Midsemester Report

The implementation of the second generation of a multimodal virtual reality system for small animals

Prepared by Raghav Prasad (2017A7TS0297G)

Under the guidance of Professor Mayank Mehta and Mr Chinmay Purandare

In partial fulfilment of the requirement of the course BITS F421T

Abstract

Electrophysiological recordings in mobile animals in a multimodal real-world environment is challenging and has led to a mediocre understanding of the neural mechanisms of multimodal integration. Thus, a non-invasive experimental setup conducive to electrophysiological recordings is desirable. This was accomplished in 2013 at the W.M. Keck Center for Neurophysics at UCLA, where a custom Virtual Reality setup was independently developed and used to non-invasively record rodents in a highly immersive virtual environment.

This system allows the body of the rodent to be fixed in the real world, without head fixation. The rat is placed on top of a silent, spherical treadmill which provides movement inputs to drive the motion of the rodent in the virtual world. This opens up a multitude of possibilities to subject the rodent to a wide variety of stimuli without the constraints of the real world.

Table of contents

1. Introduction

The current (first) generation of the VR system has been in operation since 2013. The complete development of the system took close to 4 years, starting in 2009. Broadly, the components of the VR setup are:

a. A cylindrical screen onto which the VR world is projected
b. A roughly hemispherical convex mirror which reflects the virtual world image projected onto it to the cylindrical screen
c. A spherical treadmill resting on an air cushion. The treadmill is free to rotate about all 3 axes
d. A data acquisition system for non-invasive electrophysiological recording
e. Arduino boards to control the dispensing of various stimuli

The current (first generation) implementation of the VR system was limited by both the hardware and software available to the researchers at the time. A C++ graphics engine called OGRE, which is now deprecated, formed the crux of the VR software used to generate the virtual environments. The data acquisition hardware had redundancies which can now be eliminated as hardware has become much cheaper. Optic mice were being used to track the movement of the spherical treadmill, which can now be replaced by much more robust solutions using computer vision.

The goal of this project was to improve upon the existing setup and add some new features as well. This was accomplished as follows:

a. Using the Unity game engine to replace the earlier version which used OGRE. Unity uses C# (or JavaScript), so this meant that the entire codebase had to be revamped.
b. Replacing the earlier optic mice implementation of motion tracking with a computer vision-based approach to tracking called FicTrac.
c. Replacing redundant hardware used to synchronize electrophysiological recordings with the VR frame rate with an Arduino.

2. Materials and methods

2.1. Unity

The Unity game engine is an industry-leading piece of software used to make professional games and XR applications. Given the highly robust nature of the engine and the vast amount of online community support it has, it was an obvious candidate to replace the OGRE game engine. Some of the advantages of using Unity are:

a. Includes a GUI to make development easier for non-programmers. This will be of great importance in the maintenance of code since the researchers using the VR application typically won't be from a programming background
b. Includes an online store that provides a platform for a host of third party plugins (called assets) which make development very easy and highly modular. This project makes use of three or four different assets.
c. Includes lots of functions to add and manipulate physics in a very easy manner.

Unity uses C# or JavaScript. To keep the codebase similar to the first generation VR, this project has been written in C#. This is, however, not a simple transpilation from the existing C++ codebase to C#. The reason for this is that the power of the Unity engine allows the code to be much more succinct and it would only be prudent to make use of this by revamping the codebase without sacrificing any functionality, but rather adding to it.

2.2. Blender

Blender is a free and open-source 3D computer graphics software built using Python. Blender was used to construct complex 3D models that are integral components of the VR setup. This is further elucidated in // TODO: Include section number of projection setup and mirror.

