UNIVERSITY OF WEST ATTICA
SCHOOL OF ENGINEERING
DEPARTMENT OF INFORMATICS AND COMPUTER ENGINEERING

THESIS

«Turning stigma into Awareness: The development of a serious game for Bipolar Disorder»

Anna Sfakioti
A.M. 19390223

Εισηγητής: ΙΩΑΝΝΗΣ, ΒΟΓΙΑΤΖΗΣ, ΚΑΘΗΓΗΤΗΣ

Turning stigma into Awareness: The development of a serious game for Bipolar Disorder

Anna Sfakioti
A.M. 19390223

ΕΙΣΗΓΗΤΗΣ:

IOANNIS, VOYIATZIS, PROFESSOR

Ioannis Voyiatzis

Digitally signed by Ioannis Voyiatzis
Date: 2024.10.22 17:31:33 +03'00'

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

KLEIO, SGOUROPOULOU, PROFESSOR

KLEIO SGOUROPOULOU

Digitally signed by KLEIO SGOUROPOULOU
Date: 2024.11.11 17:34:08 +02'00'

CHRISTOS, TROUSSAS, ASSISTANT PROFESSOR

Christos Troussas

Digitally signed by Christos Troussas
Date: 2024.10.22 18:01:24 +03'00'

Ημερομηνία εξέτασης
07/10/2024

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

«Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

«Βεβαιώνω ότι είμαι συγγραφέας της παρούσας διπλωματικής εργασίας και ότι έχω αναφέρει ή παραπέμψει σε αυτή, ρητά και συγκεκριμένα, όλες τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, προτάσεων ή λέξεων, είτε αυτές μεταφέρονται επακριβώς (στο πρωτότυπο ή μεταφρασμένες) είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για την συγκεκριμένη διπλωματική εργασία»

Ο/Η Δηλών/ούσα
ΣΦΑΚΙΩΤΗ ΑΝΝΑ

ACKNOWLEDGMENT

This thesis is the result of continuous research, discovery, and support from both my personal and academic environments. I would like to personally thank my tutors who embraced my idea from the very first moment and provided me with everything I needed, as well as my chosen family who stood by my side in my personal battle with Bipolar Disorder. Lastly, I dedicate my thesis to Katerina, who, even though she lost her own battle with the disorder, helped pave the way for the rest of us.

ΠΕΡΙΛΗΨΗ

Σκοπός του παρόντος εγγράφου είναι η εις βάθος θεωρητική ανάλυση και επεξήγηση του εκπαιδευτικού ηλεκτρονικού παιχνιδιού «Bipolar Odyssey» που αναπτύχθηκε από εμένα στα πλαίσια της εκπόνησης της διπλωματικής μου εργασίας στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Υπολογιστών του Πανεπιστημίου Δυτικής Αττικής. Η διπολική διαταραχή, παλαιότερα γνωστή ως μανιοκατάθλιψη, θεωρείται μια ψυχική ασθένεια που επηρεάζει περίπου το 3-4% του παγκόσμιου πληθυσμού. Αν και οι πρώτες περιπτώσεις παρατηρήθηκαν περίπου τον 18ο αιώνα[1], η Διπολική Διαταραχή εξακολουθεί να παρουσιάζει σημαντικές ερευνητικές ελλείψεις, γεγονός που συμβάλλει στον στιγματισμό και κίνδυνο των πασχόντων, με τη συμπτωματολογία να είναι αρκετά σοβαρή σε περιπτώσεις ώστε να προκαλεί ακραία δυσφορία στην καθημερινότητα των ασθενών. Περαιτέρω σε αυτή τη διατριβή, θα εμβαθύνω στα συμπτώματα, τους τύπους και τις συνολικές προκλήσεις του διπολικού φάσματος, για να αναδείξω πώς η τεχνολογία και τα εκπαιδευτικά ηλεκτρονικά παιχνίδια, στα οποία στο εξής θα αναφέρομαι ως serious games, μπορούν να χρησιμοποιηθούν για να εξυπηρετούν καλύτερα όσους μάχονται με τη διαταραχή, τον κοινωνικό τους περίγυρο και τους επαγγελματίες ψυχικής υγείας. Επιπλέον, θα αναλυθεί μια νέα προσέγγιση του serious game και του διπολικού φάσματος, μέσω της θεωρητικής επεξήγησης της δημιουργίας του «Bipolar Odyssey». Ο κύριος σκοπός του συγκεκριμένου παιχνιδιού είναι να αυξήσει την ευαισθητοποίηση και να λειτουργήσει ως χείρα βοηθείας για όλα τα άτομα που σχετίζονται με το διπολικό φάσμα, παρέχοντας ένα ελεγχόμενο περιβάλλον στο οποίο οι ασθενείς μπορούν να εξοικειωθούν με τη συμπτωματολογία της νόσους τους. Προκειμένου η εμπειρία του χρήστη να γίνει πιο διαδραστική, η μορφή του 3D serious game εξελίχθηκε σε VR experience, με την χρήστη του VR εξοπλισμού Oculus Quest 3, ο οποίος παρασχέθηκε σε εμένα από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Υπολογιστών. Τέλος, η συγκεκριμένη προσέγγιση του serious game φαίνεται να εξετάζει ένα σημαντικό ερευνητικό κενό που αφορά τα φυσικά συμπτώματα της νόσου, θέτοντας το upscalling του παιχνιδιού πολλά υποσχόμενο.

ABSTRACT

The present thesis is centered on the creation of a serious game that accurately depicts the altered sensory phenomena that occur when an individual diagnosed ιν the Bipolar Spectrum is in a Hypomanic state. Bipolar disorder, previously known as manic depression, is considered to be a mental illness that affects around 3-4% of the world population. While the first cases were noted around the 18th century [1], Bipolar Disorder still lacks adequate research, a fact that contributes to the stigmatization and endangerment of those affected, with the symptomology severe enough at cases that extreme discomfort can be caused in the patients' everyday life. Further in this thesis, the symptoms, types, and overall challenges of the Bipolar Spectrum are going to be delved into, to highlight how technology and serious gaming can be used to better assist those who battle the disorder, their social circle, and mental health professionals. Also, a new approach to serious gaming and the Bipolar Spectrum will be analyzed, through the creation and explanation of the created serious game, ”Bipolar Odyssey“. The main purpose of this particular game is to raise awareness and act as a helping hand not only for individuals associated with the Bipolar Spectrum, but also their social circle and even mental health professionals, by providing a controlled environment in which patients can familiarize themselves with the symptomatology of their condition. In order to make the user experience more interactive, the 3D serious game evolved into a VR experience, using Oculus Quest 3 VR equipment, which was provided to me by the Department of Informatics and Computer Engineering, of the University of West Attica. Finally, this approach to serious gaming seems to address an important research gap concerning the physical symptomatology of the Bipolar Spectrum, making the games’ upscaling promising.

Scientific Domain: Computer Science, Psychology and Cognitive Science
Keywords: Bipolar Disorder, Hypomania, BP II, Serious Gaming, Altered Sensory Phenomena, Bipolar Eyes

Contents

FIGURES

TABLES

I. INTRODUCTION

Bipolar Disorder (BP) is a chronic, incurable mental health disorder that triggers sudden and intense shifts in one's mood, energy, and overall abilities, affecting greatly their social, personal, and work life[2]. The symptomatology that makes the disorder so distinct and difficult to analyze is the fact that it consists of two emotional disorders; one being Depression and the other Mania/Hypomania. During the course of the 19th century[1], Parisian psychiatrist Jules Baillarger was the first to notice, during his clinical observations of patients, the fact that some were affected by alternating depressive and euphoric episodes, while a period of regular ambiance in between symptoms was also present. As years went by and the disorder was further analyzed, Emil Kraepelin presented the broad term "Manic-Depressive Insanity" at the end of the same century, leading examiners in the late 1970s to the conclusion that Bipolar Disorder can be broken down into two notable categories: Bipolar Disorder Type I and Bipolar Disorder Type II[1]. In the last few years, the term Manic-Depression was replaced by Bipolar Disorders or Bipolar Disorder Spectrum, and the diagnosis currently used are: Bipolar Disorder Type I, Bipolar Disorder Type II, Cyclothymia, and Bipolar Disorders Not Otherwise Specified. Concentrating on the fact that patients with BP have to face the reality of, sometimes even daily, intense turmoil in their mood and overall stability, while also facing difficulties getting their diagnosis early on, renders them a high-risk group. Based on research conducted in the latest years, individuals with BP are at greater risk of self-harm and suicide[3]. Thus, it is deemed of great importance for technological assistance to be provided, not only as a helping hand for the patients but also as a driving force in eliminating years of trauma and generational stigma. While examining in what ways technological assistance could be of use for individuals within the Bipolar Spectrum, a new approach to serious gaming was inspired; a VR experience that focuses solely on the physical symptomatology of the Spectrum, specifically during Hypomania. Therefore, the serious game named ‘Bipolar Odyssey’ was created, which will be further examined in this paper.

II. SYMPTOMATOLOGY

As previously mentioned, the Bipolar Disorder Spectrum can be categorized into 4 different subcategories, based on the different symptomatology that a person exhibits. The 4 types of symptoms that fit in the Bipolar Spectrum are[4], [5]:

1. Bipolar Disorder Type I (BD I): Major Depressive Episodes, alternating with at least one Manic Episode[4].
2. Bipolar Disorder Type II (BD II): Major Depressive Episodes, alternating with at least one Hypomanic Episode[4].
3. Cyclothymic Disorder: Periods of Depressive and Hypomanic symptoms, with depressive symptoms not meeting the criteria for Major Depressive Disorder (MDD)[4].
4. Bipolar Disorder Not Otherwise Specified: Periods of Depressive and Hypomanic-like symptoms that may alternate rapidly, but do not meet the full diagnostic criteria for any of the above diagnoses[4].

In this segment, the 4 different diagnoses of the Bipolar Spectrum are going to be shortly analyzed.

A. Bipolar Disorder Type I

The first subcategory of the Bipolar Spectrum is Bipolar Disorder Type 1, a condition that is characterized by Major Depressive Disorder (MDD) and Manic episodes. The diagnostic criteria for BP I require at least one episode of Mania, or a case of a mixed episode, where both depressive and manic symptoms occur simultaneously. In most cases, a depressive episode has transpired before the first manic episode of a BP I individual[6], while the manic symptoms need to be present for a period of at least a week for the diagnostic criteria to be met.

B. Bipolar Disorder Type II

The second subdivision noted in the Bipolar Spectrum is Bipolar Disorder Type II. To be diagnosed with BP II, one has to display hypomanic symptoms for at least a period of 4 days, with their mood disturbance observed by others. Similarly to BP I, patients with BP II usually experience depressive

episodes earlier on than hypomanic episodes and since hypomanic behaviors can frequently get confused with overcoming a Major Depressive Episode, the rate of misdiagnosis in BP II is particularly high, reaching around 60% of all cases[4], [5], [7].

C. Cyclothymic Disorder

Cyclothymic Disorder is considered the third branch of the Bipolar Spectrum. In this case, the individual endures multiple periods of hypomanic episodes, intertwined with phases characterized by depressive symptoms, not severe enough to be categorized as a Major Depressive Episode[4].

D. Bipolar Disorder Not Otherwise Specified

Lastly, when one experiences depressive and hypomanic-like symptoms, that can alternate quite rapidly, but do not meet the above criteria for BP I, BP II, or Cyclothymic Disorder, the diagnosis of Bipolar Disorder Not Otherwise Specified is considered. This particular diagnosis is commonly used when the patient suffers from more than one mental disorder (e.g. Borderline Personality Disorder) [6].

The most common difficulty faced by BP individuals is receiving the appropriate diagnosis and medical attention in time. As previously mentioned, disorders within the Bipolar Spectrum can be quite difficult to detect, since symptomatology can be similar to other disorders and sometimes overlap one another[6]. For example, in most bipolar patients, symptoms can resemble other mental health complications, such as Borderline Personality Disorder, Panic and Anxiety Disorders, and most importantly Major Depressive Disorder (MDD). Especially considering BP II, Hypomania can be particularly difficult to identify, since patients in this condition lack insight to their disorder. Thus, the majority of individuals eventually diagnosed with BP II have wrongfully gotten an MDD diagnosis earlier on in their lifetime. All of the above can lead mental health experts to initially prescribe medication used for treating Unipolar Depressive Disorder, a treatment that when used by Bipolar patients can lead to Mixed Mania and/or Manic switches[8]. Another cause of misdiagnosis in the Bipolar Spectrum is the fact that depressive symptoms appear to be present considerably more often compared to Manic/Hypomanic ones. It should also be stated that even though BP I affects all genders with equal frequencies, BP II appears more commonly in assigned female individuals, a group that also seems more likely to experience Manic/Hypomanic outbursts when prescribed medication for Unipolar Depression treatment[8].

III. CAUSES

In this segment, a brief explanation of the disorders' causes will be presented. A standstill that researchers have reached over the passing years while delving into the perplex world of the Bipolar Spectrum is the fact that the disorder appears to be the result of numerous factors. The most prominent amongst them are[5]:

A. Genetic Factors

Bipolar Disorders in general can be attributed to strictly hereditary reasons, with 10 to 20% of patients diagnosed also mentioning a first-degree relative that has experienced similar symptomatology.

B. Chemical Imbalances

While analyzing the biochemical background of Bipolar Disorders, researchers have concluded that in individuals with BP, the neurotransmitters responsible for serotonin, dopamine, and noradrenaline secretion seem to frequently exhibit either hyperactivity or hypoactivity, creating fluctuations in mood stability.

C. Triggers

Recent analysis has proven that BP can be developed throughout one's early years, especially when growing up in abusive households or faced with harrowing occurrences, such as childhood sexual abuse, loss of a parent or close relative, and overall heightened anxiety.

D. External Factors

Lastly, multiple cases of manic outbursts caused by alcohol and substance abuse have been noted. By using, patients risk experiencing psychotic episodes and therefore should be strictly advised against substances such as alcohol, amphetamines, cocaine, opioids, etc.

IV. TREATMENT

While analyzing the appropriate treatment for Bipolar Disorders, the 2 main categories that have proved successful should be highlighted, them being: A. Pharmacotherapy and B. Psychotherapy.

A. Pharmacotherapy

Since in most cases, Bipolar Disorders can be traced back to chemical imbalances noted in the patients' neurotransmitters, pharmacological treatment has proved fairly effective in preventing future episodes and therefore resulting in remission. Throughout the course of analyzing the spectrum, a handful of different medications have been put to the test, some of them being:

- Lithium
- Conventional Antidepressants
- Atypical Antipsychotics

Collectively, the main agents used in combating Bipolar Disorders are Lithium, a combination of Olanzapine and Fluoxetine, Quetiapine, and Lurasidone[2]. Unfortunately, medication nonadherence is very common and an issue of great importance that occurs in individuals within the Spectrum. The reasons behind it can vary and most of them are tightly connected to the stigmatization surrounding mental health matters, a fact that will be further discussed later on[4].

B. Psychotherapy

A modern-day approach to coping with Bipolar Disorders is the combination of pharmacotherapy and psychotherapy. Once the patients' symptoms have subsided due to prescribed medication, psychotherapy could be of great help in maintaining a healthy day-to-day lifestyle, with a proper sleeping schedule, avoidance of substance abuse, and prudent decision-making. In addition, addressing the patients' triggers can assist in preventing future relapse. Numerous psychological approaches can be applied, depending on the patients' needs and traumas, such as[5]:

- Behavioral Therapy
- Psychoanalysis.
- Supportive Psychotherapy
- Group Therapy
- Family Therapy

V. BIPOLAR SPECTRUM AND DANGERS

A. Altered Sensory Phenomena

Although over the course of the last few years discussion around the Bipolar Spectrum has begun, very little is shared about the altered sensory phenomena patients experience. A considerable number of individuals in the Bipolar Spectrum have mentioned sensory changes during an episode that can result in visual and auditory hallucinations, increased sensory sensitivity, and even an altered sense of danger. Additionally, some have also reported alterations in their taste and sense of smell, both in depressive episodes and during mania/hypomania [9]. For most, the main point of interest is the relation between hypomania and sleep, as it is academically proven that insomnia can lead to significant dysfunction of the Circadian Rhythm. The Circadian Rhythm is responsible for the regulation of one's sleep cycle, hormonal secretion, and mood as a whole[10], and researchers have deemed that Circadian Rhythm dysfunction, which is the result of sleep disturbances, heavily impacts mood disorders in general. Therefore, a patient with BP is not only psychologically, but also physically impacted, due to sleep

deprivation. Recent analysis has shown that sleep disturbances can lead to auditory hallucinations and errors, as well as inaccurate image formation on the individual's retina, resulting in blurry and dim vision[11]. As a result, particularly during Manic/Hypomanic episodes, the patient can be exposed to grave danger even in their day-to-day activities.

B. Stigma

A fairly common social phenomenon that affects not only patients within the Bipolar Spectrum but individuals with mental health complications as a whole is stigmatization. Stigma is the result of inadequate knowledge and prejudice considering mental health, leading to discrimination and social avoidance[12]. Due to psychotic symptoms that are common in individuals within the Bipolar Spectrum, they are commonly perceived as a danger to themselves and others and are thus isolated and excluded from typical social activities. Moreover, since the symptomatology of the spectrum includes severe and sudden shifts, a common misconception implies that patients are unable to maintain long-term relationships, careers, and everyday habits. One of the main reasons mental health stigma still has a significant negative impact on the patient's psyche is how deeply rooted it truly is. Especially considering earlier years, one with mental health complications would endure psychological and physical abuse, primarily from their caretakers, and be completely shut off from social engagements. While examining the deeply alarming case of the Lero's Asylum, where patients were chained to their bedposts, hosed down by staff daily, and living in extremely unhealthy and inhuman conditions, one can conclude that patient cruelty is a phenomenon that still took place in Greece merely four decades ago[13]. Consequently, it is of great difficulty to revoke years of generational trauma and misinformation and thus eliminate the stigma. A rather interesting inquiry is how patients with Bipolar Disorders internalize society's negative mannerisms concerning their condition, bringing about self-stigmatization [14]. When faced with constant harmful societal stereotypes, the individual tends to form a distorted self-image, deeming themself incapable of experiencing a conventional day-to-day lifestyle. For this exact reason, the acceptance of a BP diagnosis can be a particularly challenging and demanding procedure. Another significant impact of self-stigmatization is medication nonadherence. In most cases of BP diagnosis, the patient struggles to face their new reality, leading to denial and thus considering pharmacotherapy not necessary or effective. Furthermore, many tend to abuse their prescribed medication by using a higher dose. As expected, medication nonadherence, as an immediate outcome of self-stigmatization, can cause severe relapse and even hospitalization[4]. Lastly, the Bipolar Spectrum is directly linked to increased risk of self-harm with 37% of adolescents with BP indulging in some form of it, and suicide, with attempts among BP-diagnosed individuals reach a whooping 25-60%, and suicide completion at 14- 60%[4].

VI. AWARENESS

Awareness around mental health struggles has significantly increased in recent years due to public health initiatives, advocacy by influential figures, and the growing presence of mental health topics in media. Initiatives like 'World Mental Health Day' and 'World Bipolar Awareness Day' are crucial in reducing stigma and promoting understanding by providing opportunities for educational campaigns, community events, and media coverage[15]. Influential figures sharing their mental health experiences help normalize these discussions and encourage others to seek help, aiming for a more inclusive and supportive environment. This has enhanced public empathy and understanding towards those struggling with mental health conditions. However, these disclosures can sometimes lead to sensationalism or trivialization of serious conditions. Media coverage might focus more on personal drama rather than broader mental health issues[15]. If not handled sensitively, these stories can perpetuate stereotypes or misinformation and may even lead to backlash or discrimination against the individuals sharing their experiences. While increased awareness and open discussions about mental health are positive steps, it is essential to approach these topics with sensitivity and accuracy. Comprehensive education and ongoing public health efforts are necessary to ensure that awareness translates into understanding, support, and meaningful change in societal attitudes and policies regarding mental health.

