: //test
Display_Text.gameObject.GetComponent<TextMesh>().text = "If you're now familiar with the smells go ahead and click to the test. " + System.Environment.NewLine + "need more time keep going with the tutorials.";
    Button_One.gameObject.active = true;
    Button_Two.gameObject.active = true;
    Button_Three.gameObject.active = false;
    Button_Four.gameObject.active = false;
    Button_Five.gameObject.active = false;
    Button_Six.gameObject.active = false;

    G_B1.gameObject.active = true;
    G_B2.gameObject.active = true;
    G_B3.gameObject.active = false;
G_B4.gameObject.active = false;
G_B5.gameObject.active = false;
G_B6.gameObject.active = false;

Button_Text_One.GetComponentInChildren<Text>().text = "Tutorial";
Button_Text_Two.GetComponentInChildren<Text>().text = "Test";

Countdown.gameObject.GetComponent<TextMesh>().text = "10 seconds";
Countdown.gameObject.active = false;
Results.gameObject.GetComponent<TextMesh>().text = "Well done!";
Results.gameObject.active = false;

Uno.gameObject.active = false;
Dos.gameObject.active = false;
Tres.gameObject.active = false;
Cuatro.gameObject.active = false;
Cinco.gameObject.active = false;

Tapa_1.gameObject.active = false;
Tapa_2.gameObject.active = false;
Tapa_3.gameObject.active = false;
Tapa_4.gameObject.active = false;
Tapa_5.gameObject.active = false;

//Button_Text_Two.GetComponent<TextMesh>().text = "Test";

if (Play_3 == false && AudioSource.isPlaying == false)
{
    AudioSource.PlayOneShot(Audio_Step3);
    Play_3 = true;
}

if (G_B1.gameObject.GetComponent<Button>().On_Off == true &&
    Play_3 == true && AudioSource.isPlaying == false)
{
    state = 2;
    Cup_1 = false;
    Cup_2 = false;
    Cup_3 = false;
    Cup_4 = false;
    Cup_5 = false;
}
else if (G_B2.gameObject.GetComponent<Button>().On_Off == true &&
    Play_3 == true && AudioSource.isPlaying == false)
{

state = 4;
Cup_1 = false;
Cup_2 = false;
Cup_3 = false;
Cup_4 = false;
Cup_5 = false;
}
break;
case 4: // results
Display_Text.gameObject.GetComponent<TextMesh>().text = "Before you are three different cups, two are the same," +
System.Environment.NewLine +
    "and one is different. They all have random numbers on them," +
System.Environment.NewLine +
    "but your job is to discern which cup smells different." +
System.Environment.NewLine +
    "Once you’re done smelling each cup, click the button" +
System.Environment.NewLine +
    "that corresponds with the different smell." +
System.Environment.NewLine +
    "If you need to smell one again then do so before you make a choice.;"
Button_One.gameObject.active = false;
Button_Two.gameObject.active = false;
Button_Three.gameObject.active = false;

Button_Four.gameObject.active = true;
Button_Five.gameObject.active = true;
Button_Six.gameObject.active = true;

G_B1.gameObject.active = false;
G_B2.gameObject.active = false;
G_B3.gameObject.active = false;

G_B4.gameObject.active = true;
G_B5.gameObject.active = true;
G_B6.gameObject.active = true;

Button_Text_One.GetComponentInChildren<Text>().text = "1st Cup";}
Button_Text_Two.GetComponentInChildren<Text>().text = "2nd Cup";
Button_Text_Three.GetComponentInChildren<Text>().text = "3rd Cup";
Countdown.gameObject.GetComponent<TextMesh>().text = "10 seconds";
Countdown.gameObject.active = true;
Results.gameObject.GetComponent<TextMesh>().text = "Well done!"
Results.gameObject.active = false;
Uno.gameObject.active = false;
Dos.gameObject.active = false;
Tres.gameObject.active = false;
Cuatro.gameObject.active = false;
Cinco.gameObject.active = false;