2.3. FicTrac

FicTrac (Fictive path Tracking software) is a novel vision-based tracking system for estimating the path an animal makes whilst rotating an air-supported sphere using only input from a standard camera and computer vision techniques[1]. Using FicTrac in a closed-loop configuration, i.e. the output from FicTrac is fed into the VR application to move the rodent in the VR world, the need for optic mice based motion is eliminated. This is good for the following reasons:

a. The current optic mice solution for rodent motion employs circuit boards fabricated in-house at UCLA. This is a highly customized approach. While the need for the reproducibility of this setup does not arise, it is still not as good a solution as one which requires just a camera and a patterned trackball (spherical treadmill).
b. The optic mice solution requires at least 2 mice to cover the entire range of motion that the rodent is allowed to have in the VR world; one mouse for motion in the 2D plane and another for rotation about the vertical axis. Thus, this would require extra code to handle 2 mice simultaneously, on top of retrieving the data from the two mice, making the code even more cumbersome. In contrast, given that FicTrac is a third party software tool, it makes the design much more modular and simple to implement.
c. FicTrac is an open-source project curated by Dr Richard J. D. Moore and receives regular updates from the community. Thus, as time goes on, the FicTrac module of the VR project will receive updates from the community.

2.4. EyeLoop

EyeLoop is an open-source Python-based software that easily integrates custom functions via a modular logic, tracks a multitude of eyes, including rodent, human, and non-human primate eyes, and operates well on inexpensive consumer-grade hardware[2]. EyeLoop is being used in this project to track the rodent's eyes and obtain data such as blinking, pupil dimensions, pupil direction. The current (first generation) solution is a head tracking system where a head-mounted LED is being tracked. Thus, EyeLoop will be a much more robust and accurate alternative to the current solution.

2.5. VR World construction

The construction of the VR world takes place from scratch, entirely using code. Since an experimenter would want to test a number of different hypotheses, it would only make sense to subject the rodent to different stimuli. This is achieved by using a custom world for each different type of experiment. These worlds are called “tracks”. Each track has a different layout and is comprised of different components in different configurations. These tracks are encoded in track files. Track files are an XML-style document that encodes the quantitative and qualitative data associated with the components that constitute a track. Track files are parsed by Unity scripts and are translated into “GameObjects” (the basic entities in Unity).

The VR application allows the experimenter to choose the track file at the beginning of the experiment. This track file is then parsed to produce the virtual world in which the rodent is free to move around.

2.5.1. Mesh Generation

The track usually consists of a room composed of 4 walls and a floor and an elevated surface for the rat to move around on. Thus there are a number of planar GameObjects that need to be produced. These are produced using a technique called “Procedural Mesh Generation”. Procedural mesh generation is a method whereby one can create meshes of any shape and size out of *vertices* and *triangles*, from within a script. These primary meshes can then be used to create more elaborate and complex structures by warping them or composing and building on top of them.

2.5.2. Logging

Along with electrophysiological recording, it is important to log events occurring in the virtual world. To that end, the position of the rat is recorded every frame into a CSV file (see Fig x). Similarly, significant events occurring in the virtual world must also be recorded, events such as the completion of a subset of a foraging task, or receiving a reward.

2.5.3. Dispensers

Dispensers is the generic term used to describe all the components that deliver stimuli to any of the modalities except vision. This includes gustatory stimuli such as sugar water reward, olfactory stimuli such as different odours, and auditory stimuli such as audio clips. The Unity scripts handle the abstract representations (GameObjects) of the dispensers. Their real-world counterparts are essentially valves controlled using Arduinos. The Ardity asset, from the Unity Asset store, is used to synchronize the behaviour of the virtual GameObject dispenser and the real world dispenser (see Figure 1).