VII. MEDICAL FIELD AND SERIOUS GAMING

A. Serious Gaming and Medical Education

In medical education, serious games have proven to be highly effective tools. They offer interactive and immersive environments where medical students and professionals can practice procedures, diagnose conditions, and make critical decisions without the risk associated with real-life scenarios. For instance, the game "Re-Mission" helps young cancer patients understand their treatment by letting them combat cancer cells in a virtual environment, fostering a sense of control and involvement in their treatment process[16], [17]. Another example is "Operation Quest", designed to reduce anxiety in children before undergoing surgeries by familiarizing them with the hospital environment through engaging narratives and interactive challenges[16]. Several serious games are specifically designed for medical applications, such as "EndeavorRx," the first FDA-approved game used to treat children with ADHD by improving their attention through gameplay. "CliniPup" helps reduce perioperative anxiety in children by familiarizing them with surgical procedures and environment through an engaging virtual setting[16]. Another notable game is "GlucoZor," which educates children with diabetes on how to manage their condition by integrating educational content into a virtual pet care scenario[16]. Considering all the above, serious games have become valuable tools in both education and medicine, by offering innovative ways to learn, practice, and manage various aspects of health and education and leveraging technology to create engaging and effective learning experiences. The ongoing development and research in this field promises to further enhance the impact of serious gaming in improving educational outcomes and healthcare delivery.

B. Serious Gaming and Mental Health

Serious gaming has also emerged as a significant tool in mental health, offering both educational and therapeutic benefits. These games provide innovative, immersive experiences that combine entertainment with learning, helping to reduce stigma and offering therapeutic benefits to individuals with mental health disorders. Clinical applications like the European project PlayMancer[18] target behavioral and emotional processes in patients with impulse-related disorders, using biofeedback and emotion recognition to teach relaxation and self-control strategies. Serious games have also been integrated into anti-stigma programs[19], demonstrating their ability to educate the public and foster empathy by illustrating the experiences of those with mental health conditions. Additionally, targeted interventions using serious games have been applied to ADHD, schizophrenia, and anxiety disorders, providing controlled environments for practicing coping strategies and improving emotional regulation[18]. Despite the promising outcomes, further research and development are needed to validate these findings and expand their use in mental health treatment. Comprehensive research should focus on integrating psychological and physical symptom management to offer a holistic approach to care. Overall, serious games represent a valuable tool in mental health treatment and education, combining technology with therapeutic practices to support patients, reduce stigma, and improve mental health care quality[19]. The ongoing development of these tools holds significant potential for the future of mental health treatment.

VIII. BIPOLAR SPECTRUM AND SERIOUS GAMING

An engaging inquiry is, in nowadays society, how can technology better assist such individuals to not only get their diagnosis in time but also inform others about their condition. Thus far, apart from online tests that can calculate the possibility of a bipolar-related disorder based on one's symptomatology, research has shown promising results concerning serious gaming[20]. In particular, serious gaming is defined as a form of interactive computer application and its main intention is educating while combined with entrainment, creating a whole new genre called edutainment. This multifaceted approach renders them an effective tool for disseminating information to a wide audience, particularly in addressing complex issues such as trauma and generational stigma. More specifically, in recent years, a spike in the usage of electronic resources has been noticed, with research demonstrating similar results to direct contact with individuals suffering from mental health struggles[19]. Moreover, individuals who partake in serious gaming have reported stigma-related emotions toward the patients,

e.g. fear, to subside after the experience[19]. Adding to the benefits associated with serious gaming, virtual reality experiences are accessible and can be used for exposure therapy as a complimentary tool, in order to trigger the patient in a safe space, with controllable stimuli and professional guidance. Thus, the patient can control their responses, examine them, and proceed to work on them[18]. It should be noted though, that the aforementioned privileges are brought about when combining the serious gaming experience with adequately informing the user beforehand[21]. In the following sections, the already existing serious games that have been developed will be examined.

A. BIPOLIFE

Based on research funded by AstraZeneca[19], BIPOLIFE is the first serious game I would like to delve into. This particular serious game revolves around an avatar, diagnosed with BP, and the whole objective is the regulation of the avatar's emotions, by completing daily tasks. It was designed in collaboration with Ubisoft and the main target group it reaches is patients with BP through the act of decision-making, making the game mainly goal-oriented. It emphasizes[22] the importance of maintaining healthy habits, such as the regulation of sleeping habits, exercise and avoidance of substance abuse, therapy participation in case of an episode, and adherence to the appropriate prescribed medication. Though the researchers were blinded during the whole process, the participants were not, and for a time period that lasted a whole month, they were encouraged to engage as much and for as long as they wished. Before engaging with BIPOLIFE, the controlled group took part in a 12-week-long psychoeducation program[20]. Summarizing all of the above, this particular serious game is a great teaching tool and insight into the daily whirlwinds of living with BP, having to make diminutive but crucial decisions, while simultaneously highlighting the importance of oversight, either by a trained mental health professional, or a psychiatrist.

B. Stigma-Stop

Stigma-Stop is the second serious game I would like to further examine. To begin with, the game in question was funded by the Government of Andalusia, in collaboration with the University of Almeria Alborada Engineers, and with the support of the Andalusian Public Federation for Social Integration of People with Mental Disorders. Unity3D was used for the development of said game, both in a mobile app and web-accessed[21]. Opposed to BIPOLIFE, Stigma-Stop grapples with 4 different mental illnesses, specifically depression, agoraphobia, schizophrenia, and lastly BP. The game revolves around an avatar, who in collaboration with their four friends needs to create a video game for a contest they have taken part in. What seems interesting in this particular game is the fact that each of the different characters is diagnosed with one of the above-mentioned mental illnesses. Thus, the player needs to make prudent decisions, for the team to cooperate, while respecting and taking into consideration the various symptomatology of each condition[21]. Moreover, the game is highly interactive, as the player is given feedback for each of their choices, as well as for the options they did not conclude on. Additionally, after the interaction with each character has ceased, the player is presented with 4 individual mini-games, each one focused on a different mental health concept[19]. For instance, in one of the mini-games, the player needs to pair up various renowned personalities, such as Leonardo Da Vinci, based on the information given surrounding the disorder the individual was characterized by. In this way, the fact that despite their mental difficulties, individuals with mental illnesses can contribute greatly to society is highlighted. Lastly, in the case that the player has reached a standstill, the game is designed to give appropriate feedback to better guide them[21].

IX. A NEW SERIOUS GAME APPROACH

A. Initial Approach

Taking into consideration all the aforementioned, it can be concluded that even though serious games surrounding BP do exist, none of them actually deal with the physical symptoms affiliated with the disorder. Consider for example the term Bipolar Eyes[23], a symptom that occurs during hypomanic episodes, where one's pupils are enlarged, leading to intense sensitivity to light sources. Moreover, as mentioned above, hypomania is greatly connected to sleep deprivation, leading to overall heightened senses, as eyes can appear tired and cause blurry vision. Another sense greatly affected by hypomanic

episodes is hearing. Individuals who suffer from hypomania have mentioned sounds becoming louder, and great sensitivity to sharp noises, which can lead to breakdowns. Bearing all the above in mind, individuals affected by hypomania can be in immediate danger while engaging in social and everyday activities. For this exact reason, a serious game that can simultaneously educate and assist people who deal with BP can be deemed of great importance and something that I would like to bear the responsibility of. This serious game will deal strictly with the physical aspects of a hypomanic episode, from the avatar's POV (point of view) creating the needed alterations in their vision, hearing, and overall sensory perception. The main idea consists of an avatar, walking down an angled pavement and by-passing a lighting column, for the vision alterations to be highlighted. Continuing, a car will drive beside the avatar, and when close to them a honk will be heard. This way, the heightened hearing will be represented, and lastly, after the car has driven by, a shadow-like figure of the car will follow, to demonstrate the difference in the avatar's overall perception considering movement. The game will be developed using Unity3D's VR tools and the C# programming language. The options available to the user will be 1. Start, 2. Menu, where modifications can be made, and 3. General Information on the Bipolar Spectrum with references to symptomatology, causes, treatment options, and Greece's nation suicide helpline. Lastly, a theme song that best represents the spectrum is going to be included.

B. Developed Demo

During the course of the last 3 months, the aforementioned approach has been developed using Unity 3D, while all scripts used are in C# code. First of all, for the development, the 3D built-in module was used, and 5 different Scenes in total were created named Main Scene, Menu, Settings, Info, and Quit accordingly. In this segment, more details for each Scene will be reported:

a) Menu Scene: In this particular scene, the user is met with four choices: Start, Settings, Info, and Quit. This Scene was created for functionality issues, for the user to determine whether they want to start the game, change the appropriate Settings, get information on the game, or quit. For the background, an animated video featuring artwork especially created and animated was used. In the artwork, different shades and textures were used. Lastly, the Theme Song used is called 'Asximoi Hxoi', by artists Pan Pan and Nalyssa Green, a song that fits the theme of the game. All buttons were created with UX/UI settings.

b) Quit Scene: The Quit Scene consists of a message displayed to the user, suggesting whether they are sure they want to quit or not. If the 'No' option is selected, a redirection to the Menu Scene occurs, else if the user chooses the 'Yes' button, the game terminates. For the background, an animated video featuring artwork specifically created and animated was used. The Theme Song used is called 'Asximoi Hxoi', by artists Pan Pan and Nalyssa Green, with the vocals removed using AI technology, to not overstimulate the user. All buttons were created with UX/UI settings.

c) Settings Scene: By interacting with the Settings Scene, the user can control the Brightness, Music Volume, and Sensitivity. More specifically, through the three different Sensitivity buttons, the intensity of the symptomatology can be regulated to the user's liking. Similarly to the aforementioned scenes, the background is an animated video artwork especially created and animated. For the background music, the Theme Song used is called 'Asximoi Hxoi', by artists Pan Pan and Nalyssa Green, with the vocals removed using AI technology, to not overstimulate the user. All buttons were created with UX/UI settings.

d) Info Scene: A very important addition to this Serious Game experience was an informative chapter surrounding the Bipolar Spectrum. For this purpose, the Info Scene was created, featuring several details about the theoretical research executed before the development, personal information of those who helped during this project, as well as information on how to reach out for help if struggling. For the background music, the Theme Song used is called 'Asximoi Hxoi', by artists Pan Pan and Nalyssa Green, with the vocals removed using AI technology, to not overstimulate the user. The background is an animated video artwork especially created and animated. All buttons were created with UX/UI settings.

e) Main Scene: Lastly, the Scene of the utmost importance was the Main Scene. The game consists of a neighborhood, which the user can wander in, while experiencing the auditory and visual changes a person during Hypomania endures, with the use of Post Processing Effects. The dominating effects are Camera Blur, a short Depth of Field, Tremble, and Lens Distortion. Other Color-Related adjustments

were also featured, to create an eerie feeling. While the avatar walks through the neighborhood, a car is programmed to run a specific route and when close to them, a honk sound can be heard. Moreover, already existing 3D objects were imported through the Unity Register, due to time sortage, while others were created by me in Blender and added as details. The avatar is controlled through the arrow keys (Up for Forward, Down for Backwards, Right for movement to the right, and Left for movement to the left) and a 360° view is provided through the Mouse Controller. Several scripts are incorporated, in order to create boundaries for the neighborhood, interaction between the avatar and the car, and other functionality issues. In case the user wants to quit, the Esc button redirects them to the Menu Scene. Lastly, the Theme Song used is called 'Asximoi Hxoi', by artists Pan Pan and Nalyssa Green, with the vocals removed using AI technology, to not overstimulate the user. Later in this paper, more information about the development of each scene is going to be reported.

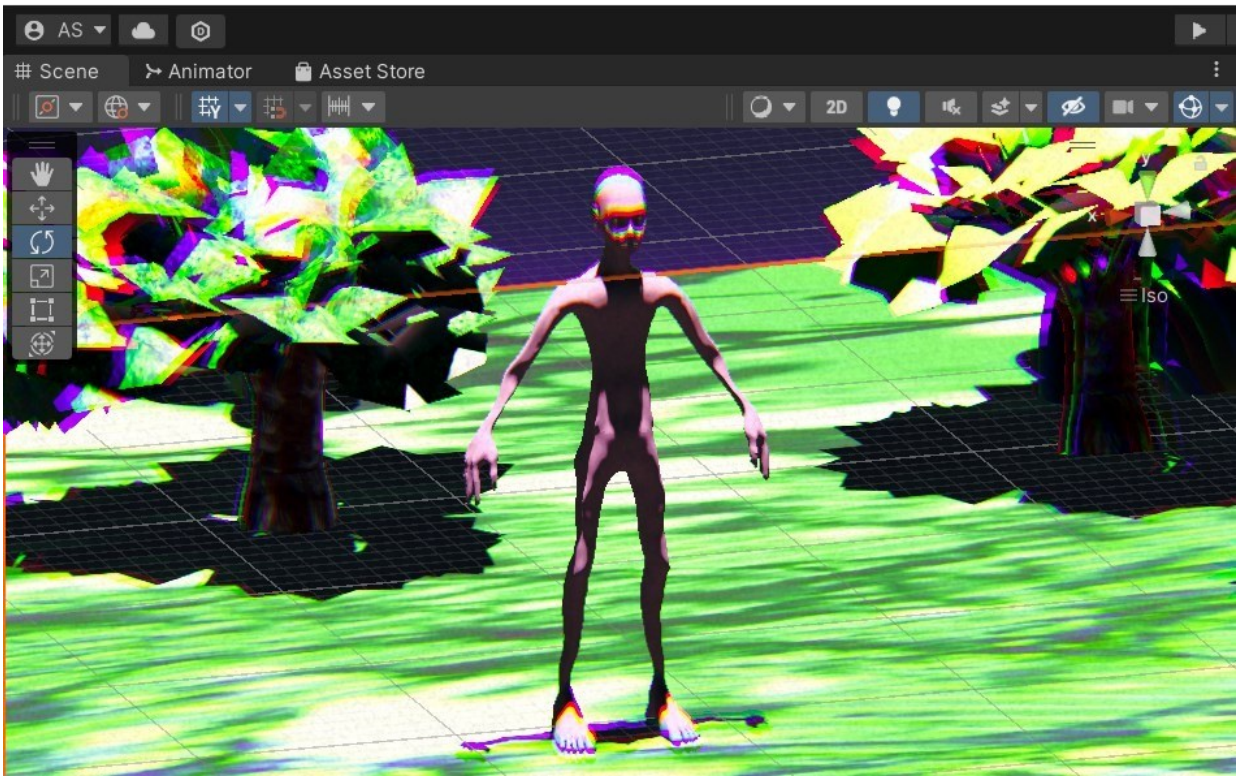

Figure 1:Snapshot of the Games' Avatar

X. POTENTIAL RISKS

Although modern-day technological achievements have acted as a helping hand in a wide range of fields, a rational inquiry that emerges is whether serious games can lead to addiction. Gaming addiction, formally known as Internet Gaming Disorder (IGD), has been recognized as a significant issue due to its potential to cause substantial harm, including poor academic performance, mental health issues, and socialization problems[24]. This exact condition can be attributed to phycological, social and environmental factors, including peer pressure, social interaction within the games, as well as more severe mental health issues such as depression and anxiety. Based on researches conducted, Cognitive Behavioral Therapy (CBT) is one of the most effective approaches, helping individuals recognize and alter problematic thought patterns and behaviors associated with gaming. However, it should be highlighted that Bipolar Odyssey is developed in order to educate and raise awareness about the Bipolar Spectrum, rather than entertain, a fact that can reduce significantly the risk of addiction.

XI. SCENES

A. *MAIN MENU SCENE:*

In this particular segment, more detailed information about the development of the Main Menu scene is going to be analyzed.

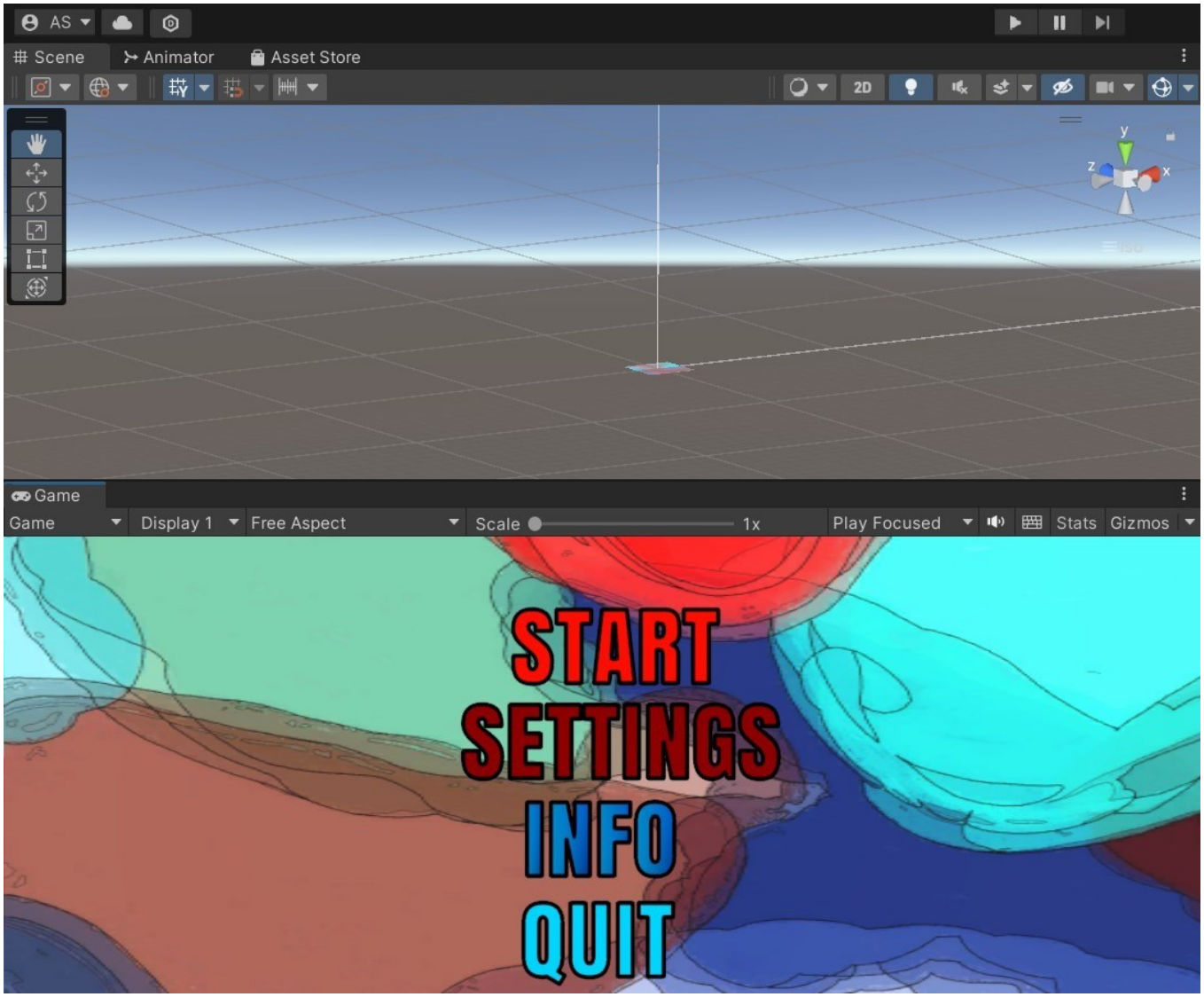

Figure 2:Snapshot of the Main Menu Scene

a) Background: An empty GameObject named background was created. Then, in the Inspector Tab:

Add component-> Video Player-> add the artwork (.mp4) file, with Play on Awake, Wait for First Frame and Loop options enabled, while the Playback Speed is set to 0.25.

All artwork used in the Main Menu Scene, were created using Krita in Huion Kanvas 13. The artwork features watercolor elements with different textures and shading. After the basic background was created, Krita's animation option was used and the result was exported as a .mp4 file for it to be used as a Video Player GameObject.

b) Music: An empty GameObject named melody was created. Then, in the Inspector Tab:

Add component-> Audio Source-> add the .mp4 file, with the options Play on Awake and Loop are enabled, as well as the 3D Sound settings. The same applies for another ,.mp4 file that adds the sound of an increasing heartrate.

c) Buttons: In order to create the needed buttons, an empty GameObject was created, named "buttons". Then*, right-click on the GameObject-> UI-> Button,* and repeat this process for all 4 buttons. Also, 4 GameObjects were created, one for each script in charge of changing the scenes. In order to further explain the functionality, the Start button will be examined:

Firstly, the Interactable option is enabled, the Transition option used is Color Tint and the Target Graphic component is START (image). The button is colored various tints of red and the Fade Duration is valued at 0.25. In the 'On Click()' window:

In the first Drop-Down-> Runtime Only is chosen-> GameStart Object is dragged and dropped-> second Drop-Down-> MainScene-> ChangeScene(string), and in the blank window that shows up below 'MainScene' is used as the input, in order for the scene to change into the games' main scene. Same logic applies to all buttons.

B. SETTINGS SCENE:

a) Background: An empty object named "background" was created. Then, in the Inspector Tab: *Add Component-> Video Player-> Add my artwork with Play on Awake, Wait for First Frame and Loop options checked*. Also, the playback speed is set to 0.25.