if (aux_Cup == false) {
    First_Cup = Random.Range(1, 6);
    do {
        Second_Cup = Random.Range(1, 6);
    } while (First_Cup == Second_Cup);
    aux_Cup = true;
    pos_1 = Random.Range(1, 4);
    do {
        pos_2 = Random.Range(1, 4);
    } while (pos_1 == pos_2);
```csharp
        do
        {
            pos_3 = Random.Range(1, 4);
        } while ((pos_3 == pos_2) || (pos_3 == pos_1));

        /*V_Pos_1 = new Vector3(-0.008899689f, -0.2876806f, -0.4890001f);
V_Pos_2 = new Vector3(-0.01279831f, -0.2876806f, -0.3197001f);
V_Pos_3 = new Vector3(-0.01279831f, -0.2797906f, -0.1365201f);*/

        if (pos_1 == 1) {
            V_Pos_1 = new Vector3(-0.008899689f, -0.2876806f, -0.4890001f);
        } else if (pos_1 == 2) {
            V_Pos_1 = new Vector3(-0.01279831f, -0.2876806f, -0.3197001f);
        } else if (pos_1 == 3) {
            V_Pos_1 = new Vector3(-0.01279831f, -0.2797906f, -0.1365201f);
        }

        if (pos_2 == 1) {
            V_Pos_2 = new Vector3(-0.008899689f, -0.2876806f, -0.4890001f);
        } else if (pos_2 == 2) {
            V_Pos_2 = new Vector3(-0.01279831f, -0.2876806f, -0.3197001f);
        } else if (pos_2 == 3) {
            V_Pos_2 = new Vector3(-0.01279831f, -0.2797906f, -0.1365201f);
        }

        if (pos_3 == 1) {
            V_Pos_3 = new Vector3(-0.008899689f, -0.2876806f, -0.4890001f);
        }
```
else if (pos_3 == 2)
{
    V_Pos_3 = new Vector3(-0.01279831f, -0.2876806f, -0.3197001f);
}
else if (pos_3 == 3)
{
    V_Pos_3 = new Vector3(-0.01279831f, -0.2797906f, -0.1365201f);
}

if (First_Cup ==1) {
    GameObject aux_Uno_1;
    GameObject aux_Uno_2;
    aux_Uno_1 = Instantiate(Uno, V_Pos_1, Quaternion.Euler (0f, 0f, 0f));
    aux_Uno_2 = Instantiate(Uno, V_Pos_2, Quaternion.Euler (0f, 0f, 0f));
    aux_Uno_1.gameObject.GetComponent<Rigidbody>() .isKinematic = true;
    aux_Uno_2.gameObject.GetComponent<Rigidbody>() .isKinematic = true;
    aux_Uno_1.gameObject.active = true;
    aux_Uno_2.gameObject.active = true;
    aux_Uno_1.transform.parent = Escenario.transform;
    aux_Uno_2.transform.parent = Escenario.transform;
    aux_Uno_1.gameObject.transform.localPosition = V_Pos_1;
    aux_Uno_2.gameObject.transform.localPosition = V_Pos_2;
}

else if (First_Cup == 2)
{
    GameObject aux_Dos_1;
    GameObject aux_Dos_2;
    aux_Dos_1 = Instantiate(Dos, V_Pos_1, Quaternion.Euler (0f, 0f, 0f));
    aux_Dos_2 = Instantiate(Dos, V_Pos_2, Quaternion.Euler (0f, 0f, 0f));
    aux_Dos_1.gameObject.GetComponent<Rigidbody>() .isKinematic = true;
    aux_Dos_2.gameObject.GetComponent<Rigidbody>() .isKinematic = true;
aux_Dos_1.gameObject.active = true;
aux_Dos_2.gameObject.active = true;

aux_Dos_1.transform.parent = Escenario.transform;
aux_Dos_2.transform.parent = Escenario.transform;

aux_Dos_1.gameObject.transform.localPosition = V_Pos_1;
aux_Dos_2.gameObject.transform.localPosition = V_Pos_2;

} else if (First_Cup == 3)
{
    GameObject aux_Tres_1;
    GameObject aux_Tres_2;
    aux_Tres_1=Instantiate(Tres, V_Pos_1, Quaternion.Euler (0f, 0f, 0f));
    aux_Tres_2=Instantiate(Tres, V_Pos_2, Quaternion.Euler (0f, 0f, 0f));
    aux_Tres_1.gameObject.GetComponent<Rigidbody> ().isKinematic = true;
    aux_Tres_2.gameObject.GetComponent<Rigidbody> ().isKinematic = true;
    aux_Tres_1.gameObject.active = true;
aux_Tres_2.gameObject.active = true;

    aux_Tres_1.transform.parent = Escenario.transform;
aux_Tres_2.transform.parent = Escenario.transform;

    aux_Tres_1.gameObject.transform.localPosition = V_Pos_1;
aux_Tres_2.gameObject.transform.localPosition = V_Pos_2;