2.6. Projection setup and predistortion

The virtual world is projected on to a cylindrical screen of radius 36 centimetres and height 75 centimetres, which covers a horizontal field of view of 330 degrees. Thus, a flatscreen projector would not be able to cover this. The solution to this was to use a roughly hemispherical convex mirror fixed at the centre of the top of the cylindrical enclosure and have a regular flatscreen projector project onto it. The image would then be reflected off the mirror and onto the cylindrical screen (see Figure 2). This would work except that when the flatscreen image, i.e. the image of the virtual world as we see it on a flat screen, is projected directly onto the mirror, the mirror will distort this image and we will get a distorted image on the cylindrical screen. Thus, the flatscreen image of the virtual world must be transformed such that after reflection off the mirror, it appears undistorted on the cylindrical screen. This transformation is what we are calling predistortion (see Figure 3)

In order to achieve this predistortion, a virtual setup was created in the virtual world, to exactly mirror the real setup at the lab, i.e. a cylinder of matching dimensions and an overhead mirror was constructed using Blender and procedural mesh generation respectively. Positioned at a height of 2 inches (~ 5.08 centimetres) from the centre of the base of the cylinder, to coincide with the position of the rodent's eyes, is an array of 6 virtual cameras, each with a vertical and horizontal field of view of 90 degrees (see Figure 4). The view from each of these virtual cameras is then projected onto the inside of the virtual cylinder, each of them projecting it onto

the region of the cylinder that falls within the camera frustum of that camera. Thus, in order to ensure that the projection of the view of each camera is on the correct region of the cylinder, the cylinder is cut along the points of intersection of the camera frustum with the cylinder (see Figure 5). Once the projection of the camera views is obtained on the inside of the virtual cylinder (see Figure 6), the virtual mirror will now reflect this and there is one virtual camera pointed at the mirror from underneath, to mimic the real projector projecting onto the mirror (see Figure 7). Thus, we are using the principle of reversibility of the path of light in order to perform the predistortion.

2.7. Neuralynx Cheetah data acquisition system

Neuralynx Cheetah is the hardware system responsible for recording the neural signals from the rodent during the experiment. The VR application will interface with this using TTL pulses via an Arduino. The neural recording data timestamps and the corresponding TTL port values are recorded in a binary format in an “event” file generated by the Neuralynx system. The VR application logged data is later synchronized with the Neuralynx generated data offline to prepare all the data for analysis.

3. Conclusion

This project was aimed at improving the current generation of the VR system and add new features to the same. The core graphics engine and the codebase have been revamped to produce a more efficiently performing VR application. The motion tracking system and the eye-tracking system are upgraded to use computer vision-based tools. The neural recording system has been upgraded to use less hardware in an efficient manner.

Future improvements can include an increased variety of tracks, more and varied stimuli such as virtual subjects like a person walking in the virtual world, video cues and so on. The projection setup and predistortion also have scope for improvement and could use a raycasting solution in lieu of the solution proposed in this project. This would eliminate the need for creating a special cut cylindrical mesh and could instead use an ordinary cylinder.

4. Acknowledgements

I would like to thank Professor Mehta for giving me the opportunity to work on this project and for the guidance he has provided throughout the duration of the project. I would also like to thank Mr Chinmay Purandare for being an excellent mentor, for his able guidance and patience and for always leading me towards a solution whenever I hit a mental block and did not know how to proceed.

I would like to acknowledge Professor Sujith Thomas for agreeing to be my supervisor from BITS and also Professor Bharat Deshpande, for overseeing the off-campus thesis program at BITS which has presented me with this opportunity.

Lastly, I would like to express my gratitude for the open-source community and the members of various forums for being prompt and responsive towards some the queries I posted when I had difficulty using some of the software tools. Special thanks to Dr Richard J. D. Moore for helping me through the challenges I faced during the installation of FicTrac. And also to the team developing and maintaining EyeLoop who have been prompt in addressing the concerns I posed regarding bugs in the software.

5. Bibliography

[1] Moore, R. J. D., Taylor, G. J., Paulk, A. C., Pearson, T., Swinderen, B. v., & Srinivasan, M. V. (2014). FicTrac: a visual method for tracking spherical motion and generating fictive animal paths. *Journal of neuroscience methods*, *225*, 106-119.

[2] Arvin, S., Rasmussen, R., & Yonehara, K. (2020, July 04). EyeLoop: An open-source, high-speed eye-tracker designed for dynamic experiments. *bioRxiv*.