All artwork used in the Main Scene, as well as in the Menu, Settings and Info scenes were created using Krita in Huion Kanvas 13. The artwork features watercolor elements with different textures and shading. After the basic background was created, Krita's animation option was used and the result was exported as a .mp4 file for it to be used as a Video Player GameObject.

b) Music: First, an empty GameObject named "melody" was created. During the Settings scene of the game, the vocals of the song were removed using AI Vocal Remover, so that only the songs' melody is heard, in order to avoid overstimulating the user. Then, in the Inspector Tab:

Add component-> Audio Source-> Add the AI edited song, with the vocals removed. Also, the options Play on Awake and Loop are checked, as well as the 3D Sound setting.

c) Buttons: An empty GameObject named Sliders was created. Each slider has two components, one used for the text needed and one for the actual functional slider.

- *Text*: *Inside the parent GameObject-> Right-click-> UI-> Text*. All text above the sliders is created the same way. In the Text Input window, the message that needs to be shown in the scene is written. The Text Style chosen is 'Normal', Font Asset is "Anton SDF", Font Style is AB and Font Size is 120, while the Outline of the text is black and has a value of 0.25 in Thickness. Also, the Color Gradient option is enabled, with the Color Preset being yellow to orange. Lastly, the Wrapping option is enabled, as well as the Overflow one and both the Horizontal and Vertical Mapping are set as "Character". All above settings are applied to all text used in the settings scene.
- *Sliders*: *Inside the parent GameObject-> Right-click-> UI-> Slider.* Firstly, the Interactable option is enabled and the Transition chosen is the 'Color Tint' one. All colors used are shades of white, with tints of grey and the Fade Duration is set to 0.1. The values acceptable for the slider vary from 0 to 1 and the Direction of the slider is Left to Right. For each slider, a script was created in order to control the music and brightness values. Lastly, in the Music Slider for example:

  in the 'OnValueChanged()' window-> press the + symbol-> Drag and Drop the appropriate slider-> Select Runtime only in the first Drop-Down-> Click on the second Drop-Down-> Select Sound Slider and select ChangeVolume(), which is the method used in the Sound Slider script in order to change the volume of the background music. Same logic applies to all sliders.

d) Back to Menu Button: Firstly, an empty GameObject named "Back" was created. *Inside the parent GameObject-> Right-Click-> UI-> Button and then again Right-Click-> UI-> Text.* In the Button component the Interactable option is enabled and the Target Graphic is matched with the Back

image created in the Text component. The colors chosen are white and fade into a light burgundy color, while the Fade Duration is set to 0.1. Then in the 'On Click()' window:

Press the + symbol-> select 'Runtime Only' in the first Drop-Down-> Drag and Drop the Back to Menu script in the window below-> click in the second Drop-Down-> select Menu-> select ChangeScene(string), which is the method used in the Change Scene Script, in order to change scenes. Lastly, in the blank window that appears below, 'Menu' is written in order for the scene to change into the Menu one. Also, for the Text Component, the Text Style used is Normal and the Font Asset chosen is called 'Anton SDF', accompanied by the 'Anton SDF Material' Material Preset. The Font Style is AB and the Font Size is set to 60, colored red, with a 0.25 Thickness, black outline.

Figure 3:Snapshot of the Settings Scene

C. INFORMATION SCENE

a) Background: An Empty GameObject named 'background' was created. Then, in the Inspector Tab:

Add component-> Video Player-> Add the artwork (.mp4) file, with Play on Awake, Wait for First Frame and Loop options enabled, while the Playback Speed is set to 0.25.

All artwork used in the Info scene were created using Krita in Huion Kanvas 13. The artwork features watercolor elements with different textures and shading. After the basic background was created, Krita's animation option was used and the result was exported as a .mp4 file, for it to be used as a Video Player GameObject.

b) Music: An empty GameObject named "melody" was created. During the Info scene of the game, the vocals of the song were removed using AI Vocal Remover, so that only the songs' melody is heard, in order to avoid overstimulating the user. Then, in the Inspector Tab:

Add component-> Audio Source-> Add the AI edited .mp3 file, with the vocals removed, with Play on Awake and Loop options enabled, as well as the 3D Sound setting.

c) Info Text: In order to create the explanatory text, the following steps were followed. In the Hierarchy Tab:

Right-Click-> Create-> Empty GameObject named "Info". Then, Right-Click on the empty GameObject-> UI-> Text.

In the Text Input window a brief illustrative message is featured, in order to inform the user about the cause of the experience they are about to interact with, as well as all personnel that supervised and helped with the project, and lastly all hardware and software used, as well as the theme song featured. Additionally, a disclaimer is mentioned, for the user to refrain from using the experience if they are sensitive to such a subject, along with the national helpline used for suicide prevention. The Text Style used is 'Anton SDF', accompanied by the 'Anton SDF' Material Preset and the AB Font Style. The Font Size used is set to 40, colored light blue, with a black, 0.25 thick outline.

d) Scroller: Since the text featured is quite spacious, a scroller was included in order for the user to be able to comfortably read it. For the scroller creation:

Right-Click on the "Info" parent GameObject-> UI-> Scroller. The scroller is a deep blue color and the settings in the Scroll Rect Window are as following:

- Content: Text,
- Vertical: enabled,
- Movement: Elastic,
- Movement Type: set to 5,
- Inertia: Enabled with a Deceleration Rate valued at 0.135 and Scroll Sensitivity at 10.
- Viewport is matched to the scroller object
- Vertical Scrollbar is matched to the Scrollbar Vertical object.
- Spacing: set to -3.

e) Back to Menu Button: Firstly, an empty GameObject named "back" was created. *Inside the parent GameObject-> Right-Click-> UI-> Button and then again Right-Click-> UI-> Text.* In the Button component the Interactable option is enabled and the Target Graphic is matched to the "Back" image created in the Text component. The colors chosen are white and fade into a light burgundy color, while the Fade Duration is set to 0.1. *Then, in the 'On Click()' window-> press the + symbol-> select 'Runtime Only' in the first Drop-Down-> Drag and Drop the "Back to Menu" script in the window below-> click in the second Drop-Down-> select Menu-> select ChangeScene(string)*, which is the method used in the "Change Scene" Script, in order to change scenes. Lastly, in the blank window that appears below, 'Menu' is written in order for the scene to change into the Menu one. Also, for the Text Component, the Text Style used is Normal and the Font Asset chosen is called 'Anton SDF', accompanied by the 'Anton SDF Material' Material Preset. The Font Style is AB and the font size is set to 60, colored Prussian blue, with a 0.25 thickness, black outline.

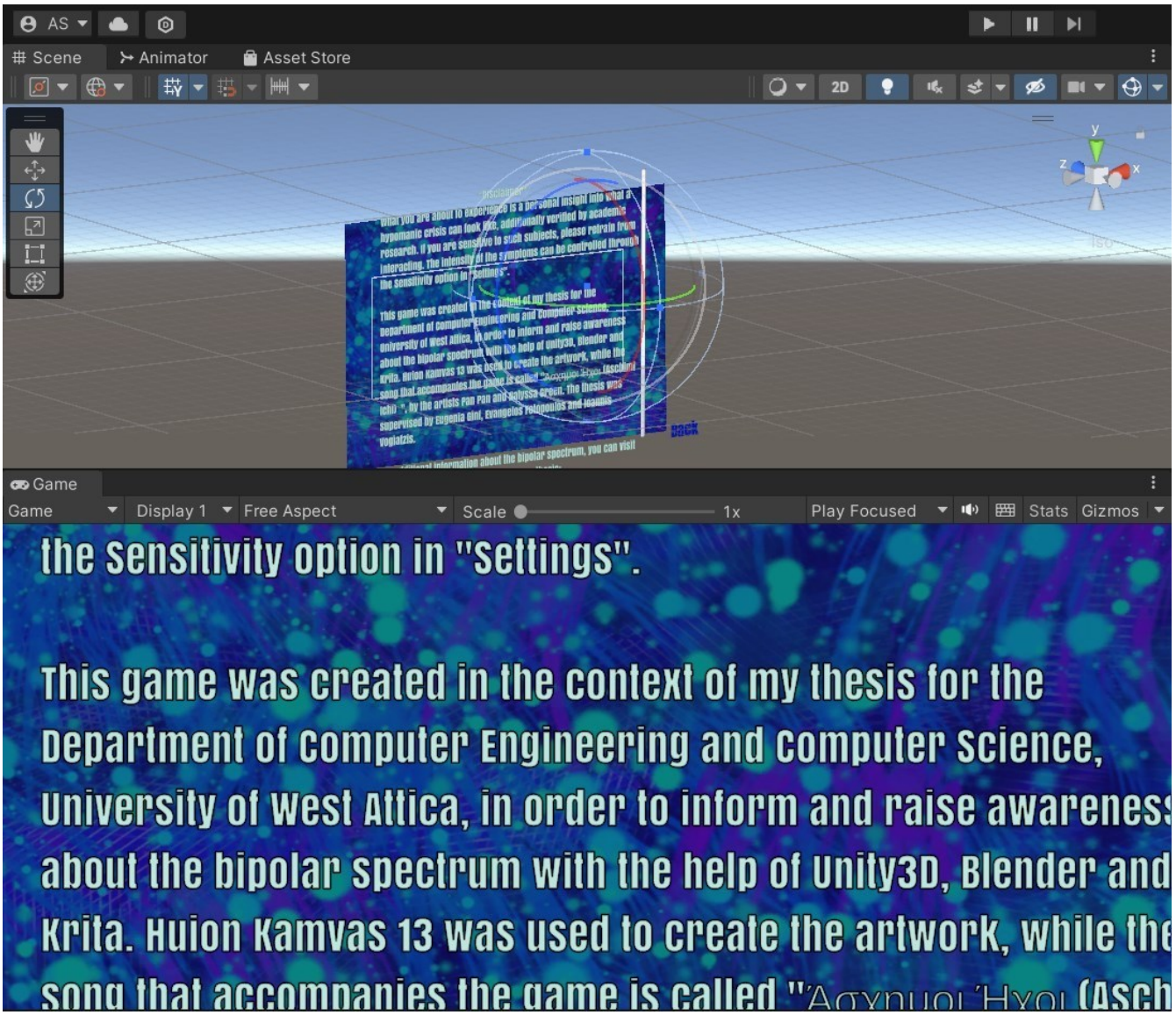

Figure 4:Snapshot of the Information Scene

D. QUIT SCENE

a) Background: An empty object named “background” was created. Then in the Inspector Tab:
Add component-> Video Player-> Add the artwork (.mp4) file, with Play on Awake, Wait for First Frame and Loop options enabled, while the Playback Speed is set to 0.25.
All artwork used in the Quit scene were created using Krita in Huion Kanvas 13. The artwork features watercolor elements with different textures and shading. After the basic background was created, Krita's animation option was used and the result was exported as a .mp4 file for it to be used as a Video Player GameObject.

b) Music: An empty GameObject named “melody” was created. During the Quit scene of the game, the vocals of the song were removed using AI Vocal Remover, so that only the songs’ melody is heard, in order to avoid overstimulating the user. Then, in the Inspector Tab:
Add component-> Audio Source-> Add the AI edited .mp3 file, with the vocals removed, with the Play on Awake and Loop options enabled, as well as the 3D Sound setting.

c) Displayed Message: In the Hierarchy Tab:
Right Click-> Create empty GameObject named 'buttons'. In the Text Input Window the message 'Are you sure you want to quit' is written, with the 'Normal' Text Style, in the 'Anton SDF' Font Asset and

with the 'Anton SDF-Drop Shadow' Material Preset. The Font Style chosen is AB, and the font is colored pink with the Font Size set to 50 and a 0.25 sized black Outline.

d) Buttons: For the functionality of the Quit Scene, two buttons were needed, one for the affirmative and one for the negative option accordingly. For this purpose, the following actions took place:

Right-Click on the "buttons" GameObject-> UI-> Button. The Interactable option was enabled, the Transition option was set to Color Tint and the buttons were colored various tints of burnt sienna. The Fade Duration of the Button was set to 0.1. Additionally, two scripts were created, one in order to exit the game and one in order to lead the user back to the "Main Menu" scene. For example, for the Yes button, *in the 'On Click()' window-> in the first Drop-Down, Runtime Only is chosen-> the "Quit Manager" script is Dragged and Dropped-> in the second Drop-Down-> Exit-> QuitGame()*, which is the method used in the "Exit" script in order to quit the game application. In the same way, for the No button, *in the 'On Click()' window-> in the first Drop-Down Runtime Only is chosen-> the Back To Menu script is dragged and dropped->in the second Drop-Down-> Menu-> ChangeScene(string)*. In the blank window that occurs below, 'Menu' is written, in order to change the scene back to the Menu one when the No button is pressed.

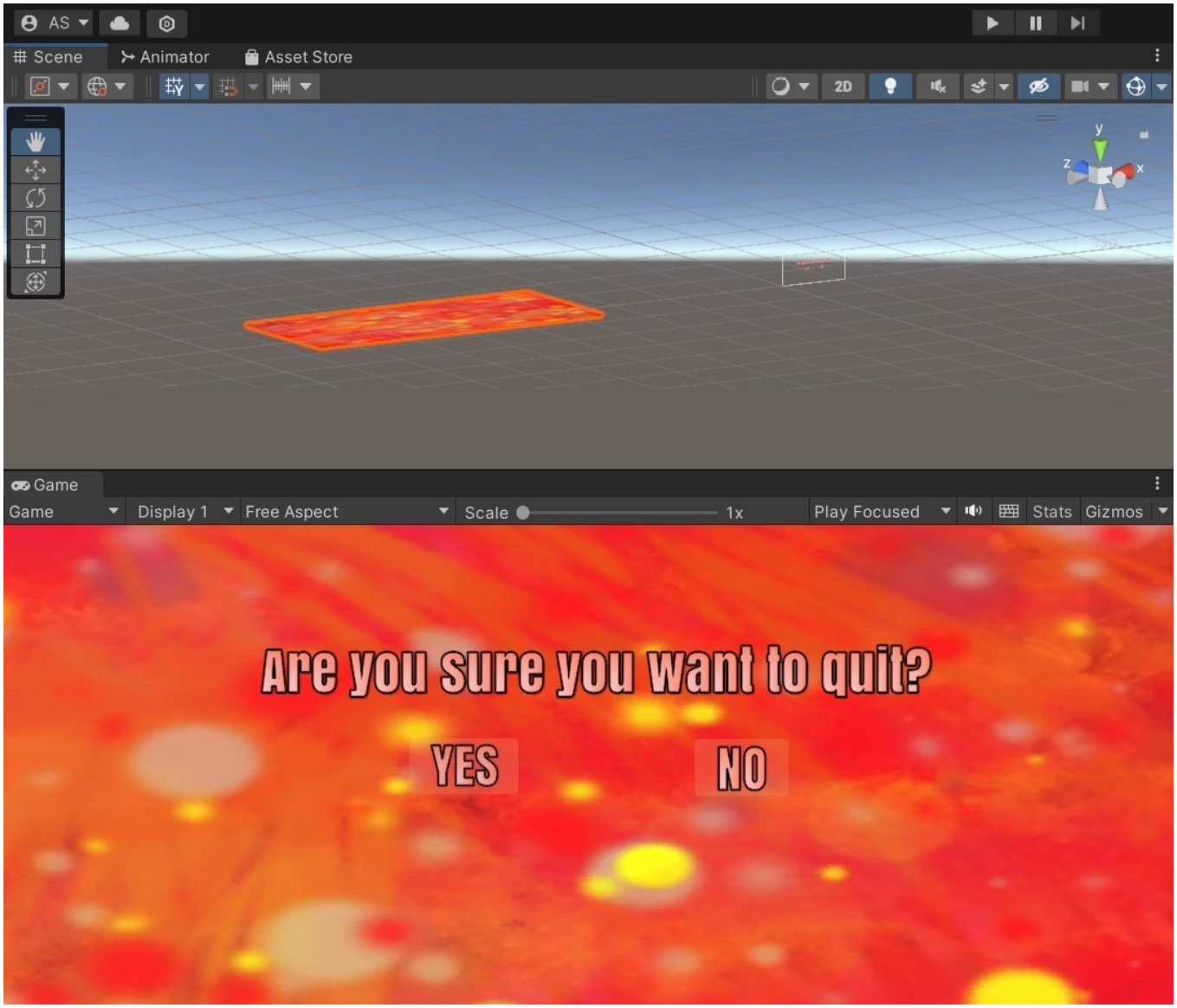

Figure 5:Snapshot of the Quit Scene

E. MAIN SCENE

a) Directional Light: For the illumination of the Main Scene, a Directional Light is used. More specifically, the light is positioned and rotated in the X, Y and Z axes as such:

	X	Y	Z
Position	76	486	38
Rotation	90.2	-30	0
Scale	1	1	-2

Table 1: Position, Rotation and Scale of Directional Light

In the Inspector window:
Right-click-> Light, set the Type as Directional, in order to create a distant light source that simulates the sun, and the Mode is set as Realtime, for any changes happening in the position, rotation, intensity of color of the light to be immediately reflected in the rendered scene. In the Emission option, the Light Appearance is set to Filter and Temperature, with the intention of choosing a white hue in the Filter option and a warmer overall color of the illumination. The Temperature filter is valued at 6728 Kelvin, the Intensity at 10 and the Indirect Multiplier at 10. The Render Mode is set as Important, while the Culling Mask, a factor that optimizes the rendering performance and the visibility of objects, is set as Everything. For the Shadows, the Shadow Type used is Soft Shadows, with the Strength parameter valued at 1, the Near Plane, a value that affects the precision and range of shadows is valued at 0.2 and the Bias option, which offsets the comparison between the depth values of the shadow map and the scene, uses settings from Render Pipeline.