} else if (First_Cup == 4)
{
    GameObject aux_Cuatro_1;
    GameObject aux_Cuatro_2;
    aux_Cuatro_1=Instantiate(Cuatro, V_Pos_1, Quaternion.Euler(0f, 0f, 0f));
    aux_Cuatro_2=Instantiate(Cuatro, V_Pos_2, Quaternion.Euler(0f, 0f, 0f));
    aux_Cuatro_1.gameObject.GetComponent<Rigidbody> ().isKinematic = true;
    aux_Cuatro_2.gameObject.GetComponent<Rigidbody> ().isKinematic = true;
    aux_Cuatro_1.gameObject.active = true;
aux_Cuatro_2.gameObject.active = true;

    aux_Cuatro_1.transform.parent = Escenario.transform;
aux_Cuatro_2.transform.parent = Escenario.transform;
aux_Cuatro_1.gameObject.transform.localPosition = V_Pos_1;
aux_Cuatro_2.gameObject.transform.localPosition = V_Pos_2;
}
else if (First_Cup == 5)
{
    GameObject aux_Cinco_1;
    GameObject aux_Cinco_2;
    aux_Cinco_1 = Instantiate(Cinco, V_Pos_1, Quaternion.Euler(0f, 0f, 0f));
    aux_Cinco_2 = Instantiate(Cinco, V_Pos_2, Quaternion.Euler(0f, 0f, 0f));
    aux_Cinco_1.gameObject.GetComponent<Rigidbody>()
        .isKinematic = true;
    aux_Cinco_2.gameObject.GetComponent<Rigidbody>()
        .isKinematic = true;
    aux_Cinco_1.gameObject.active = true;
    aux_Cinco_2.gameObject.active = true;
    aux_Cinco_1.transform.parent = Escenario.transform;
    aux_Cinco_2.transform.parent = Escenario.transform;

    aux_Cinco_1.gameObject.transform.localPosition = V_Pos_1;
    aux_Cinco_2.gameObject.transform.localPosition = V_Pos_2;
}

if (Second_Cup == 1)
{
    GameObject aux2_Uno_1;
    aux2_Uno_1 = Instantiate(Uno, V_Pos_3, Quaternion.Euler(0f, 0f, 0f));
    aux2_Uno_1.gameObject.GetComponent<Rigidbody>()
        .isKinematic = true;
    aux2_Uno_1.gameObject.active = true;
    aux2_Uno_1.transform.parent = Escenario.transform;

    aux2_Uno_1.gameObject.transform.localPosition = V_Pos_3;
}
else if (Second_Cup == 2)
{
    GameObject aux2_Dos_1;
    aux2_Dos_1 = Instantiate(Dos, V_Pos_3, Quaternion.Euler(0f, 0f, 0f));
    aux2_Dos_1.gameObject.GetComponent<Rigidbody>()
        .isKinematic = true;
    aux2_Dos_1.gameObject.active = true;
    aux2_Dos_1.transform.parent = Escenario.transform;