6. Figures and tables

timestamp	player_x	player_y	player_z
13:01:19.0942	5.034976	2	8.377412E-21
13:01:20.0893	6.63902	2	7.78929
13:01:21.1041	8.985002	2	7.90052
13:01:22.1093	-1.065781	2	7.90052
13:01:23.1664	-8.389506	2	7.90052
13:01:24.1724	-13.35973	2	7.457509
13:01:25.2501	-13.35973	2	-1.629274
13:01:26.2604	-5.318634	2	-1.701439
13:01:27.2597	0.7710616	2	-5.719994
13:01:28.2804	2.851722	2	-9.741574
13:01:29.3533	-6.106517	2	-7.979264
13:01:30.3911	-4.739483	2	0.9938482
13:01:31.4194	4.617547	2	2.254708
13:01:32.4757	1.73891	2	7.850473
13:01:33.4957	0.2158394	2	-0.1031206
13:01:34.5198	0.6087957	2	-0.6171932
13:01:35.5243	0.6087957	2	-0.6171932

Table 1: Log of position data of the rodent over time

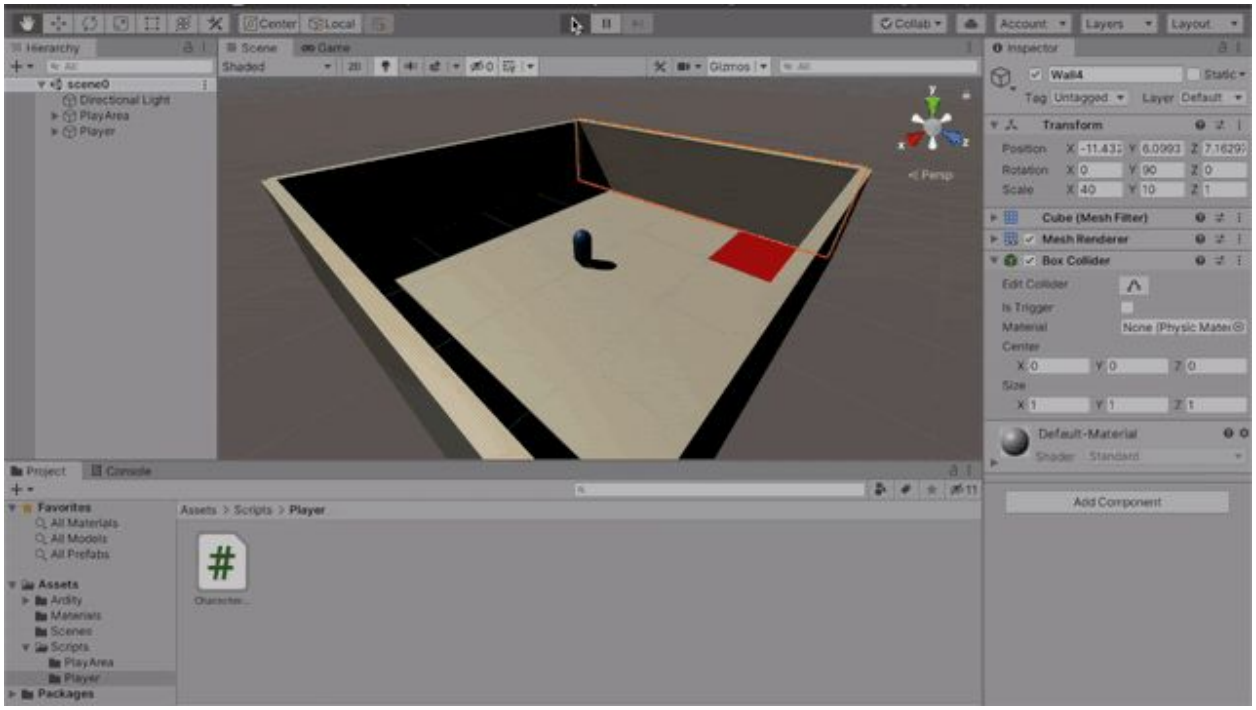

Figure 1: A dispenser being triggered when the rodent enters the reward zone (shown in red). Inset: An Arduino Uno that lights up an LED when the rodent enters the reward zone. The LED is a proxy for any type of dispenser.