It should be noted though, that since the brightness of each scene can be controlled through the Settings Scene, an empty GameObject named "*brightness adjustment"* was created for the correct functionality. More specifically, this particular GameObject was created in order to add the "Apply Brightness" script, which will be furtherly examined later on.

b) Creation of roads: In the hierarchy tab*:*
Right-Click-> New Object and create a parent GameObject called "roads" in order to create children GameObjects for each road. This way all roads will be in the same position in the Y axis. The neighborhood will consist of 5 main subareas that are created with 3D objects and Planes. These areas are:

1. forest1
2. forest2
3. school
4. backyard
5. block1 and
6. block2

The following planes were created in order to create the roadwork:

1. traffic_island (road island that is set to divide the two main roads of the neighborhood),
2. block2_main_road (the initial avatar position, positioned to the right of the 2nd block and the backyard area)
3. school_road_front, (positioned in the front of the school area)
4. school_road_right, (positioned to the right of the school area)
5. school_road_behind, (positioned in the back side of the school area)
6. school_road_left, (positioned to the left of the school area)
7. school_pavement_left, (positioned next to the school road left GameObject)
8. school_pavement_behind, (positioned next to the school road behind GameObject)
9. school_pavement_right and (positioned next to the school road right GameObject)

10. block1_road (positioned to the left of the block1 area and used to create the surface for the block1 area)
11. block1_pavement (positioned in-between the school pavement right and block1 GameObject)
12. forest1_floor (used in order to create the surface for the forest1 area)
13. forest2_floor (used in order to create the surface for the forest2 area)
14. block2_floor (used in order to create the surface for the block2 area)
15. bloc1_floor_back (used in order to create the surface for the back of the block1 area)
16. school_floor (used in order to create the surface for the school area)

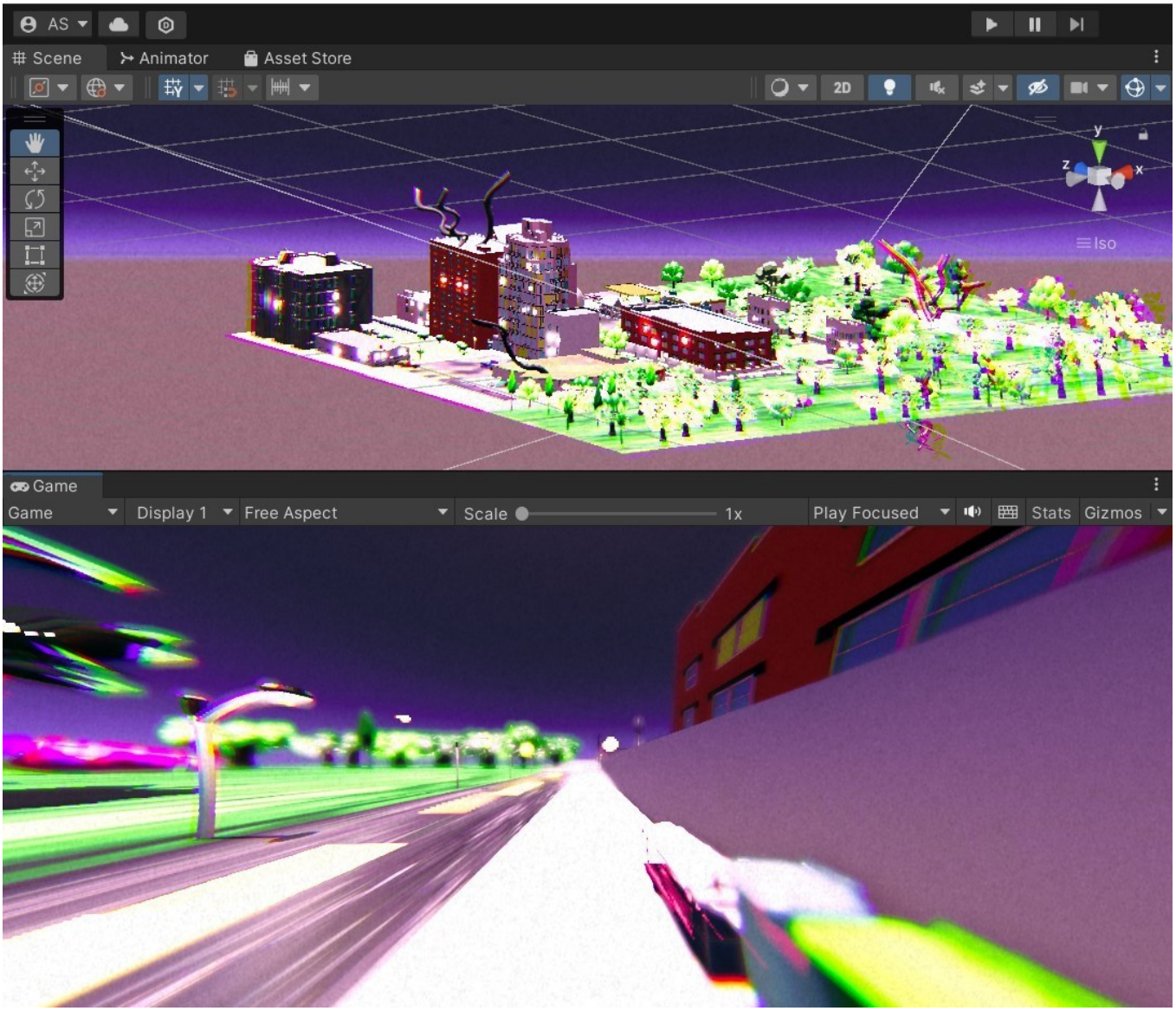

Figure 6:Snapshot of the Main Scene

All planes are placed at Y=0. For the GameObjects “block2_main_road”, “school_road_front”, “school_road_right”, “school_road_behind”, “school_road_left” and “block1_road” lane markings were added by creating children GameObjects for each one, by using “Plane” 3D Objects. This way, the roads appear more realistic. In order to personalize the neighborhood, all artwork created was then used as the material for the actual roads, the pavements, the lane markings and the surfaces for forest1, forest2, block1 and block2. Before further examining each and every subarea of the neighborhood, all buildings and details used (such as trees, cars, flowers etc.) were imported directly through the Unity

Asset Store. Specifically, the packages used are *CITY package by 255 PIXEL STUDIOS, Fantasy landscape by PXLTIGER, Low Poly Environment by PALMOV INSLAND, Grass Flowers Pack Free by ALP, Rusty Cars by DUANE'S MIND, fantasy spider by DANTE'S ANVIL* and *Horror Assets by LUKAS BOBOR*. For the import of the aforementioned packages, the Unity Asset Store was visited and said packages were purchased. By opening Package Manager in Unity, the appropriate assets were selected, all the appropriate data were downloaded and imported to the Project Window. Some 3D Objects were created by me using Blender, but more will be mentioned later.

c) Music: The music that accompanies the Main Scene is called "Asximoi Ixoi" by artists Pan-Pan and Nalyssa Green. In order to include it, it was imported as a .MP3 file, for it to be compatible with Unity3D in the Assets folder, and then an empty GameObject was created called Background Music. Selecting the GameObject, *click Add Component and select Audio Clip and then the add .mp3 file in the assigned field.* The Play on Awake and Loop options were checked, as well as the 3D Sound option. During the Main Scene of the game, the vocals of the song were removed using AI Vocal Remover, so that only the melody is heard, in order to avoid overstimulating the user.

d) Avatar creation: The avatar used in the game was created by me, via sculpting in Blender software. More specifically, the software was accessed: *Blender -> New file -> Sculpt and proceed to sculpt the character as desired. Then: Save the character -> Export as .obj object* in order to be able to import it and add it to the MainScene in Unity3D. The avatar seems to be malnourished, in order to create an overall tired and eerie aesthetic. The avatar does not add functionality to the game, it only serves the purpose of humanizing the patient.

e) Light sources creation: In the Hierarchy tab then: *Create new -> GameObject -> Light -> Spot light.* Each light is positioned in the X and Y axis accordingly, and placed in a buildings' window of my choice, in order to give the effect of the light shining inside the building and then the Intensity value was increased, in order for the changes in vision noted during hypomania to be highlighted, e.g. brighter lights. The Range the light reaches is also increased. Point lights were selected, because they emit light in a cone shape with a specified angle and direction, and are useful for creating focused beams of light, such as flashlights, spotlights, or car headlights and can only cast shadows with adjustable softness and quality. In all Spot Lights the Intensity Value is set to 20, the Range value to 10, and the Indirect Multiplier, which controls the intensity of the indirect light sources, value to 5. Moreover, the Rendering Mode is set as "Important", in order for the light sources not to glitch, and the Culling Mask option, responsible for which layers will be affected, is set to "Everything". Lastly, the Shadow Type option is set to Soft Shadows, for the artificial softening of the shadows cast.

f) Materials: In order to create the Materials used, in the Project Tab: *Right-Click -> Create -> Folder,* and then name it "Materials", in order to be easily accessed. *Inside the Materials Folder -> Right-click-> Create -> Materials, and* name it appropriately. Lastly, in the Base Map option, under the Surface Category the .png image that contained the artwork was dragged and dropped. The static artwork used as materials in the main scene were also created with Krita in Huion Kanvas 13 and also feature different textures and shading. In order to be used as a material, the artwork was exported as a .png file, in order to be compatible with Unity3D and then in the Project Window-> *Create-> Material-> Shader Component-> Drag and Drop desired file.*

g) Moving Car: For the moving car GameObject two components need to be examined: the Waypoints used and the actual Car GameObject.

- Waypoints: An empty GameObject named "Points" is created, in order to set the waypoints of the route that the car needs to follow. Under the "Points" parent GameObject, 4 children GameObjects were created, called point1, point2, point3 and point4. For each GameObject, their position in the X, Y and Z axis are:

POINTS	X AXIS	Y AXIS	Z AXIS
Point1	-44.6	69	28.2
Point2	-45.5	69	-85.5
Point3	54.1	69	-85.5

Point4	52.8	69	29.1

Table 2:Positions of Waypoints in the X, Y and Z axis

For each GameObject, their rotation in the X, Y and Z axis are:

POINTS	X AXIS	Y AXIS	Z AXIS
Point1	0	-180	0
Point2	0	0	0
Point3	0	90	0
Point4	0	0	0

Table 3:Rotation of Waypoints in the X, Y and Z axis

Then, in order to create the route, a script was written and added. In the Inspector window: *Add Component-> Script and add the "CarController.cs" script.* Also, in the Inspector window*: Add Component-> Script and add the "HonkControl.cs" script.* Then, in the blank window below the .mp3 file (honk) is dragged and dropped.

After the script was added, in the Waypoint window, the four GameObjects that represent each point of the route are added. In the Inspector window:
CarController (script)-> Waypoints-> click on the '+' window-> Add four elements-> Drag and Drop the GameObjects, with the Speed valued at 4, the Move Speed at 4 and the Rotation Speed at 5.

- Car: Firstly, an empty GameObject named "Car" was created. Then, a Car 3D Object imported from the Unity Registry (Rusty Cars by DUANE'S MIND) was used, with personalized artwork created used as the Material. In the Inspector Tab:
  Add Component-> Audio Source, in order to add a honk sound that sounds when the Car passes by the player. For this purpose, *a .mp4 file is imported in the Project Tab and then Dragged and Dropped in the AudioClip window*. The Bypass Effects option is enabled, as well as the 3D Sound setting. In the Inspector Window: *Add Component->Rigidbody,* in order to add Mass to the Car GameObject. The Mass is valued at 1, Drag at 0, Angular Drag at 0.05 and the Use Gravity and Is Kinematic options are enabled. Lastly, the Interpolate option is set as None and the Collision Detection is Discrete. In the Inspector Tab: *Add Component-> Box Collider*, in order to create a trigger effect to sound the honk when the car is close to the player. The collider was edited accordingly to cover the entirety of the Car GameObject and the Is Trigger option is enabled. In the Inspector Tab: *in the Tag window-> click the Drop-Down-> Add Tag->in the Tags window-> click the '+' symbol-> New Name Tag (name it Car)-> Save and click the Car GameObject again->Tag->select Car*. In order to create the trigger, a collider needs to also be added to the player, as well as a Rigidbody, with Mass valued at 1, Drag at 0, the Is Kinematic enabled, Interpolate set as Interpolate and Collision Detection as Discrete. Choosing the avatar GameObject, in the Inspector window: *Add Component-> Capsule Collider*. The collider is edited accordingly to fit the avatar and the Is Trigger option is enabled. Also, in both the Avatar and the Car GameObjects, in the Inspector window: *Add Component-> Audio Listener*, in order for the sound settings to be functional. With both GameObjects having Rigidbody and Collider components, they can interact with each other and act as triggers. The last step is adding scripts to both GameObjects in order to sound the honk when either one comes in contact with the others' collider. More specifically, in the Inspector window: *Add Component-> Script and add*

the "AvatarController.cs" script. Then, in the blank window below the .mp3 file (honk) is dragged and dropped.

h) *Game Termination:* Firstly, an empty GameObject named Back To Menu is created. Then, in the Inspector Tab: *Add Component-> Script, and add the Menu script, as well as the ESC_exit script*, that both are furtherly examined in the forthcoming chapter.

i) *Neighborhood Boundaries:* In order to create the neighborhood boundaries, an empty GameObject named "limitations" was created, with four children GameObjects named limit, limit2, limit3 and limit4 accordingly. In order to further examine the boundaries, the limit GameObject will be explained as an example, with the same logic applying to all four. In the Inspector window: Add Component-> Rigidbody. With the Rigidbody component accurate collision detection is ensured. More specifically, the Mass is valued at 5, Drag at 0 and Angular Drag at 0.05, while the Is Kinematic and Interpolate options are disabled, and each GameObejcts' Rigidbody is positioned as such:

limit	X	Y	Z
Position	-183.2	-41.6	6.7
Rotation	0	0	0
Scale	10	100	400

Table 4:Position, Rotation and Scale of limit

limit2	X	Y	Z
Position	174.75	-29.5	1.78
Rotation	0	0	0
Scale	10	100	400

Table 5:Position, Rotation and Scale of limit2

limit3	X	Y	Z
Position	-14.24	-29.68	-154.4
Rotation	0	0	0
Scale	10	100	400

Table 6:Position, Rotation and Scale of limit3

limit4	X	Y	Z
Position	9.75	-29.68	172.78
Rotation	0	-90	0
Scale	10	100	400

Table 7:Position, Rotation and Scale of limit4

Moreover, in the Inspector window: *Add Component-> Box Collider*, with the collider properly edited in order to cover all four sides of the neighborhood. The Is Trigger option is enabled and the Material assigned is the "bouncemat", in order to create a bouncing effect when interacted with. Lastly, in the Inspector window: *Add Component->Script, and add the "Boundaries.cs" script*, which will be examined in the forthcoming chapter.

XII. LOGO CREATION

In order to provide a more personalized experience, a logo for "Bipolar Odyssey" was designed and used for the final BipolarOdyssey.exe file. More specifically, the logo was created by me, inspired by the duality of the Bipolar Spectrum, using both Krita and Canva. With the use of both warm and cool color pallets, representing feelings of Hypomania and Depression accordingly, I managed to create a design that signifies how distinct the two episodes can be, but also how when intertwined they can form a whole. Some graphic elements were also added, while the font used is "UnifrakturMaguntia" (provided by Canva), in a 22 size. In order to use the logo in the BipolarOdyssey.exe file the following was done: First step was resizing the .png file to a 256x256 size of pixels and setting the background as transparent. Afterwards, the resized picture was imported in Assets folder of the Hierarchy window. Then, by clicking Edit on the top of the Unity page, I navigated to "Project Settings". In the Projects Settings window, the Player option was chosen and under Player Settings, I navigated to the Icon section. There, the custom icon was dragged and dropped, and finally the game was built. Below, the logo can be seen:

Figure 7:Bipolar Odyssey logo

XIII. VISUAL IMPAIRMENT EFFECTS

In order to represent the impaired visual phenomena when in a hypomanic state, 3 different Volume Profiles were created, in order for the user to be able to control the severity of the symptomatology. More specifically, the 3 Volume Profiles were created named Low, Medium and High accordingly. Below, a more detailed description of the effects used are presented:

A. High Volume Profile

Below, all effects used in order to create the visual changes that occur when the symptomatolgy is set to "High" are mentioned.

BLOOM

A Post processing effect to simulate the way the light interacts with the camera lens. The values are set as such:

- Threshold (valued at 3): controls the brightness level of the pixel, so that only very bright ones will bloom
- Intensity (valued at 10): controls the intensity of the bloom effect
- Scatter (valued at 0.45): controls the spread of the blooming effect, where the higher the scatter level, the more visible is the effect around a bright area
- Tint: used black, adding a blackish hue to the blooming effect
- Clamp (valued at 65472): limits the maximum intensity of the bloom effect
- High quality filtering (on): enhances the quality of the bloom effect for smoother and more realistic blooms
- Skip Iterations (valued at 8): determines the number of iterations used to calculate the bloom effect
- Dirt Texture: adds texture overlay
- Dirt Intensity (valued at 4.3): controls the intensity of the dirt texture

VOLUME

A Post processing stack that controls the overall volume of the audio in the main scene. The values are set as such:

- Mode: global: the volume will affect all audio sources uniformly
- Weight (valued at 1): adjusts the strength of the volume effect
- Priority (valued at 0): determines the priority of the volume effect compared to other

CHANNEL MIXER

A Post processing effect that manipulates the color channels independently. The values are set as such:

- Chromatic Aberration: simulates a color distortion in lenses
- Intensity (valued at 0.6): controls the intensity of the effect

COLOR ADJUSTMENTS

A Post processing effect that applies various color-related parameters. The values are set as such:

- Post Exposure (valued at 0.5): adjusts the overall exposure of the image after rendering
- Contrast (valued at -9): makes the picture appear more washed out
- Color Filter (chose white): adds a white hue to the image
- Hue Shift (valued at 0): shifts the hue of all the colors
- Saturation (valued at 65): controls the purity of color in the image

DEPTH OF FIELD

A Post processing effect that simulates the way a camera lens focuses on objects at different distances. The values are set as such:

- Mode (chose bokeh): adds a more realistic effect
- Focus Distance (valued at 5): specifies the distance from the camera to the point of focus

- Focal Length (valued at 75): controls the magnification of the scene
- Aperture (valued at 4.5): refers to the size of the opening of the lens (here the value adds a more blurred background and a narrower field of view)
- Blade Count (valued at 5): produces smoother and less distinct shapes
- Blade Curvature (valued at 0.35): controls the roundness of the shapes
- Blade Rotation (valued at -3): changes the orientation of the bokeh highlights

FILM GRAIN

A Post processing effect that adds a grainy texture. The values are set as such:

- Type (chose Thin 2): adds a finer grain pattern
- Intensity (valued at 0.6): controls the intensity of the effect
- Response (valued at 0.4): produces an even distribution of grain

LENS DISTORTION

A Post processing effect that simulates the distortion that occurs when light passes through a lens. The values are set as such:

- Intensity (valued at -0.45): controls the strength of the effect
- X Multiplier (valued at 1): distortion is applied evenly across the X axis
- Y Multiplier (valued at 1): distortion is applied evenly across the Y axis
- Center (valued at X:0.5, Y:0.5): distortion is centered in the middle of the screen
- Scale (valued at 1): distortion is applied at its original size

MOTION BLUR:

A Post processing effect that simulates the blurring that occurs when objects move rapidly

- Quality (set as high): the effect is computed with high accuracy
- Intensity (valued at 0.95): controls the intensity of the effect
- Clamp (valued at 0.2): sets the maximum amount of blur (here it is set to 20%)

SPLIT TONING

A Post processing effect that allows to independently adjust the color tones of shadows. The values are set as such:

- Shadows: shadows are colored in burnt sienna
- Highlights: applies white tone to the highlights
- Balance (valued at 0.2): controls the balance between the shadow and the highlight tones

TONE MAPPING:

A Post processing effect that maps the HDR and SDR colors

- Mode (set to aces): a standardized color space and color management system

VIGNETTE

A Post processing effect that darkens or lightens the edges of the screen. The values are set as such:

- Color (set to white): the effect will have no tint
- Center (set at X:0.5 Y:0.5): the center of the effect is aligned with the center of the screen
- Intensity (valued at 0.8): controls the intensity of the effect

- Smoothness (valued at 0.35): provides a smooth transition
- Rounded (checked): provides softer edges for a smooth appearance

WHITE BALANCE

A Post processing effect that simulates the adjustment of color temperature. The values are set as such:

- Temperature (valued at -9): adds a cooler tone to the scene
- Tint (valued at 3): adds a slight magenta tint

The matrix included below can make the effect settings easier to understand.

EFFECT	SUB-REGULATIONS	VALUES
BLOOM	THRESHOLD	3
	INTENSITY	10
	SCATTER	0.45
	TINT	BLACK
	CLAMP	65472
	SKIP ITERATIONS	8
	DIRT INTENSITY	4.3
VOLUME	MODE	GLOBAL
	WEIGHT	1
	PRIORITY	0
CHANNEL MIXER	INTENSITY	0.6
COLOR ADJUSTMENTS	POST EXPOSURE	0.5
	CONTRAST	-9
	COLOR FILTER	WHITE
	HUE SHIFT	0
	SATURATION	65
DEPTH OF FIELD	MODE	BOKEH
	FOCUS DISTANCE	5
	FOCUS LENGTH	75
	APERTURE	4.5
	BLADE COUNT	5
	BLADE CURVATURE	0.35
	BLADE ROTATION	-3
FILM GRAIN	TYPE	THIN 2

	INTENSITY	0.6
	RESPONSE	0.4
LENS DISTORTION	INTENSITY	-0.45
	X MULTIPLIER	1
	Y MULTIPLIER	1
	CENTER	X:0.5, Y:0.5
	SCALE	1
	QUALITY	HIGH
	INTENSITY	0.95
	CLAMP	0.2
SPLIT TONING	SHADOWS	BURNT SIENNA COLOR
	HIGHLIGHTS	WHITE
	BALANCE	0.2
TONEMAPPING	MODE	ACES
VIGNETTE	COLOR	WHITE
	CENTER	X:0.5, Y:0.5
	INTENSITY	0.8
	SMOOTHNESS	0.35
	ROUNDED	CHECKED
WHITE BALANCE	TEMPERATURE	-9
	TINT	3

Table 8:Post-Processing Effects for High Version

Here, some snippets of the game set in High can be seen.