    aux2_Dos_1.gameObject.transform.localPosition = V_Pos_3;
(0f, 0f, 0f));
    aux2_Dos_1.gameObject.GetComponent<Rigidbody>()
    .isKinematic = true;
    aux2_Dos_1.gameObject.active = true;
    aux2_Dos_1.transform.parent = Escenario.transform;
    aux2_Dos_1.gameObject.transform.localPosition = V_Pos_3;
}
else if (Second_Cup == 3)
{
    GameObject aux2_Tres_1;
    aux2_Tres_1 = Instantiate(Tres, V_Pos_3, Quaternion.Euler(0f, 0f, 0f));
    aux2_Tres_1.gameObject.GetComponent<Rigidbody>()
    .isKinematic = true;
    aux2_Tres_1.gameObject.active = true;
    aux2_Tres_1.transform.parent = Escenario.transform;
    aux2_Tres_1.gameObject.transform.localPosition = V_Pos_3;
}
else if (Second_Cup == 4)
{
    GameObject aux2_Cuatro_1;
    aux2_Cuatro_1 = Instantiate(Cuatro, V_Pos_3, Quaternion.Euler(0f, 0f, 0f));
    aux2_Cuatro_1.gameObject.GetComponent<Rigidbody>()
    .isKinematic = true;
    aux2_Cuatro_1.gameObject.active = true;
    aux2_Cuatro_1.transform.parent = Escenario.transform;
    aux2_Cuatro_1.gameObject.transform.localPosition = V_Pos_3;
}
else if (Second_Cup == 5)
{
    GameObject aux2_Cinco_1;
    aux2_Cinco_1 = Instantiate(Cinco, V_Pos_3, Quaternion.Euler(0f, 0f, 0f));
    aux2_Cinco_1.gameObject.GetComponent<Rigidbody>()
    .isKinematic = true;
    aux2_Cinco_1.gameObject.active = true;
    aux2_Cinco_1.transform.parent = Escenario.transform;
    aux2_Cinco_1.gameObject.transform.localPosition = V_Pos_3;
}
if (Play_4 == false & Audio_Source.isPlaying == false)
{
    Audio_Source.PlayOneShot(Audio_Step4);
    Play_4 = true;
}

else if (Play_4 == true & Play_4_2 == false &
          Audio_Source.isPlaying == false)
{
    Audio_Source.PlayOneShot(Audio_Step4_2);
    Play_4_2 = true;
}

if (G_B4.gameObject.GetComponent<Button>().On_Off == true &
    Play_4 == true & Play_4_2 == true &
    Audio_Source.isPlaying == false)
{
    if (pos_3 == 1)
    {
        state = 5;
    }
    else {
        state = 6;
    }

    G_B4.gameObject.GetComponent<Button>().On_Off = false;
}
else if (G_B5.gameObject.GetComponent<Button>().On_Off == true &
          Play_4 == true & Play_4_2 == true &
          Audio_Source.isPlaying == false)
{
    if (pos_3 == 2)
    {
        state = 5;
    }
}
else
{
    state = 6;
}
G_B5.gameObject.GetComponent<Button>().On_Off = false;
}
else if (G_B6.gameObject.GetComponent<Button>().On_Off == true && Play_4 == true && Play_4_2 == true && Audio_Source.isPlaying == false)
{
    if (pos_3 == 3)
    {
        state = 5;
    }
    else
    {
        state = 6;
    }
    G_B6.gameObject.GetComponent<Button>().On_Off = false;
}
break;

case 5: // Well Done


Button_One.gameObject.active = false;
Button_Two.gameObject.active = false;
Button_Three.gameObject.active = true;
Button_Four.gameObject.active = false;
Button_Five.gameObject.active = false;
Button_Six.gameObject.active = false;
G_B1.gameObject.active = false;
G_B2.gameObject.active = false;
G_B3.gameObject.active = true;
G_B4.gameObject.active = false;
G_B5.gameObject.active = false;
G_B6.gameObject.active = false;

Button_Text_One.GetComponentInChildren<Text>().text = "1st Cup";
Button_Text_Two.GetComponentInChildren<Text>().text = "2nd Cup";
Button_Text_Three.GetComponentInChildren<Text>().text = "Quit";

Countdown.gameObject.GetComponent<TextMesh>().text = "10 seconds";
Countdown.gameObject.active = false;
Results.gameObject.GetComponent<TextMesh>().text = "Well done!!";
Results.gameObject.active = false;

Uno.gameObject.active = false;
Dos.gameObject.active = false;
Tres.gameObject.active = false;
Cuatro.gameObject.active = false;
Cinco.gameObject.active = false;

// Tapa_1.gameObject.active = false;
// Tapa_2.gameObject.active = false;
// Tapa_3.gameObject.active = false;
// Tapa_4.gameObject.active = false;
// Tapa_5.gameObject.active = false;

if (Play_5 == false && AudioSource.isPlaying == false)
{
    AudioSource.PlayOneShot(Audio_Step5);
    Play_5 = true;
}

if (G_B3.gameObject.GetComponent<Button>().On_Off == true && Play_5 == true && AudioSource.isPlaying == false)
{
    state = 7;
    G_B3.gameObject.GetComponent<Button>().On_Off = false;
}

break;

case 6: // Fail

will need to wait in order to retake the test." + System.Environment.NewLine +

"This is so your nose can reset. If this is the second re-take," + System.Environment.NewLine +

"you need to go back to the tutorial and " + System.Environment.NewLine +

"refamiliarize yourself with the smells."