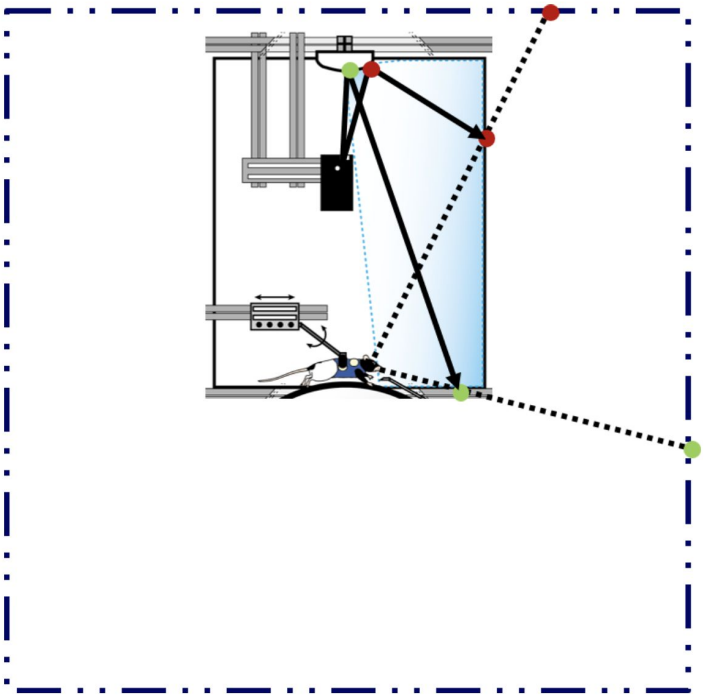

Figure 2: Projector, mirror and screen setup

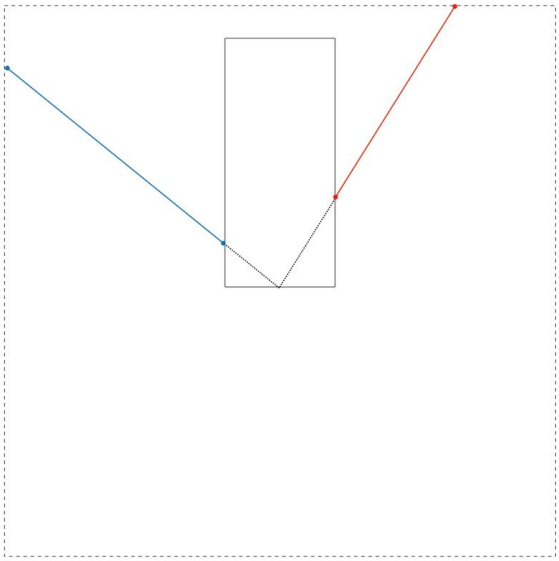

Figure 3: Predistortion; mapping the virtual world onto the virtual cylinder. This is a cross-sectional view of the virtual world + hollow cylinder setup. The red pixel is projected from the cubical room to the cylinder such that the extended ray's path passes through the centre of the base of the cylinder. Similarly for the blue pixel.