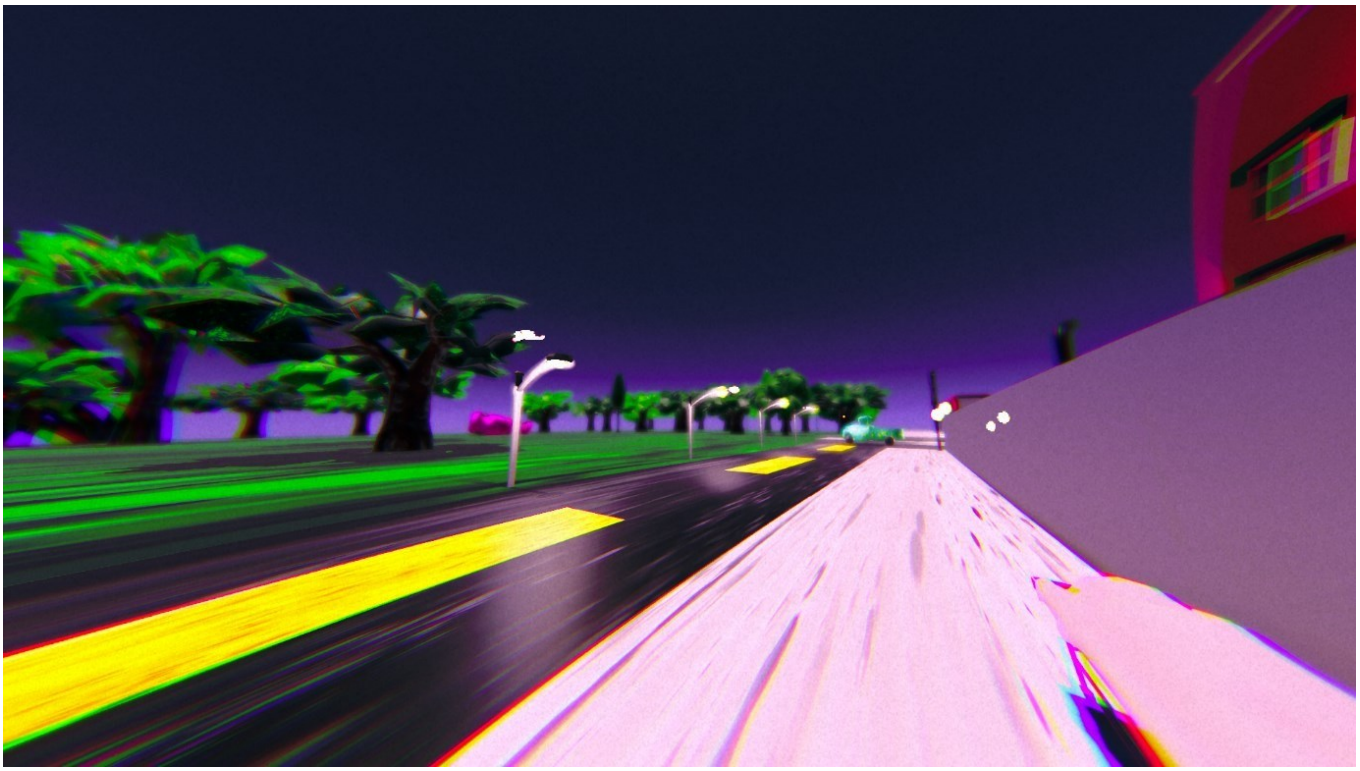

Figure 8:High Version of Post-Processing Effects

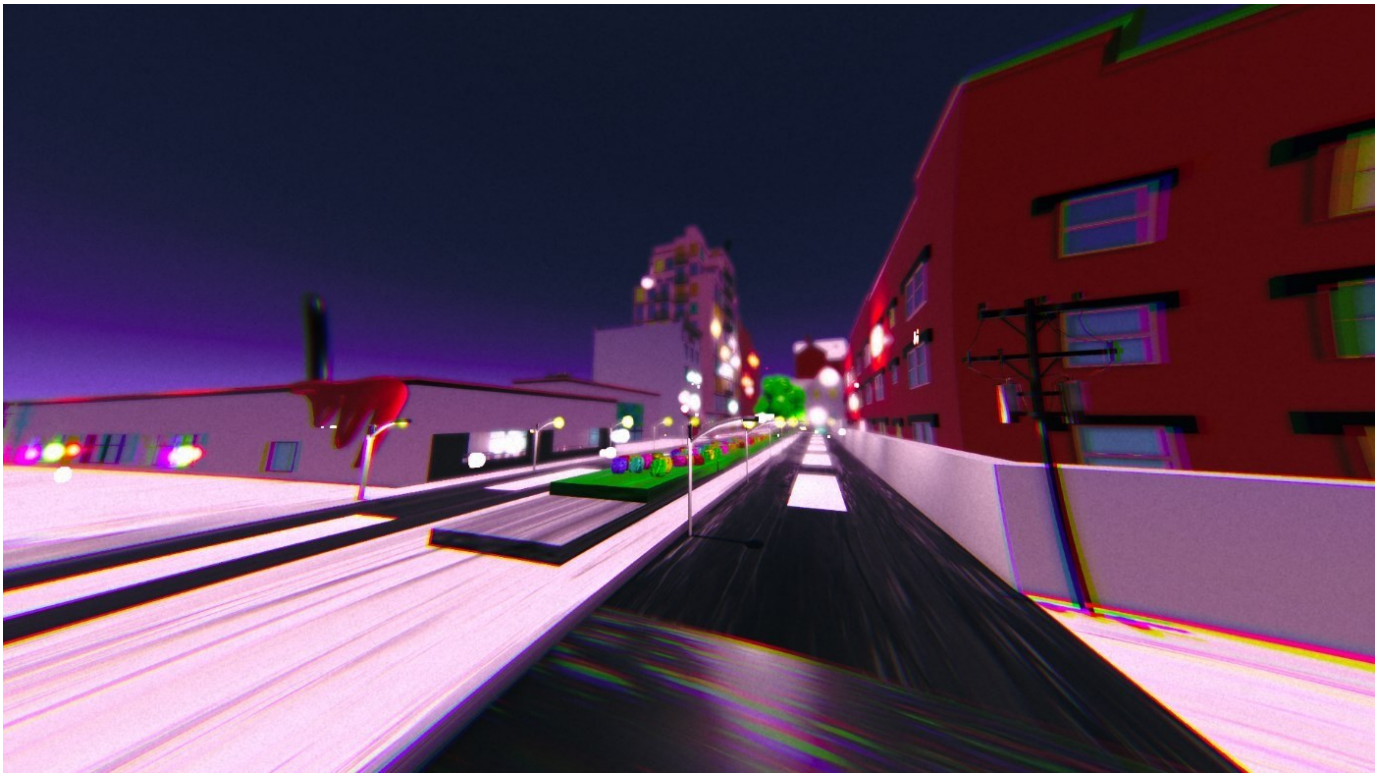

Figure 9:High Version of Post-Processing Effects

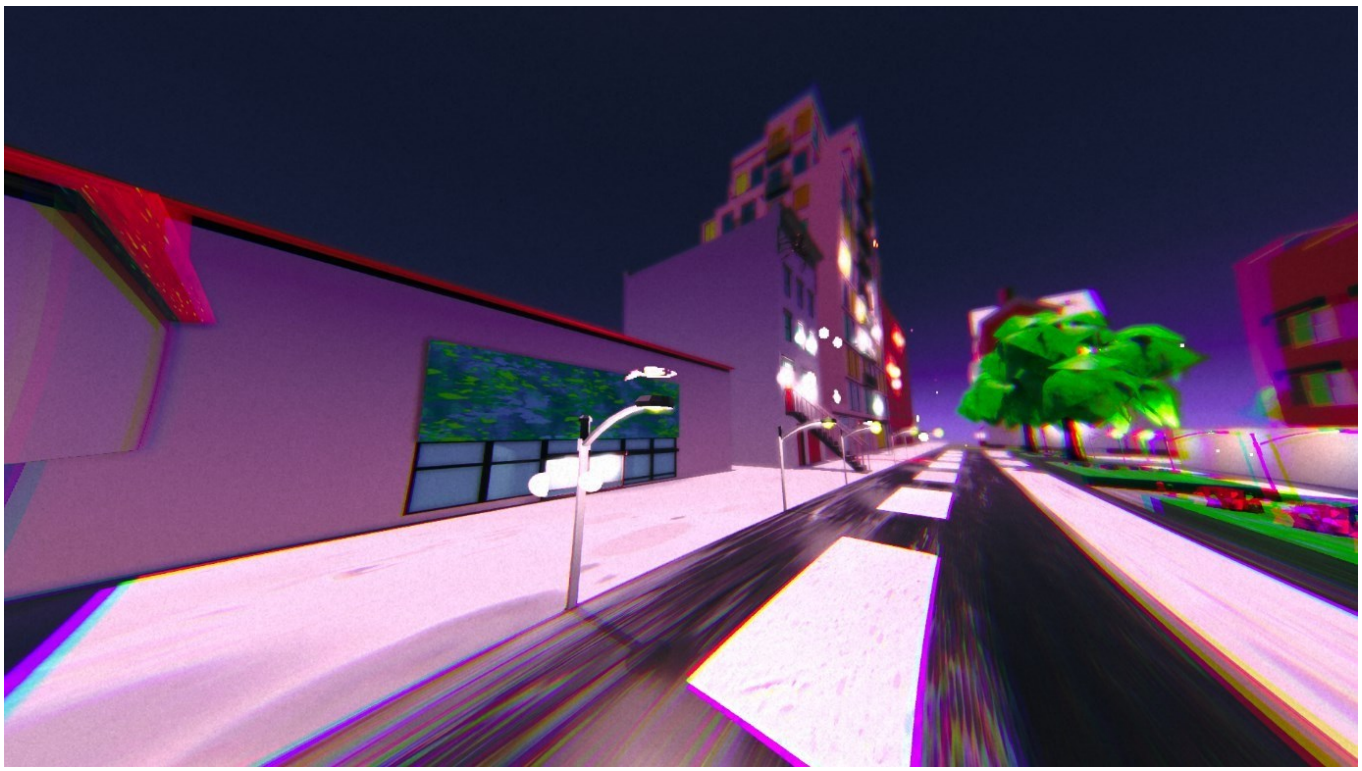

Figure 10:High Version of Post-Processing Effects

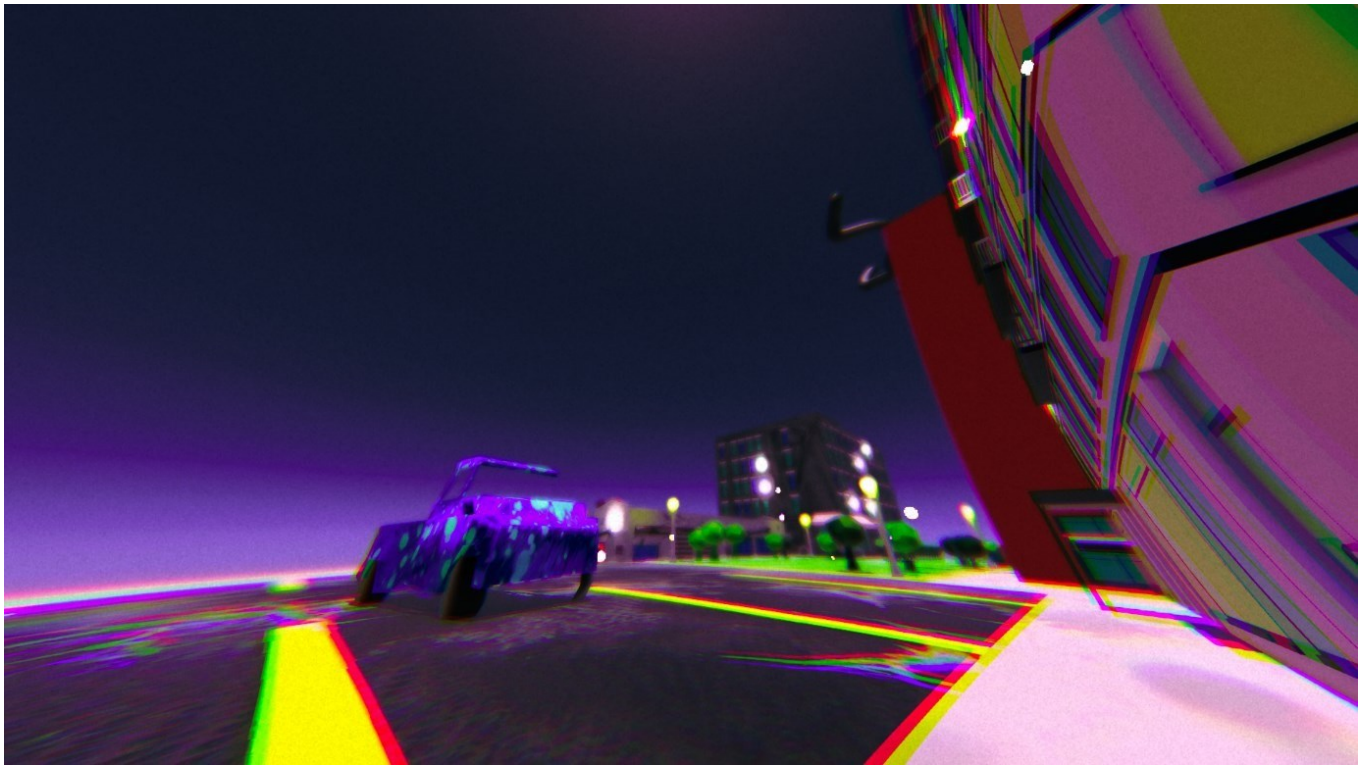

Figure 11:High Version of Post-Processing Effects

B. Medium Volume Profile

Below, all effects used in order to create the visual changes that occur when the symptomatolgy is set to "Medium" are examined.

BLOOM

A Post processing effect to simulate the way the light interacts with the camera lens. The values are set as such:

- Threshold (valued at 2): controls the brightness level of the pixel, so that only very bright ones will bloom
- Intensity (valued at 5): controls the intensity of the bloom effect
- Scatter (valued at 0.25): controls the spread of the blooming effect, where the higher the scatter level the more visible is the effect around a bright area
- Tint: used black, adding a blackish hue to the blooming effect
- Clamp (valued at 65468): limits the maximum intensity of the bloom effect
- High quality filtering (on): enhances the quality of the bloom effect for smoother and more realistic blooms
- Skip Iterations (valued at 12): determines the number of iterations used to calculate the bloom effect
- Dirt Texture: adds texture overlay
- Dirt Intensity (valued at 2): controls the intensity of the dirt texture

CHANNEL MIXER

A Post processing effect that manipulates the color channels independently. The values are set as such:

- Chromatic Aberration: simulates a color distortion in lenses
- Intensity (valued at 0.33): controls the intensity of the effect

COLOR ADJUSTMENTS

A Post processing effect that applies various color-related parameters. The values are set as such:

- Post Exposure (valued at 0.25): adjusts the overall exposure of the image after rendering
- Contrast (valued at 38): makes the picture appear more washed out
- Color Filter (chose white): adds a white hue to the image
- Hue Shift (valued at 0): shifts the hue of all the colors
- Saturation (valued at 100): controls the purity of color in the image

DEPTH OF FIELD

A Post processing effect that simulates the way a camera lens focuses on objects at different distances. The values are set as such:

- Mode (chose bokeh): adds a more realistic effect
- Focus Distance (valued at 10): specifies the distance from the camera to the point of focus
- Focal Length (valued at 36): controls the magnification of the scene
- Aperture (valued at 6,9): refers to the size of the opening of the lens, here the value adds a more blurred background and a narrower field of view
- Blade Count (valued at 4): produces smoother and less distinct shapes
- Blade Curvature (valued at 0.214): controls the roundness of the shapes
- Blade Rotation (valued at -34): changes the orientation of the bokeh highlights

FILM GRAIN

A Post processing effect that adds a grainy texture. The values are set as such:

- Type (chose Thin 1): adds a finer grain pattern
- Intensity (valued at 0.414): controls the intensity of the effect
- Response (valued at 0.584): produces an even distribution of grain

LENS DISTORTION

A Post processing effect that simulates the distortion that occurs when light passes through a lens. The values are set as such:

- Intensity (valued at -0.2): controls the strength of the effect
- X Multiplier (valued at 1): distortion is applied evenly across the X axis
- Y Multiplier (valued at 1): distortion is applied evenly across the Y axis
- Center (valued at X:0.5, Y:0.5): distortion is centered in the middle of the screen
- Scale (valued at 1): distortion is applied at its original size

MOTION BLUR

A Post processing effect that simulates the blurring that occurs when objects move rapidly

- Quality (set as High): the effect is computed with high accuracy
- Intensity (valued at 0.57): controls the intensity of the effect
- Clamp (valued at 0.114): sets the maximum amount of blur, here it is set to 20%

SPLIT TONING

A Post processing effect that allows to independently adjust the color tones of shadows. The values are set as such:

- Shadows: shadows are colored in burnt sienna
- Highlights: applies white tone to the highlights
- Balance (valued at 34): controls the balance between the shadow and the highlight tones

TONEMAPPING

A Post processing effect that maps the HDR and SDR colors

- Mode (set to aces): a standardized color space and color management system

VIGNETTE

A Post processing effect that darkens or lightens the edges of the screen. The values are set as such:

- Color (set to white): the effect will have no tint
- Center (set at X:0.5 Y:0.5): the center of the effect is aligned with the center of the screen
- Intensity (valued at 0.762): controls the intensity of the effect
- Smoothness (valued at 0.307): provides a smooth transition
- Rounded (checked): provides softer edges for a smooth appearance

WHITE BALANCE

A Post processing effect that simulates the adjustment of color temperature. The values are set as such:

- Temperature (valued at 3): adds a cooler tone to the scene
- Tint (valued at 3): adds a slight magenta tint

The matrix included below can make the effect settings easier to understand.

EFFECT	SUB-REGULATIONS	VALUES
BLOOM	THRESHOLD	2
	INTENSITY	5
	SCATTER	0.25
	TINT	BLACK
	CLAMP	65468
	SKIP ITERATIONS	12
	DIRT INTENSITY	2
CHANNEL MIXER	INTENSITY	0.33
COLOR ADJUSTMENTS	POST EXPOSURE	0.25
	CONTRAST	38
	COLOR FILTER	WHITE
	HUE SHIFT	0
	SATURATION	100
DEPTH OF FIELD	MODE	BOKEH
	FOCUS DISTANCE	10
	FOCUS LENGTH	36
	APERTURE	6.9
	BLADE COUNT	4
	BLADE CURVATURE	0.214
	BLADE ROTATION	-34
FILM GRAIN	TYPE	THIN 1
	INTENSITY	0.414
	RESPONSE	0.584
LENS DISTORTION	INTENSITY	-0.2
	X MULTIPLIER	1
	Y MULTIPLIER	1
	CENTER	X:0.5, Y:0.5
	SCALE	1
	QUALITY	HIGH
	INTENSITY	0.57

	CLAMP	0.114
SPLIT TONING	SHADOWS HIGHLIGHTS BALANCE	BURNT SIENNA COLOR WHITE 34
TONEMAPPING	MODE	ACES
VIGNETTE	COLOR CENTER INTENSITY SMOOTHNESS ROUNDED	WHITE X:0.5, Y:0.5 0.762 0.307 CHECKED
WHITE BALANCE	TEMPERATURE TINT	3 3

Table 9:Post-Processing Effects for Medium Version

Here, some snippets of the game set in Medium can be seen.