Button_One.gameObject.active = false;
Button_Two.gameObject.active = false;
Button_Three.gameObject.active = true;
Button_Four.gameObject.active = false;
Button_Five.gameObject.active = false;
Button_Six.gameObject.active = false;

G_B1.gameObject.active = false;
G_B2.gameObject.active = false;
G_B3.gameObject.active = true;
G_B4.gameObject.active = false;
G_B5.gameObject.active = false;
G_B6.gameObject.active = false;

Button_Text_One.GetComponentInChildren<Text>().text = "1st Cup";
Button_Text_Two.GetComponentInChildren<Text>().text = "2nd Cup";
Button_Text_Three.GetComponentInChildren<Text>().text = "Quit";

Countdown.gameObject.GetComponent<TextMesh>().text = "10 seconds";
Countdown.gameObject.active = false;
Results.gameObject.GetComponent<TextMesh>().text = "Wrong answer!";
Results.gameObject.active = false;

Uno.gameObject.active = false;
Dos.gameObject.active = false;
Tres.gameObject.active = false;
Cuatro.gameObject.active = false;
Cinco.gameObject.active = false;

// Tapa_1.gameObject.active = false;
// Tapa_2.gameObject.active = false;
// Tapa_3.gameObject.active = false;
// Tapa_4.gameObject.active = false;
// Tapa_5.gameObject.active = false;
if (Play_6 == false && AudioSource.isPlaying == false)
{
    AudioSource.PlayOneShot(AudioStep6);
    Play_6 = true;
}

if (6_B3.gameObject.GetComponent<Button>().On_Off == true && Play_6 == true && AudioSource.isPlaying == false)
{
    state = 7;
    6_B3.gameObject.GetComponent<Button>().On_Off = false;
}
break;
case 7: //Quit
{
    Debug.Log("Quit!");
    Application.Quit();
    break;
}
default:
{
    break;
}

// if(state == 1)
{
    // AudioSource.clip = AudioStep1;
    // AudioSource.Play();
    // AudioSource.Play(AudioStep1);
    // AudioSource.PlayOneShot(AudioStep1_2);
    //}
    //if (state == 2){
    // AudioSource.clip = AudioStep1_2;
    //AudioSource.Play();
    //AudioSource.PlayOneShot(AudioStep1);
    //AudioSource.PlayOneShot(AudioStep1_2);
    //}
}
Appendix D

Arduino Uno scripting for pump activation.

```cpp
String StringInByte;
int IntByte;

void setup()
{
  Serial.begin(9600);
  StringInByte="";
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
}

void loop()
{
  /* digitalWrite(ledPin, HIGH);  // sets the LED on
    delay(1000);                // waits for a second
    digitalWrite(ledPin, LOW);  // sets the LED off
    delay(1000); */
  if (Serial.available() > 0) {
    // read the incoming byte:

    StringInByte = Serial.readString();
    IntByte = StringInByte.toInt();

    if(IntByte==1){
      digitalWrite(2, HIGH);
      digitalWrite(3, LOW);
      digitalWrite(4, LOW);
      digitalWrite(5, LOW);
      digitalWrite(6, LOW);
      digitalWrite(7, LOW);
    }
  }
}
```
StringInByte="";
}
else if(IntByte==2){
digitalWrite(2, LOW);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
digitalWrite(5, LOW);
digitalWrite(6, LOW);
digitalWrite(7, LOW);
StringInByte="";
}
else if(IntByte==3){
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, HIGH);
digitalWrite(5, LOW);
digitalWrite(6, LOW);
digitalWrite(7, LOW);
StringInByte="";
}
else if(IntByte==4){
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
digitalWrite(6, LOW);
digitalWrite(7, LOW);
StringInByte="";
}
else if(IntByte==5){
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
else if (IntByte==5) {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(7, LOW);
    StringInByte=""
}

else if (IntByte==6) {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(6, LOW);
    digitalWrite(7, HIGH);
    StringInByte="";
}

else if (IntByte==0) {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(6, LOW);
    digitalWrite(7, LOW);
    StringInByte=""
}