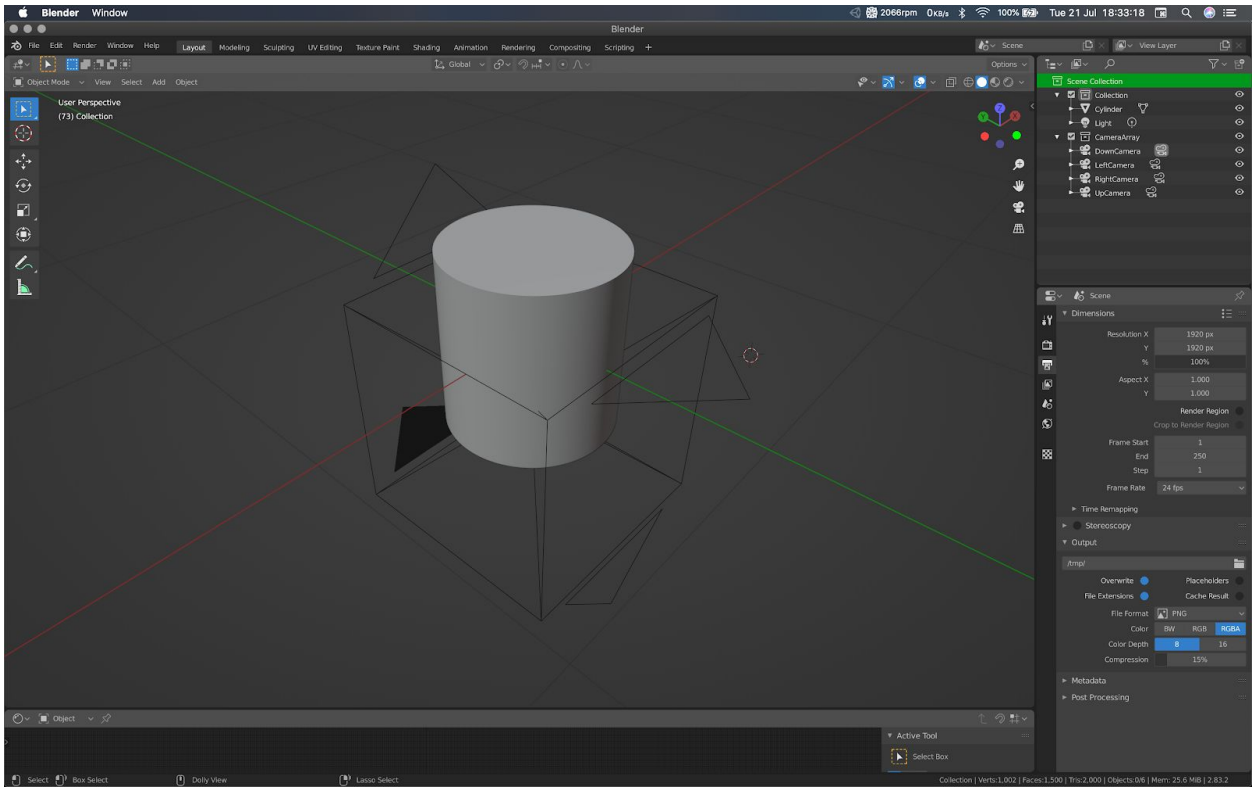

Figure 4: Virtual cylinder setup with 6 virtual camera array