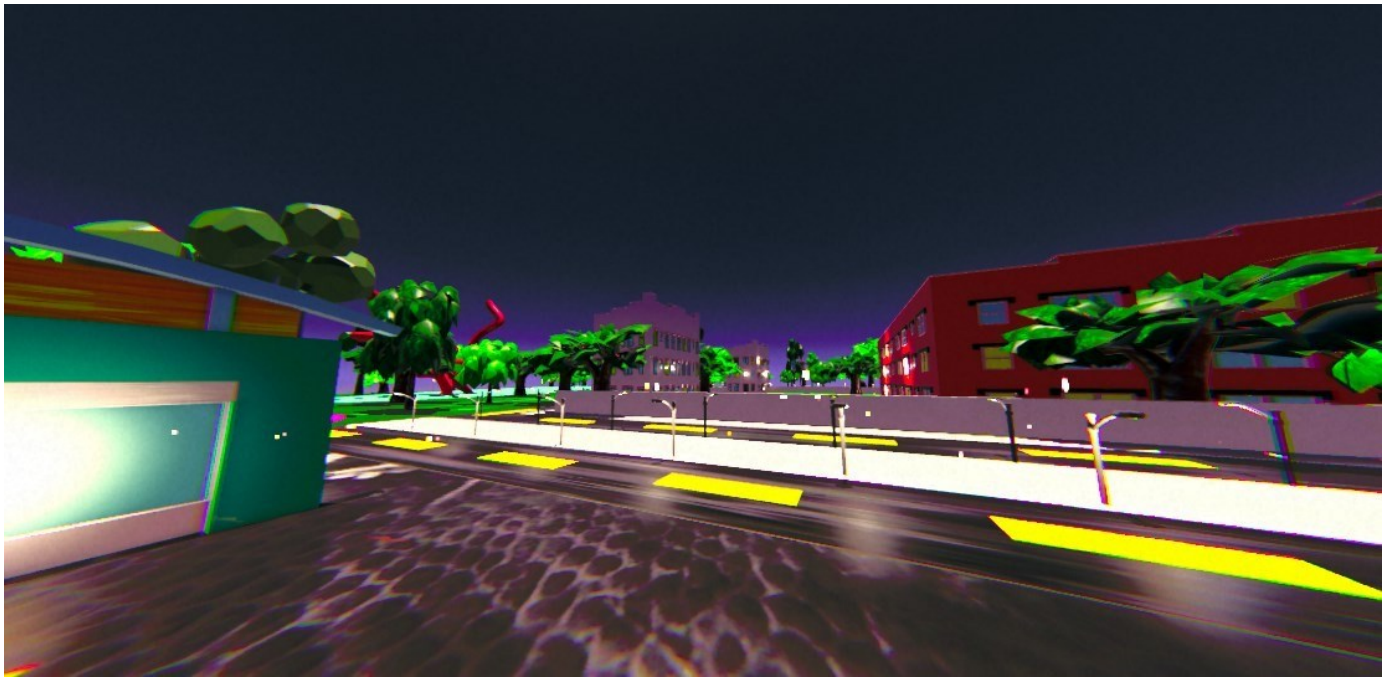

Figure 12:Medium Version of Post-Processing Effects

Figure 13:Medium Version of Post-Processing Effects

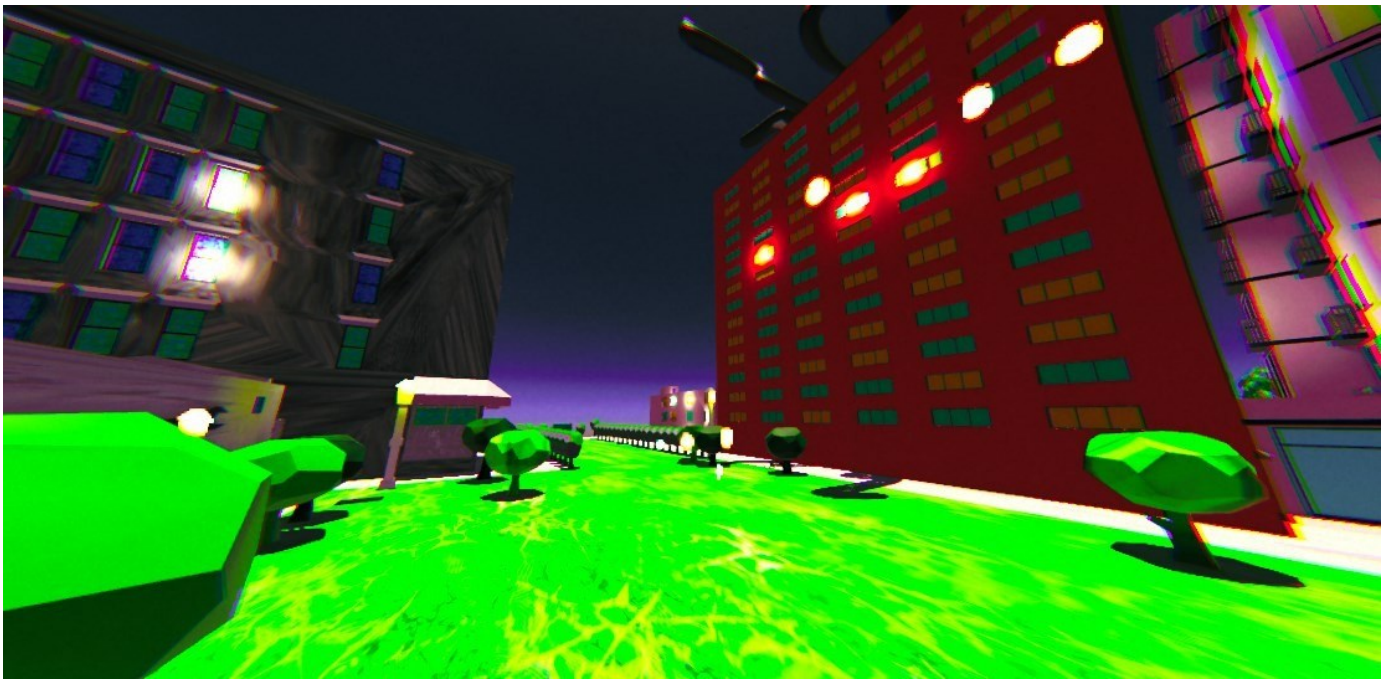

Figure 14:Medium Version of Post-Processing Effects

C. Low Volume Profile

Below, all effects used in order to create the visual changes that occur when the symptomatolgy is set to "Low" are examined.

COLOR ADJUSTMENTS

A Post processing effect that applies various color-related parameters. The values are set as such:

- Post Exposure (valued at 0): adjusts the overall exposure of the image after rendering
- Contrast (valued at 8): makes the picture appear more washed out
- Color Filter (chose white): adds a white hue to the image
- Hue Shift (valued at 0): shifts the hue of all the colors
- Saturation (valued at 51): controls the purity of color in the image

DEPTH OF FIELD

A Post processing effect that simulates the way a camera lens focuses on objects at different distances. The values are set as such:

- Mode (chose bokeh): adds a more realistic effect
- Focus Distance (valued at 5): specifies the distance from the camera to the point of focus
- Focal Length (valued at 75): controls the magnification of the scene
- Aperture (valued at 4.5): refers to the size of the opening of the lens, here the value adds a more blurred background and a narrower field of view
- Blade Count (valued at 5): produces smoother and less distinct shapes
- Blade Curvature (valued at 0.35): controls the roundness of the shapes
- Blade Rotation (valued at -3): changes the orientation of the bokeh highlights

FILM GRAIN

A Post processing effect that adds a grainy texture. The values are set as such:

- Type (chose Thin 2): adds a finer grain pattern
- Intensity (valued at 0.125): controls the intensity of the effect
- Response (valued at 0.107): produces an even distribution of grain

LENS DISTORTION

A Post processing effect that simulates the distortion that occurs when light passes through a lens. The values are set as such:

- Intensity (valued at -0.06): controls the strength of the effect
- X Multiplier (valued at 0): distortion is applied evenly across the X axis
- Y Multiplier (valued at 0.038): distortion is applied evenly across the Y axis
- Center (valued at X:0.5, Y:0.5): distortion is centered in the middle of the screen
- Scale (valued at 1.5): distortion is applied at its original size

MOTION BLUR

A Post processing effect that simulates the blurring that occurs when objects move rapidly

- Quality (set as High): the effect is computed with high accuracy
- Intensity (valued at 0): controls the intensity of the effect
- Clamp (valued at 0): sets the maximum amount of blur (here it is set to 20%)

SPLIT TONING

A Post processing effect that allows to independently adjust the color tones of shadows. The values are set as such:

- Shadows: shadows are colored in burnt sienna
- Highlights: applies white tone to the highlights
- Balance (valued at 47): controls the balance between the shadow and the highlight tones

TONEMAPPING

A post processing effect that maps the HDR and SDR colors

- Mode (set to aces): a standardized color space and color management system

VIGNETTE

A Post processing effect that darkens or lightens the edges of the screen. The values are set as such:

- Color (set to white): the effect will have no tint
- Center (set at X:0.5 Y:0.5): the center of the effect is aligned with the center of the screen
- Intensity (valued at 0.06): controls the intensity of the effect
- Smoothness (valued at 0.01): provides a smooth transition
- Rounded (checked): provides softer edges for a smooth appearance

WHITE BALANCE

A post processing effect that simulates the adjustment of color temperature. The values are set as such:

- Temperature (valued at 29): adds a cooler tone to the scene
- Tint (valued at 0): adds a slight magenta tint

The matrix included below can make the effect settings easier to understand.

EFFECT	SUB-REGULATIONS	VALUES
COLOR ADJUSTMENTS	POST EXPOSURE	0
	CONTRAST	8
	COLOR FILTER	WHITE
	HUE SHIFT	0
	SATURATION	51
DEPTH OF FIELD	MODE	BOKEH
	FOCUS DISTANCE	5
	FOCUS LENGTH	75
	APERTURE	4.5
	BLADE COUNT	5
	BLADE CURVATURE	0.35
	BLADE ROTATION	-3

FILM GRAIN	TYPE	THIN 2
	INTENSITY	0.125
	RESPONSE	0.107
LENS DISTORTION	INTENSITY	-0.06
	X MULTIPLIER	0
	Y MULTIPLIER	0.038
	CENTER	X:0.5, Y:0.5
	SCALE	1
	QUALITY	HIGH
	INTENSITY	0
	CLAMP	0
SPLIT TONING	SHADOWS	BURNT SIENNA
	HIGHLIGHTS	WHITE
	BALANCE	47
TONEMAPPING	MODE	ACES
VIGNETTE	COLOR	WHITE
	CENTER	X:0.5, Y:0.5
	INTENSITY	0.06
	SMOOTHNESS	0.01
	ROUNDED	CHECKED
WHITE BALANCE	TEMPERATURE	29
	TINT	0

Table 10:Post-Processing Effects for Low Version

Here, some snippets of the game set in Low can be seen.

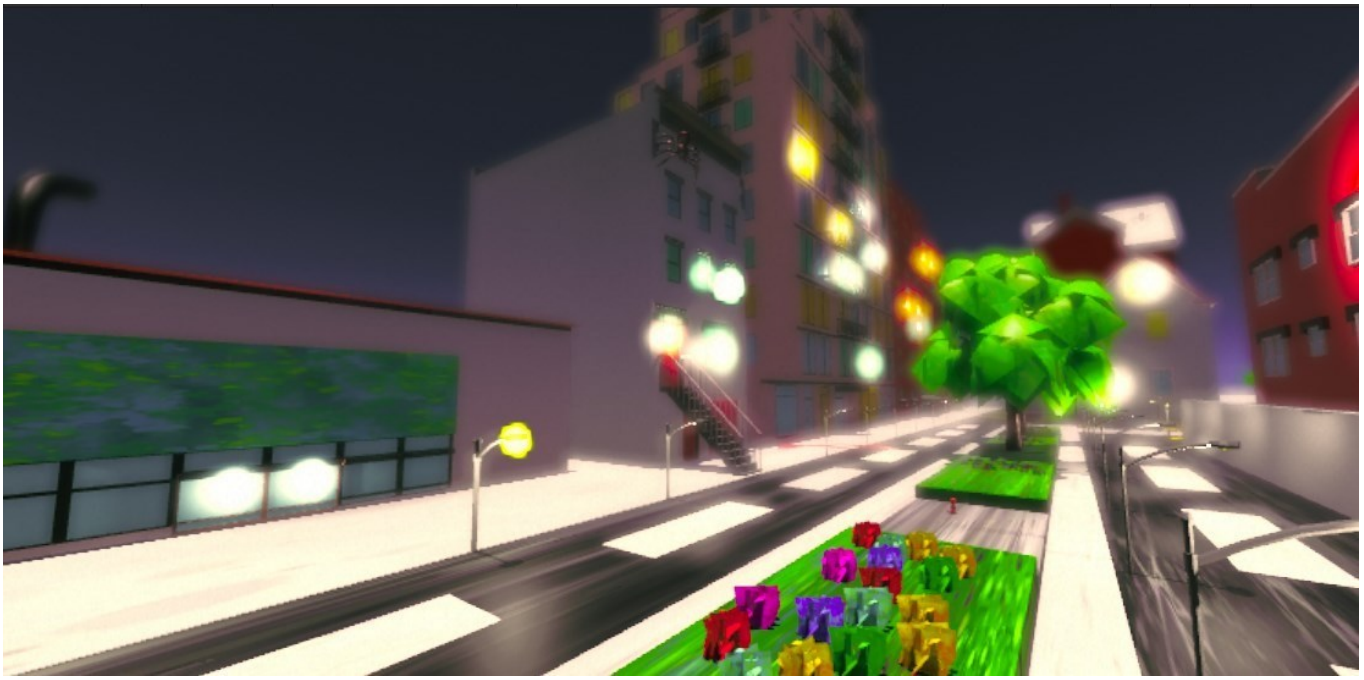

Figure 15:Low Version of Post-Processing Effects

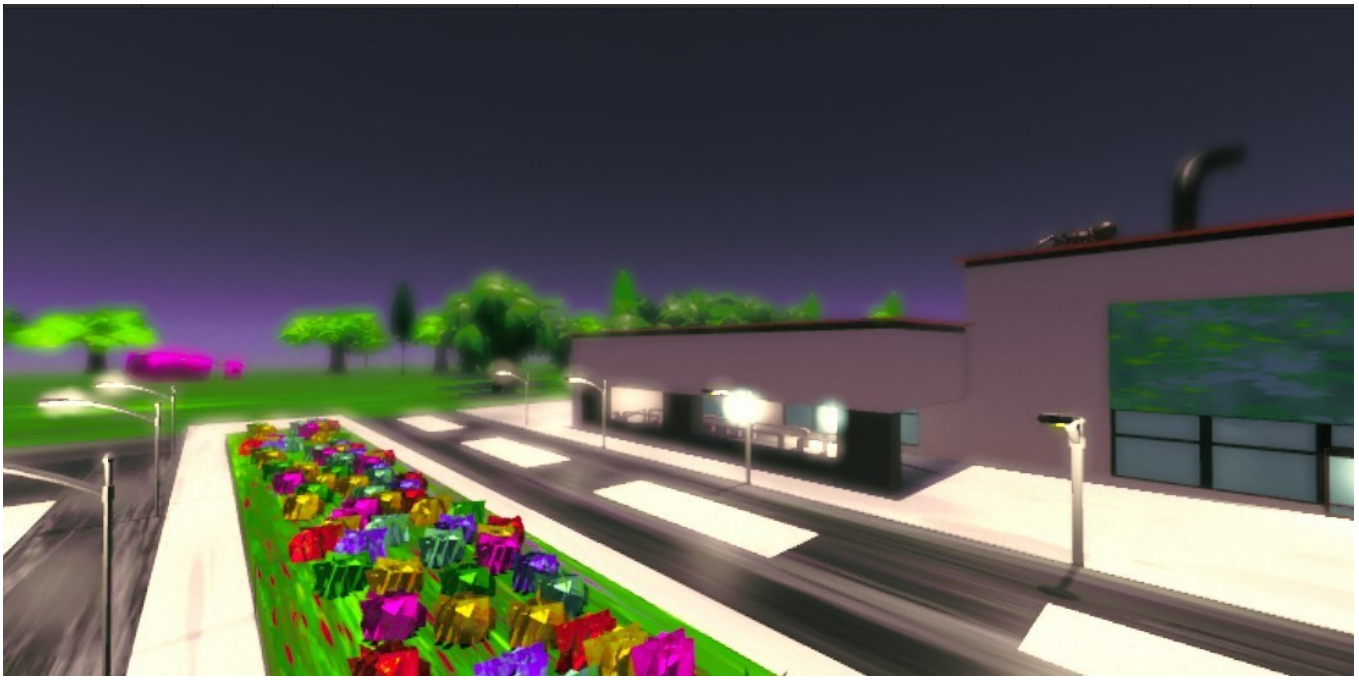

Figure 16:Low Version of Post-Processing Effects

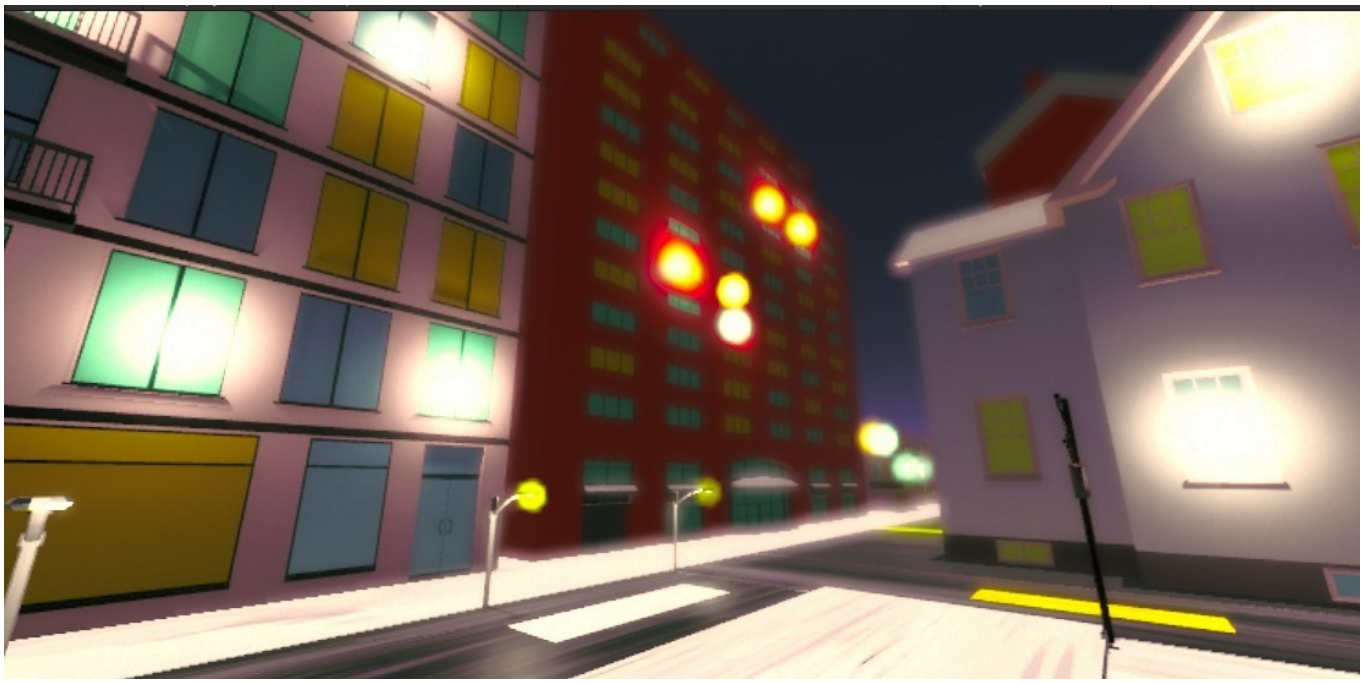

Figure 17:Low Version of Post-Processing Effects

Lastly, some snippets of the game without the use of Post-Processing effects are shown for comparison.

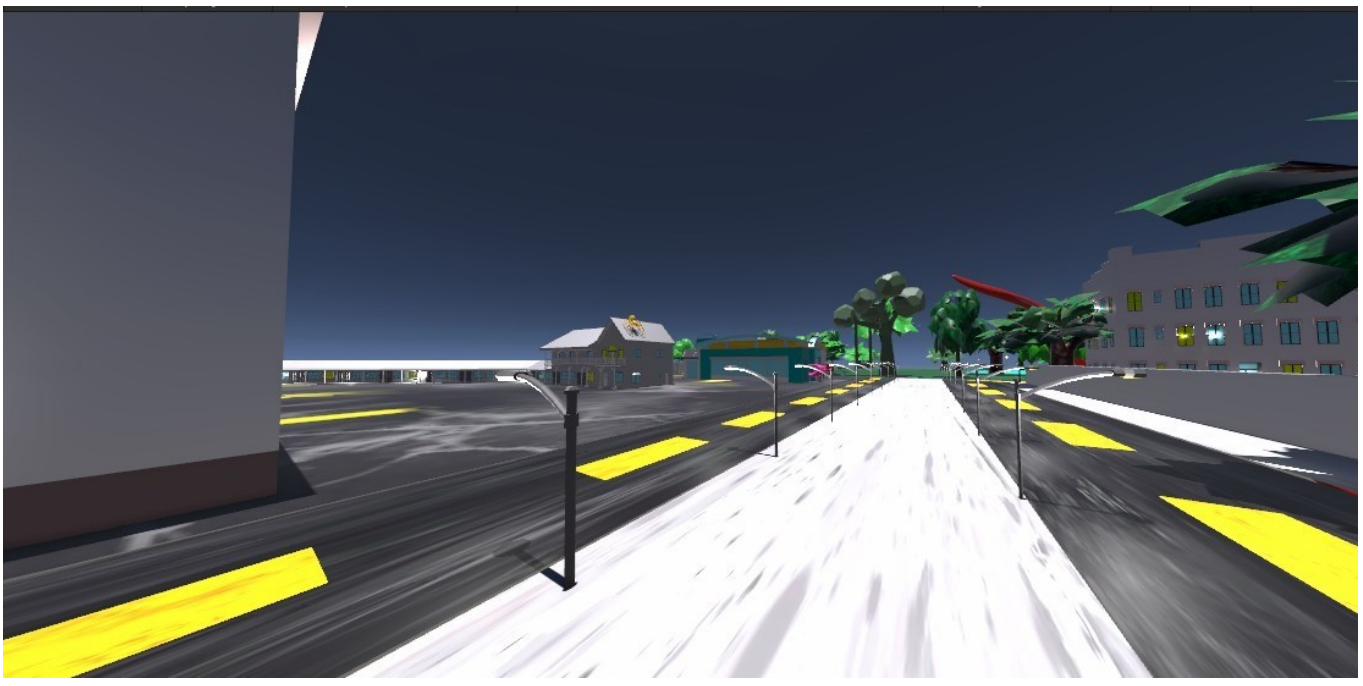

Figure 18:Game Version without Post-Processing Effects

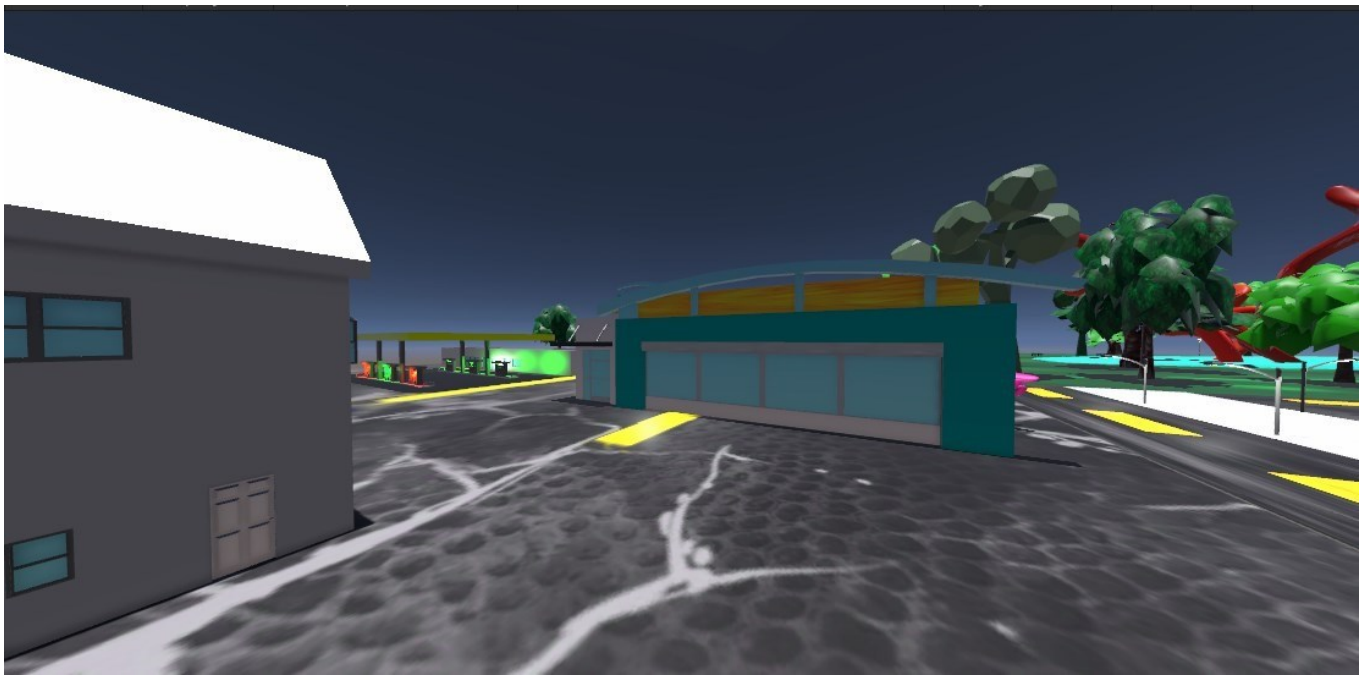

Figure 19:Game Version without Post-Processing Effects

XIV. VR EXPERIENCE

During the research for this thesis, I had the privilege of collaborating with inspiring academic personnel who supported me in transforming my 3D serious game into a VR experience. The Informatics and Computer Science Department at the University of West Attica granted me access to an experimental lab, allowing me to work with high-end technology and VR equipment. Specifically, the lab was equipped with the Oculus Quest 3 VR headset. Once the 3D game was completed and fully functional as a desktop application, I began the process of customizing it for a VR experience.

A. VR Headset Connection

The first step to customizing "Bipolar Odyssey" as a VR Experience was enabling the headsets' connection to the Labs' Computer. To connect the Oculus Quest 3 headset to a Windows PC for the purpose of utilizing PCVR applications and games, several steps are required. Firstly, I ensured the headset was up to date with the latest software update, in order to ensure compatibility and high-end performance. For the connection, a USB-C cable was used. Additionally, the Quest Link PC app[25] had to be downloaded and installed on the computer. The setup process involves both the PC and the headset. On the PC, once the Quest Link app opened, academic personnel gave me access to their credentials in order to use their Meta account. Once the headset was on, I made sure to familiarize with the headsets' functionalities, and continued to access the universal menu by pressing the appropriate button on the right controller, then navigate to Quick Settings. From there, I selected System, scroll to Quest Link, and choose the connected PC to initiate the link by selecting Launch. This setup allows for a seamless connection between the Oculus Quest 3 headset and the PC, enabling the use of advanced VR applications and enhancing the overall virtual reality experience.

B. Unity Setup

To set up Unity3D for Virtual Reality (VR) development[26], several critical steps and configurations must be undertaken to ensure compatibility and optimal performance. The first step was ensuring both Unity3D and the VR headset were both up to date with all software installations. Then in the Edit Menu:

Navigate to the 'Project Settings' >Under the 'XR Plug-in Management' tab, enable the VR plugin by checking the box for the appropriate platform, in this case Oculus. Through the Asset Store, I made sure to download and Install the Oculus Integration package, in order to gain access to all appropriate Prefabs and incorporated Scripts. After installation, the SDK was configured in the Project Settings under XR Plug-in Management, enabling the appropriate XR SDKs for the Android platform.

The next step was the proper configuration of the Player Settings. In the Edit Menu:
Edit > Project Settings > Player. Navigate to the XR Settings section and enable the Virtual Reality support option by checking the "Virtual Reality Supported" box.
A very important step was checking whether the SDKs and JDKs used were compatible with all prerequisites set by the Oculus Quest 3. When setting up a development environment for VR using Unity3D, ensuring that the Software Development Kit (SDK) and Java Development Kit (JDK) are correctly configured is critical for a smooth and efficient workflow. To begin with, it should be verified that the latest version of the JDK is installed, as it is essential for compiling and running Java-based components, which are often required for Android-based VR platforms like Oculus Quest. In case the latest JDK release is not being used, it should be downloaded from the official Oracle website or an open-source alternative, ensuring the version matches the requirements specified by the VR SDK documentation. For the configuration of the JDK path in Unity: In the *Edit menu > Preferences > External Tools* and set the JDK location to the installed directory of your choosing.

It should be noted that “Bipolar Odyssey” was built for an Android-based VR headset, meaning that the installation of Android Studio which includes the Android SDK Manager was needed. Using the SDK Manager the latest SDK platforms, build tools, and emulator images that are compatible with the Oculus VR device were downloaded. Lastly, it was ensured that the SDK path is correctly set in Unity by going to *Edit menu > Preferences > External Tools* and pointing the SDK location to the directory where Android Studio installed the SDK.

C. 3D Main Scene to VR Main Scene

For the Main Scene to be functional while using the Oculus Quest 3 VR Headset, some changes were deemed important. In order to transform the 3D Unity Game into a VR experience, several adjustments and configurations were made. After deleting or disabling the existing main camera, the VR Camera Rig prefab was imported and placed from the installed VR SDK package into the scene. Additionally, for functionality reasons, several C# scripts, both provided by Oculus and created by me, were featured, such as:

- *OVR Manager (Oculus Provided):* This script is a necessary component in managing Oculus VR functionalities, such as tracking, input, rendering, VR display and interactions within the Oculus platform.
- *OVR Headset Emulator (Oculus Provided):* This C# script allows VR Headset simulation, without requiring the connection of an actual VR Headset.
- *Joystick:* The “Joystick.cs” script, created by me, is responsible for the controlled movement of the character using the VR headsets’ joysticks, by retrieving horizontal and vertical input values, in order to create movement along the X and Y axis.
- *Head Movement:* The “HeadMovement.cs” script, created by me, manages the rotation of the characters’ head, by calculating the offset between the body and the head of the character.
- *OVR Hand (Oculus Provided):* This script is designed to manage the hand tracking functionality while using the VR Headset, with key features including hand tracking integration, interaction with VR object, gesture detection and more.
- *OVR Custom Skeleton (Oculus Provided):* This Oculus script customizes the hand tracking system in VR headsets, using custom hand models, customizable parameters, animation controls etc.
- *Tremble:* The ‘Tremble.cs’ C# script, created by me, was designed to provide vibrations when holding the VR headsets’ joysticks. The duration, frequency and strength of the vibration can be manipulated through the script to provide a better simulation of an increased heartbeat during the VR experience.

- *OVR Grabbable (Oculus Provided):* This script is essential for all interactions within the VR world, such as grabbing and manipulating VR objects.
- *OVR Player Controller (Oculus Provided):* Oculus provides this script in order to simplify all player actions, such as rotation, teleportation and movements, ensuring the smooth functionality of the VR experience.

D. Bugs And Possible Fixes

During the transformation of the 3D game to a VR experience, numerous steps and actions were taken in order for it to be fully functional. Despite numerous changes in scripts, adjustments in scale and continuous research, some bugs that occurred during the transformation could not be resolved. The main bug is related to the players' head movement and more specifically the way that the VR Headset interacts with the environment when head movement is detected. Even though the player can perfectly control the body movement using the VR Headsets' joysticks, when they turn their head in order to look around, the characters' position changes as well, as in a teleporting mode. This way, when head movement is detected, the overall position of the character changes. After several trials and research, the root of the problem is the fact that the game was initially developed using Unitys' 3D module, instead of a pre-existing VR module and thus, the bug could not be resolved. Except from developing the game in a VR module, several steps could be followed to resolve the issue such as:

- Separating the Head Rotation from Body Movement: One possible fix that was tried but did not completely resolve the issue, was ensuring that the head rotation and body movement are decoupled.
- Adjusting the characters' Parent-Child Hierarchy: By reorganizing the hierarchy, a more clear and organized structure is achieved, in order to make sure that the camera is completely independent of the body transform.
- Misalignment of Center of Movement: It should be ensured that the headsets' center is aligned with the players' body.

Apart from that, minute errors were noticed, such as a rather high rotation speed, a higher intensity in colors than expected and a more intense blurriness of the vision. In order to eliminate these bugs, the rotation speed could be determined with the use of a SpeedRotation variable to better control it through the C# script and additionally, changes could be made in the post-processing effects used to represent the altered sensory phenomena.

XV. C# SCRIPTS

In this particular segment, all scripts created and used for the functionality of the serious game are going to be briefly analyzed.

A. ApplyBrightness.cs Script

The ApplyBrightness script is designed to adjust the intensity of the main light source in all Unity scenes based on a saved value. In the Start() method, the script calls the ApplyBrightnessValue() to apply the brightness settings when the game begins. Within ApplyBrightnessValue(), the script first checks if the sceneLight reference is assigned and if it is not, an error message is logged. If the light is assigned, it then checks whether a brightness value is stored in Unity's PlayerPrefs under the key "Intensity". If such a value exists, it retrieves it, adds 1 to it, and sets this new value as the light's intensity. The script also logs the applied brightness value for debugging purposes. If the brightness value is not found or the light is not assigned, appropriate warning or error messages are logged to help identify issues. The most important code fragment is featured below:

```
private void ApplyBrightnessValue() // Private method that applies the saved
brightness value
    {
```

```
        if (sceneLight != null) // Checking whether if sceneLight value has been
assigned
        {
            if (PlayerPrefs.HasKey("Intensity")) // Checking whethet a saved value
pre-exists in PlayerPrefs
            {
                float intensity = PlayerPrefs.GetFloat("Intensity") + 1; // Calculate
the new value by finding the saved intensity value from PlayerPrefs and increasing it
by 1
                sceneLight.intensity = intensity; // Set the light intensity
                Debug.Log("Applied brightness: " + intensity); // Debug message

            }
            else
            {
                Debug.LogWarning("Brightness value not found in PlayerPrefs."); //
Debug message
            }
        }
        else
        {
            Debug.LogError("Scene light is not assigned."); // Debug message
        }
                                            }
```

B. ApplyEffect.cs Script

The ApplyEffect script is created to manage the application of different post-processing effects based on the player's preferred sensitivity settings. The script uses three predefined post-processing Volume profiles (low, medium, and high), that are assigned to the Post Processing Volume component, which handles rendering visual effects in the Main Scene. When the game starts, the script retrieves the saved quality setting from PlayerPrefs, a system that stores player preferences. Depending on the retrieved setting, the script applies the appropriate post-processing profile, and in case that no setting is found, the default high sensitivity profile is applied. The most important code fragment is featured below:

```
private void ApplySavedPostProcessingProfile() //Definition of a private method to apply
the saved post-processing profile based on the player's quality setting
    {
        string profileSetting = PlayerPrefs.GetString("QualitySetting", "High"); //
Retrieving the saved quality setting from PlayerPrefs, and if there is not one, the
default profile is used

        switch (profileSetting) // Switch statement to apply the appropriate post-
processing profile
        {
            case "Low":
                postProcessingVolume.profile = low; // Applying low profile
                break;
            case "Medium":
                postProcessingVolume.profile = medium; // Applying medium profile
                break;
            case "High":
                postProcessingVolume.profile = high; // Applying high profile
                break;
            default:
                postProcessingVolume.profile = medium; // Default to high profile in
case of some mistake
                break;
        }
```

```
        Debug.Log("Applied Post-Processing Profile: " + profileSetting);// Debugging
message
       }
```

C. *ApplyVolume.cs Script*

The ApplyVolume script, is in charge of applying a saved music volume setting throughout all of the games' scenes. It references an AudioSource component, that is responsible for controlling the music playback in the game. When the game begins, a method is called to apply the pre-existing volume level from PlayerPrefs, which stores player preferences, and if the saved volume value exists, the script retrieves and applies it to the musicSource. The most important code fragment is featured below:

```
private void ApplyVolumeValue() // Definition of private method to apply the saved
volume value to the music source
    {
        if (musicSource != null)   // Checking whether a musicSource value has been
assigned
        {
            if (PlayerPrefs.HasKey("Volume")) // Checking whther a saved "Volume" value
pre-exists in PlayerPrefs
            {
                float volume = PlayerPrefs.GetFloat("Volume"); // Retrieving the saved
volume value from PlayerPrefs, if it already exists
                musicSource.volume = volume; // Setting the volume of the AudioSource
to the retrieved value
                Debug.Log("Applied volume: " + volume); // Debugging message
            }
            else
            {
                Debug.LogWarning("Volume value not found in PlayerPrefs.");// Debugging
message
            }
        }
        else
        {
            Debug.LogError("Music source is not assigned.");// Debugging message
        }
       }
```

D. *AvatarController.cs Script*

This particular script was created in order to control the behavior of the player when interacting with a moving car and is attached to the avatar GameObject. The code utilizes various namespaces to access essential Unity and .NET functionalities, such as System.Collections and System.Collections.Generic, namespaces that provide foundational data structures, type safety and performance benefits. Additionally, namespaces that contain the core functionality of the Unity Engine and allow access to Unity-specific features were imported. The 'AvatarController' class includes a private method named OnTriggerEnter, which is called when a collider enters the trigger area of the player. Then, the code checks if the collider belongs to a GameObject with the tag "Car" and if so, the "honk" sound is played at the position of the avatar. Additionally, it logs a debug message to the console for debugging purposes. The most important code fragment is featured below:

```
private void OnTriggerEnter(Collider other) //Here, an 'OnTriggerEnter' Method is used
    {
        if (other.CompareTag("Car")) //Checking whether the collider that triggered the
event has the 'Car' tag
        {
```

```
            AudioSource.PlayClipAtPoint(honk, transform.position);} //The 'honk' sound
is played
```

E. AvatarEnd.cs Script

The AvatarEnd script is designed to control the behavior of a GameObject with a Rigidbody component when it interacts with a specific boundary within the game. Upon the GameObject colliding with a trigger collider tagged as "Limit," the script stops the GameObject's movement by setting its Rigidbody's velocity to zero, effectively stopping it. This functionality is useful for ensuring that the GameObject does not exceed certain limits within the game environment. Additionally, the script includes a debug statement to output a message when the collision occurs. The most important code fragment is featured below:

```
private void OnTriggerEnter(Collider other) // Definition of a private method that is
called when another collider enters the trigger collider attached to this GameObject
    {
        if (other.CompareTag("Limit"))          // Checking whether the other collider
has the tag "Limit"
        {
            rb.velocity = Vector3.zero; // Stop movement by setting the velocity to zero
        }

        Debug.Log("oria2");          // Debugging message
       }
```

F. BrightnessSlider.cs Script

The BrightnessSlider script is created to control and adjust the brightness of a scene with the use of a UI slider. At the beginning of the game, the script checks whether a saved brightness value by the name of Intensity already exists in PlayerPrefs. If not, a default value of 1 is set and the value is loaded into the slider, otherwise, if the value exists, the stored brightness setting is loaded. The Load() method sets the slider's value to the stored brightness level, while the Save() method updates this stored value with the current slider value and ensures it's saved. The Adjust() method modifies the intensity of the Light component based on the slider's and then calls Save() to update the stored brightness setting, allowing the brightness to be adjusted and applied in the same way throughout all of the games' scenes. The most important code fragment is featured below:

```
void Start()
    {
        if (!PlayerPrefs.HasKey("Intensity")) // Check whether the "Intensity" value
pre-exists in PlayerPrefs
        {
            PlayerPrefs.SetFloat("Intensity", 1);    // If not, setting the default
intensity to 1
            Load(); //Loading the default value
        }
        else
        {
            Load(); // Otherwise if the intensity value pre-exists, load it
        }
    }
    public void Adjust() // Definition of a private method that adjusts the brightness
based on the slider value
    {
        sceneLight.intensity = brightness_slider.value + 1; // Setting the scene light's
intensity based on the slider value and adding 1 to avoid a value of 0
        Save(); }} // Saving the value
```

G. Boundaries.cs Script

The 'Boundaries.cs' script is designed to limit the player within the neighborhoods' boundaries. Various namespaces are used in order to access essential Unity and .NET functionalities, such as System.Collections and System.Collections.Generic, which provide core functionalities and performance benefits. The 'Boundaries' class is responsible for defining boundaries within the 3D environment and handling collisions with these boundaries, by including a private method called 'OnTriggerEnter', which is called when a collider enters the boundary. Within this method, the code checks whether the collider belongs to a GameObject with the tag "Player", and if so the Rigidbody component attached to the GameObject is retrieved and the velocity value of the Rigidbody component is set to 0 to prevent the player from moving beyond the boundary. The most important code fragment is featured below:

```
private void OnTriggerEnter(Collider other) //Private method that is called when a collider enter the neighborhood boundary
    {
        if (other.CompareTag("Player")) //The GameObjects' Tag gets compared and gets checked if its name is Player,in order to examine whether the collider belongs to the Player
        {
            Rigidbody rb = other.GetComponent<Rigidbody>(); //Retrieving the Rigidbody component attached to the GameObject that collided with the collider
            if (rb != null) //Checking whether the rb variable holds a valid reference to a Rigidbody component
            {
                rb.velocity = Vector3.zero; //Velocity is stopped
            }
           }
```

H. CarController.cs Script

The 'CarController.cs' script is created to manage the movement of a passing car along a predefined route around the neighborhoods' block. The UnityEngine namespace was used, in order to allow basic Unity functionalities, while the 'CarController' class includes various public variables, such as speed, and a private integer to manage the car's movement along waypoints. A 'waypoints' array is declared, that stores the positions of waypoints representing the route that the car follows. In the Update( ) method the code ensures whether the waypoints array is empty or not, and if it is, the method returns without executing any further code. Then, the position of the current waypoint is calculated and the car is moved towards it. Once the car reaches the current waypoint, it updates the currentWaypointIndex to move to the next waypoint, while ensuring that the car continuously loops through the waypoints by resetting the index to zero when reaching the last waypoint. Lastly, the direction vector is calculated towards the next waypoint, the car rotates to face that direction accordingly, and then moves forward. The most important code fragment is featured below:

```
if (currentWaypointIndex < waypoints.Length) //Checking whether the current waypoint is thelast one
        {
            Transform currentWaypoint = waypoints[currentWaypointIndex]; //Finding out which is the current waypoint
            Vector3 direction = (currentWaypoint.position - transform.position).normalized; //Calculating the direction vector from the current position toward the next waypoint
            Quaternion targetRotation = Quaternion.LookRotation(direction); //Calculating the target rotation
            transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, Time.deltaTime * rotationSpeed); //Smoothly rotating the object
```

```
            transform.Translate(Vector3.forward * moveSpeed * Time.deltaTime); //Moving
the object fowards

            if (Vector3.Distance(transform.position, currentWaypoint.position) < 0.1f)
//Checking if the object has reaced the current waypoint
            {
                currentWaypointIndex++;//Incresing the value of the waypoint array by
1, so that the object continuously loops thrpugh the waypoints

                if (currentWaypointIndex >= waypoints.Length)
                {
                    currentWaypointIndex = 0; // Reset index to loop back to the first
waypoint
                }
            } }
```