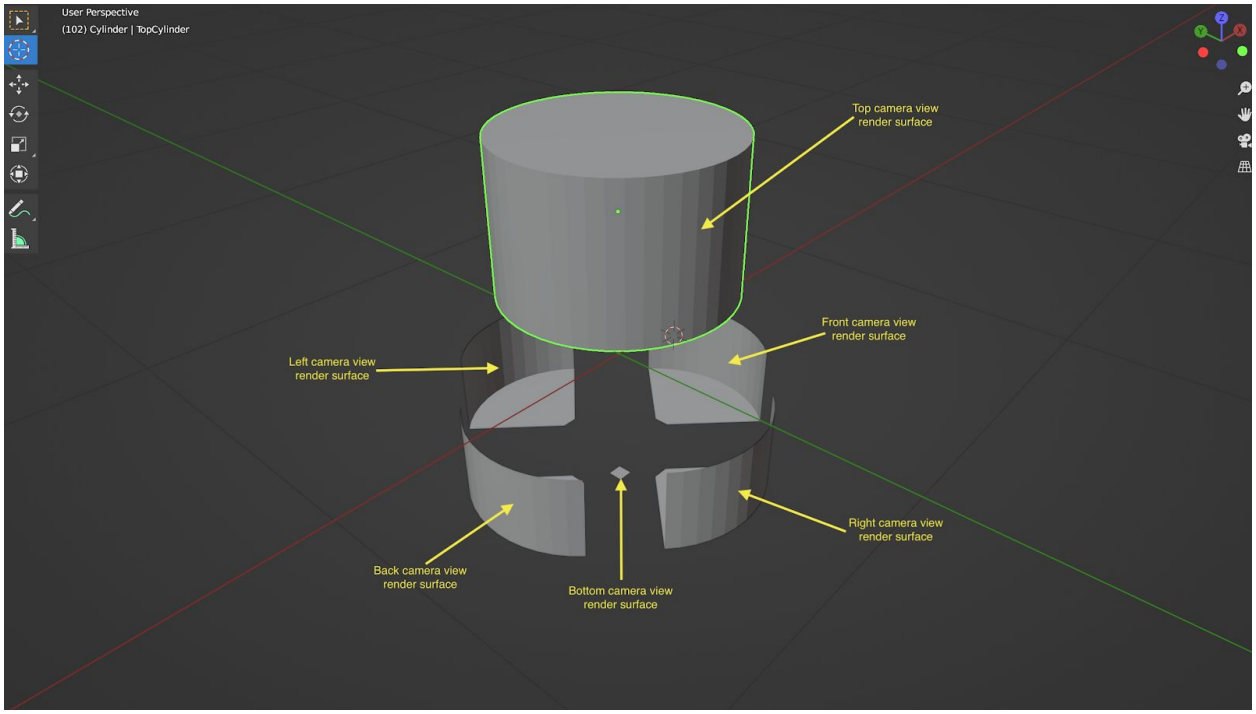

Figure 5: Separation of the cylinder into discrete regions for each camera to project its view

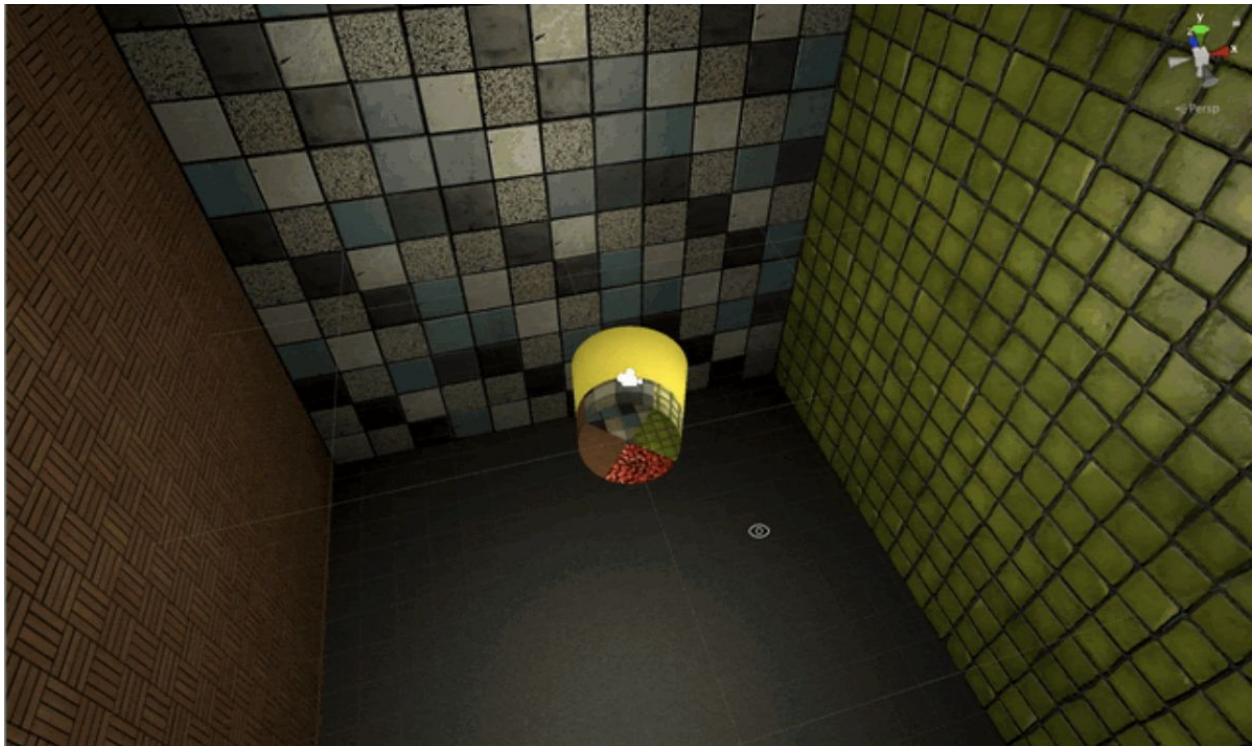

Figure 6: Projection of the world onto the inner surface of the virtual cylinder

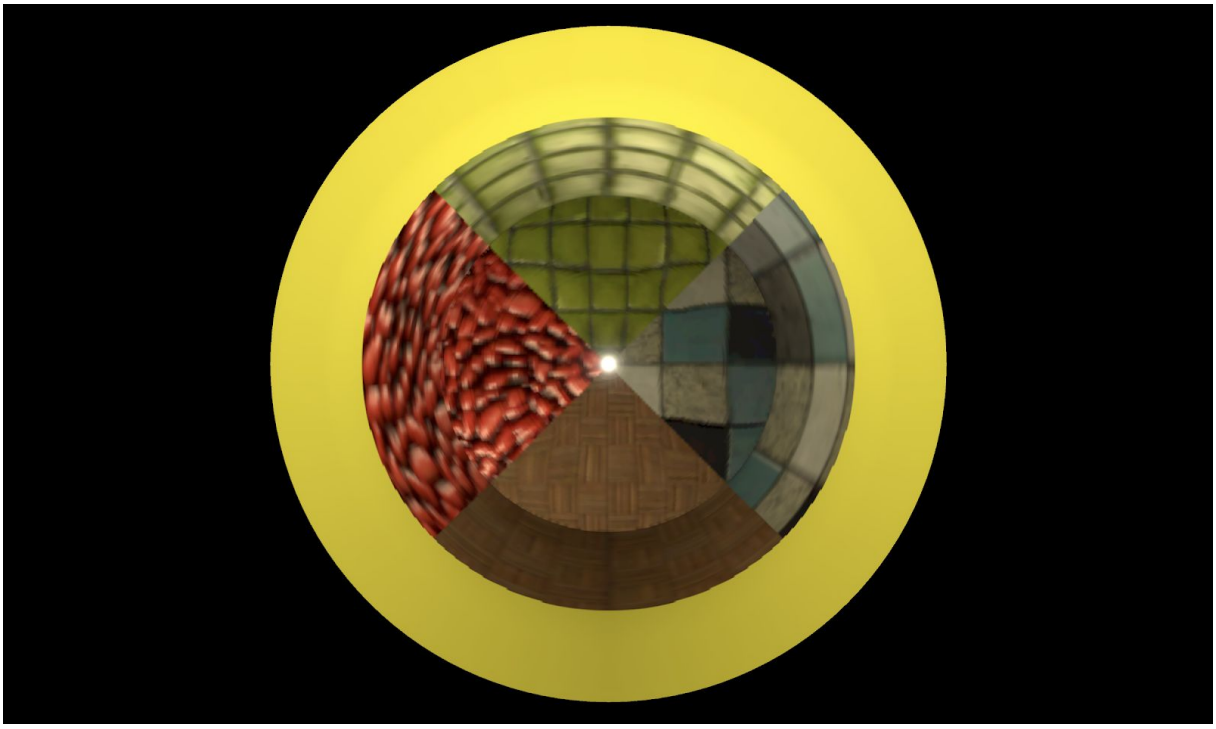

Figure 7: The view of the reflection in the virtual mirror, corresponding to the projection obtained in Figure 6