I. *Controller.cs Script*

The 'Controller.cs' script is in charge of managing the sensitivity settings of the game through a simple UI interface with three choices. Each choice is represented by a button, corresponding to Low, Medium, and High levels of the post-processing effects used. A class named "controller" is defined, which contains three private button variables and by using the Start() method, the corresponding quality setting is passed as an argument to the SetQuality() method. This method saves the selected quality setting and logs the selected quality to the console. The most important code fragment is featured below:

```
private void Start()
    {
        lowButton.onClick.AddListener(() => SetQuality("Low"));    // Adding a listener
to buttons and when lowButton is pressed, the quality gets set to "Low"
         mediumButton.onClick.AddListener(() => SetQuality("Medium")); // Adding a
 listener to buttons and when mediumButton is pressed, the quality gets set to "Medium"
       highButton.onClick.AddListener(() => SetQuality("High"));  // Adding a listener
to buttons and when highButton is pressed, the quality gets set to "High"
    }

    public void SetQuality(string quality) // Definition of a private method to save
the selected quality setting to PlayerPrefs
    {
        PlayerPrefs.SetString("QualitySetting", quality); // Saving the selected
quality setting as a string in PlayerPrefs
        PlayerPrefs.Save();
        Debug.Log("Quality set to: " + quality); // Debugging message
    }
```

J. *ESC_exit.cs Script*

The 'ESC_exit.cs' script is designed to manage scene transitions and application quitting based on user input. It specifically responds to the Escape key press. When the Escape key is detected, the script first makes the cursor visible and unlocks it, ensuring that the user can interact with the UI if needed. The script then checks if the current active scene is the specified menu scene. If the current scene is not the menu, it calls the ChangeScene() method to load the menu scene. If the user is already in the menu scene, the script executes the Exit() method, which quits the application. The most important code fragment is featured below:

```
void Update() // Built-in Unity method that updates the state of the GameObject
    {
        if (Input.GetKeyDown(KeyCode.Escape)) // Checking whether the Escape key is
pressed
```

```
        {
            Cursor.visible = true; // Make the cursor visible
            Cursor.lockState = CursorLockMode.None; // Unlock the cursor

            if (SceneManager.GetActiveScene().name != menuSceneName) // Checking if the
current scene is the menu scene
            {
                ChangeScene(menuSceneName); // If not, change it to the menu
            }
            else
            {
                Exit(); // Otherwise exit the application
            }
        }
    }

    public void ChangeScene(string sceneName) // Method to change the active scene to
the one specified by 'sceneName'
    {
        SceneManager.LoadScene(sceneName); // Loading the specified scene
       }
```

K. Exit.cs Script

The 'Exit.c' script is a basic script used to exit the application when the player presses the Yes button in the Quit Scene. It works by defining a public method called QuitGame which, when called, triggers Application.Quit(). The most important code fragment is featured below:

```
public void QuitGame()//Public method that exits the application
    {
        Application.Quit(); //Quiting the application
       }
```

L. Firstp.cs Script

The 'firstp.cs' script controls the character's movement and camera look functionality in a first-person gaming technic. This script allows the player to move the character using the keyboard inputs for horizontal and vertical directions. Two variables are declared, one that controls the walking speed and one for the running speed. Gravity is also applied to simulate falling when the character is not on the ground. Additionally, the script manages the player's view direction based on mouse input, where horizontal mouse movement rotates the character around the Y-axis and vertical mouse movement tilts around the X-axis. Another crucial variable controls the maximum and minimum values of the rotation on both axes. Also, the cursor is initially locked and hidden to prevent distractions while looking around. The most important code fragment is featured below:

```
void Update() //Built-in Unity method that updates the state of the GameObject
    {
        Vector3 forward = transform.TransformDirection(Vector3.forward); //Calculating
the foward direction of the character
        Vector3 right = transform.TransformDirection(Vector3.right); //Calculating the
right direction of the character
        float horizontalInput = Input.GetAxis("Horizontal"); //Horizontal input  of the
character
        float verticalInput = Input.GetAxis("Vertical"); //Vertical input    of the
character
        Quaternion cameraRotation = playerCamera.transform.rotation; //Calculating the
rotation of the characters' camera
```

```
        Vector3 moveDirection = cameraRotation * new Vector3(horizontalInput, 0f, verticalInput); //Creating a vector that represents the movement direction and multiplying it according to the camera rotation

        bool isRunning = Input.GetKey(KeyCode.LeftShift); //Check whether the Left Shift button is pressed down and if it is, the isRunning is set to true, so that the character runs
        float curSpeedX = canMove ? (isRunning ? runningSpeed : walkingSpeed) * Input.GetAxis("Vertical") : 0; //Calculating the current speed in the X-axis
        float curSpeedY = canMove ? (isRunning ? runningSpeed : walkingSpeed) * Input.GetAxis("Horizontal") : 0; //Calculating the current speed in the Y-axis
        float movementDirectionY = moveDirection.y; //Storing the current vertical movement direction

        moveDirection = (forward * curSpeedX) + (right * curSpeedY); //Application of movement speed to the moveDirection variable

        if (Input.GetButton("Jump") && canMove && characterController.isGrounded) //Checking if the Spacebar is pressed, and whether the character is allowed to move and grounded
        {
            moveDirection.y = jumpSpeed; //The character moves upward
        }
        else
        {
            moveDirection.y = movementDirectionY; //The character maints their position
        }

        if (!characterController.isGrounded) //Checking if the character is currently grounded
        {
            moveDirection.y -= gravity * Time.deltaTime; //Simulating gravity
        }

        characterController.Move(moveDirection * Time.deltaTime); //Here, the Move method moves the character in the desired direction

        if (canMove) // Checking if the character is allowed to move
        {
            float mouseX = Input.GetAxis("Mouse X") * lookSpeed * Time.deltaTime; //Retrieving the input that comes from the horizontal movement of the mouse
            float mouseY = -Input.GetAxis("Mouse Y") * lookSpeed * Time.deltaTime; //Retrieving the input that comes from the vertical movement of the mouse
            transform.Rotate(0f, mouseX, 0f); //Rotation of the character point-of-view depending on the horizontal mouse movement
            rotationX += mouseY; //Updating the value of the vertical rotation
            rotationX = Mathf.Clamp(rotationX, -lookXLimit, lookXLimit); //Setting the limits in the Y-axis
            playerCamera.transform.localRotation = Quaternion.Euler(rotationX, 0, 0); //Applying the vertical rotation to the characters camera
        }
       }
```

M. HonkControl.cs Script

The ‘HonkControl.c’ script is created to play a honking sound effect when the car drives by the character. More specifically, when the character with the "Player" tag enters a trigger collider attached to the GameObject the sound effect is played. The script checks if the collider that triggered the event has the "Player" tag using other.CompareTag("Player"). If this condition is met, the honk sound is played, providing auditory feedback when the player interacts with the trigger zone. Concerning the

sound effect, the script includes a public AudioClip variable named 'honk', which stores the sound effect wished to be played. The most important code fragment is featured below:

```
private void OnTriggerEnter(Collider other) //Here, an 'OnTriggerEnter' Method is used
    {
        if (other.CompareTag("Player")) //Checking whether the collider that triggered
the event has the 'Player' tag
        {
            AudioSource.PlayClipAtPoint(honk, transform.position); //The 'honk' sound
is played
        }

        Debug.Log("Trigger entered!2"); //Debugging info
    }
```

N. Info.cs Script

The 'Info.cs' script is used in order to change scenes and exit the application. The ChangeScene method accepts a string parameter representing the name of the scene (in this case, "Info") and uses the SceneManager.LoadScene( ) method to load it. Additionally, it makes the cursor visible when the scene has changed. The second method, Exit( ), allows the game to quit by calling Application.Quit(), effectively closing the application when invoked. The most important code fragment is featured below:

```
public void ChangeScene(string Info) //Public method that loads a new scene title 'Info'
       {
              SceneManager.LoadScene(Info); //Loading the scene named 'Info'
              Cursor.visible = true; //Making the cursor visble
       }
```

O. MainScene.cs Script

The 'MainScene.cs' script is used to change scenes and exit the application. It works the same way as the 'Info.cs' script, with the exception that the scene changes lead to the Main Scene. The most important code fragment is featured below:

```
public void ChangeScene(string MainScene) //Public method that loads a new scene title
'Info'
       {
              SceneManager.LoadScene(MainScene); //Loading the scene named 'Info'
              Cursor.visible = true; //Making the cursor visble
       }
```

P. Menu.cs Script

The 'Menu.cs' script is used to change scenes and exit the application. It works the same way as the 'Info.cs' script, with the exception that the scene changes lead to the Menu Scene. The most important code fragment is featured below:

```
public void ChangeScene(string Menu) //Public method that loads a new scene title 'Info'
       {
              SceneManager.LoadScene(Menu); //Loading the scene named 'Info'
              Cursor.visible = true; //Making the cursor visble
       }
```

Q. MouseLook.cs Script

The 'MouseLook.cs' script is designed to calculate the rotation of the camera based on mouse movement, to represent the characters' head movement. A MouseLook class is defined and offers three rotation modes including rotating on both the X and Y axes, only on the X axis, or only on the Y axis. The default mode rotates the object on both axes. Two variables control the sensitivity of the rotation, eventually calculating how smoothly the camera rotates based on mouse movement. The script also restricts the rotation within defined limits, using two variables on the X-axis and two on the Y-axis. In the Update( ) method, the script reads mouse input and applies rotation accordingly The Start( ) method checks if the GameObject has a Rigidbody component and, if so, freezes its rotation to prevent the physics engine from interfering with the scripted rotations. The most important code fragment is featured below:

```
void Update()
    {
        if (axes == RotationAxes.MouseXAndY) // Checking if both Mouse X and Y axes are
used for rotation
        {
            float rotationX = transform.localEulerAngles.y + Input.GetAxis("Mouse X") *
sensitivityX; // Rotation of the X axis

            rotationY += Input.GetAxis("Mouse Y") * sensitivityY; // Updating  the Y
axis rotation
            rotationY = Mathf.Clamp(rotationY, minimumY, maximumY); // Clamping the Y
axis rotation
            transform.localEulerAngles = new Vector3(-rotationY, rotationX, 0); //
Applying the new rotation values
        }

        else if (axes == RotationAxes.MouseX) // Checking if just the Mouse X axis is
used for rotation
        {
            transform.Rotate(0, Input.GetAxis("Mouse X") * sensitivityX, 0); // Rotating
the object according to the Y-axis
        }
        else
        {
            rotationY += Input.GetAxis("Mouse Y") * sensitivityY; // Updating  the Y
axis rotation
            rotationY = Mathf.Clamp(rotationY, minimumY, maximumY); // Clamping the Y
axis rotation
            transform.localEulerAngles        =        new        Vector3(-rotationY,
transform.localEulerAngles.y, 0); // Apply the Y rotation, while keeping the X and Z
rotations the same
        }
    }
```

R. Quit.cs Script

The 'Menu.cs' script is used to change scenes and exit the application. It works the same way as the 'Info.cs' script, with the exception that the scene changes lead to the Quit Scene. The most important code fragment is featured below:

```
public void ChangeScene(string Quit) //Public method that loads a new scene title 'Info'
       {
              SceneManager.LoadScene(Quit); //Loading the scene named 'Info'
              Cursor.visible = true; //Making the cursor visble
       }
```

S. Setting.cs Script

The 'Menu.cs' script is used to change scenes and exit the application. It works the same way as the 'Info.cs' script, with the exception that the scene changes lead to the Settings Scene. The most important code fragment is featured below:

```
public void ChangeScene(string Settings) //Public method that loads a new scene title
'Info'
	{
		SceneManager.LoadScene(Settings); //Loading the scene named 'Info'
		Cursor.visible = true; //Making the cursor visble
	}
```

T. SoundSlider.cs Script

The 'SoundSlider.cs' script manages the background music volume using a UI slider in the Setting Scene. When the game starts, it checks if a previously saved music volume setting exists in PlayerPrefs. If not, it sets the default volume to 0 and loads the value into the slider. The ChangeVolume( ) method is called whenever the slider's value is changed by the player. It adjusts the global audio volume to the slider's value and saves the new setting. The Load( ) method loads the saved music volume from PlayerPrefs and applies it to the slider's value. Lastly, the Save( ) method saves the current slider value to PlayerPrefs, ensuring that the player's preferences are stored for future sessions. The most important code fragment is featured below:

```
void Start()
    {
        if (!PlayerPrefs.HasKey("musicVolume")) // Checking whether there is a pre-
existing music volume in PlayerPrefs
        {
            PlayerPrefs.SetFloat("musicVolume", 0); // If there is not pre-saved volume,
the default volume is set to 0
            Load(); // Loading the saved or default value
        }
        else
        {
            Load(); // Loading the saved volume setting.
        }
    }

    public void ChangeVolume() // Defintion of a public method that adjusts the volume
slider
    {
        AudioListener.volume = music_slider.value; // Setting the global audio volume
based on the slider's value
        Save(); // Saving the value
    }
```

XVI. UPSCALING

Even though the Demo has already been created and is functional as a Computer Video Game, the ultimate target is the creation of a VR experience that features auditory and visual changes, as well as physical symptoms, e.g. a rising pulse. In order to achieve this effect, a pulse-like tremble has been added to the VR Handset, which will intensify as the game progresses. In this manner, the user will be able to have a full, physical experience of what a Hypomanic Crisis can feel like. Furthermore, a future plan for this project would include more detailed buildings and 3D objects in general, the ability to walk in the multiple houses and places available, while completing mental health-related tasks. With these

additions, a user will be able to gain a full experience of Hypomania, while simultaneously enhancing their knowledge surrounding the Bipolar Spectrum, in order to eradicate years of stigma and trauma.

XVII. TESTING

In order to provide a more thorough scientific analysis of the serious game and VR experiment, I considered it essential for it to be tested with participants and record the results. To better document these results, biometric tracking devices were used, with particular emphasis on measuring heart rates before and after the subjects' participation. Additionally, participants completed questionnaires both before the start of the experience and after its completion. It should also be noted that each participant entered the testing lab alone, without having any interactions with the subjects that had already partaken in the testing. The clinical testing they participated in will be detailed below.

A. Techonological Equipment

For the research part of the serious game, multiple technological devices were used. More specifically, the VR headset, specifically the Oculus Quest 3 model, was used, which the participants used before the start of the experience in order to familiarize themselves. An electronic watch with biometric data capture capabilities was then placed on each participant's wrist. Prior to the subjects' interaction with the game, multiple measurements of their heart rate in a resting state were taken in order to them to be later compared with the results after the experience.

B. Questionnaires

The creation of questionnaires, and their completion by the participants, both before and after interacting with the VR experience, was deemed critical. Below are the questions that were asked in the questionnaires.

a) First Questionnaire (filled before interacting)

- How familiar are you with the Bipolar Spectrum (rate 1 to 5 with 1 being the lowest)?
- How familiar are you with serious gaming (rate 1 to 5 with 1 being the lowest)?
- Are you familiar with the term "Hypomania"?
- What is your age?
- What is your gender?
- What is your highest form of education?
- Have your ever attended therapy?
- Have you ever been diagnosed with a mental health condition?
- How would you rate your current mood on a scale of 1 to 10 (1 being very low and 10 being very high)?
- How high would you rate your current state of anxiety on a scale of 1 to 10 (1 being very low and 10 being very high)?
- How comfortable do you feel around someone who is experiencing symptoms of hypomania?
- How familiar are you with VR Experiences?
- How comfortable are you with using VR technology on a scale of 1 to 10 (1 being not comfortable at all and 10 being very comfortable)?
- Do you have any conditions that might affect your experience with VR (e.g., motion sickness, epilepsy, vision impairments)?
- What are your expectations for this VR experience about hypomania?
- Do you have any concerns or anxieties about participating in this VR experience? If yes, please describe
- What do you hope to learn or understand better after participating in this VR experience?

- Is there anything else you would like us to know before you begin the VR experience?
- Do you have any questions about the VR experience or the study in general?
- Do you consent to participating in this study and the VR experience? (Yes, No)

b) Second Questionnaire (filled after interacting)

- How would you rate your overall experience with the VR simulation?
- How engaging did you find the VR experience?
- How did the VR experience make you feel?
- Did the VR experience help you better understand what hypomania is?
- Did the VR experience increase your empathy towards individuals experiencing hypomania?
- How comfortable did you feel during the VR experience?
- Did the VR experience change your perception of hypomania? If yes, please describe how.
- Did you experience any fear or anxiety during the VR simulation? If yes, please describe.
- How easy was it to navigate and interact with the VR environment?
- Did you encounter any technical issues during the VR experience? If yes, please describe.
- Did you experience any motion sickness or discomfort related to the VR technology?
- How informative did you find the content of the VR experience?
- How much do you feel your knowledge about hypomania has improved after the VR experience?
- Do you have any suggestions for improving the VR experience? If yes, please describe.
- Would you recommend this VR experience to others interested in learning about hypomania?
- How has this VR experience impacted your perception of individuals with hypomania?
- Is there anything else you would like to share about your experience?

XVIII. RESULTS

A. The results of the first questionnaire are summarized as follows:

The total of the participants consisted of individuals aged 22-42 years, with various gender identities, who have completed their secondary education, with some individuals holding a degree and a postgraduate title. Regarding their previous experience, 57,1% have participated in psychotherapy sessions with mental health specialists, with only 42,9% reporting that they had no such experience, and of the total, 78,6% had not received a formal diagnosis of a mental illness. When asked how familiar they were with the term serious gaming on a scale of 1 to 5, with 1 being the lowest, 14,3% answered that they were not very familiar, while 21,4% responded that they were familiar, and the remaining 28,6% answered as very familiar. It is important to note that prior to interacting with the VR experience, participants marked their mood and anxiety levels on a scale of 1 to 10 (where 1 is the lowest and 10 is the highest), with the results emerging as followed:

Mood level:

1	2	3	4	5	6	7	8	9	10
0%	7.1%	7.1%	21,4%	7.1%	28,6%	21,4%	0%	7.1%	0%

Table 11:Participant's mood levels

Anxiety level:

1	2	3	4	5	6	7	8	9	10
7.1%	14.4%	7.1%	14.3%	7.1%	7.1%	14.3%	14.3%	14.3%	0%

Table 12:Participants' anxiety levels

Regarding their knowledge of the bipolar spectrum, participants were asked to rate their familiarity on a scale of 1 to 5, with 1 being the lowest and 5 being the highest, 28,6% reported that they were not familiar, 21,4% said they were adequately familiar, 28,6% felt they were familiar, and the remaining 21,4% considered themselves very familiar. Considering the term hypomania, 21,4% were not very familiar, 14,5% were adequately familiar, 28,6% were familiar, and 16.7% were very familiar. When asked how comfortable they would feel around a person during a hypomanic episode, 14,3% indicated they would feel uncomfortable, 42,9% would feel neutral, 35,7% would feel comfortable, and 7,1% would feel very comfortable. Participants were also asked about their thoughts and previous experiences with using a VR headset. Specifically, 28,6% reported that they had never used a VR headset before, 35,7% had some knowledge, 21,4% had moderate knowledge, and the remaining 14.3% felt very familiar with the equipment. Also, when asked how comfortable they would feel using VR technology, their responses on a scale of 1 to 10, with 10 being the highest, were as follows:

Familiarity of VR Headset Use:

1	2	3	4	5	6	7	8	9	10
14.3%	7.1%	0%	14.3%	14.3%	14.3%	14.3%	7.1%	7.1%	7.1%

Table 13:Familiarity of VR Headset Use

When asked about their possible concerns regarding the use of the equipment and their participation in the experience, some participants mentioned dizziness as a factor. There is also the possibility of recording health problems that could affect the process, though nothing significant was mentioned. Finally, participants were asked to indicate their expectations from the experience and where they hoped to gain more information about the concept of sub-mania. In general, the responses reflected an expectation of gaining a better understanding of the symptoms of the disease, as well as the experience of the individual affected by it. Finally, all subjects consented in taking part in this testing and using their data.

Here, all charts occurring from the data above are depicted:

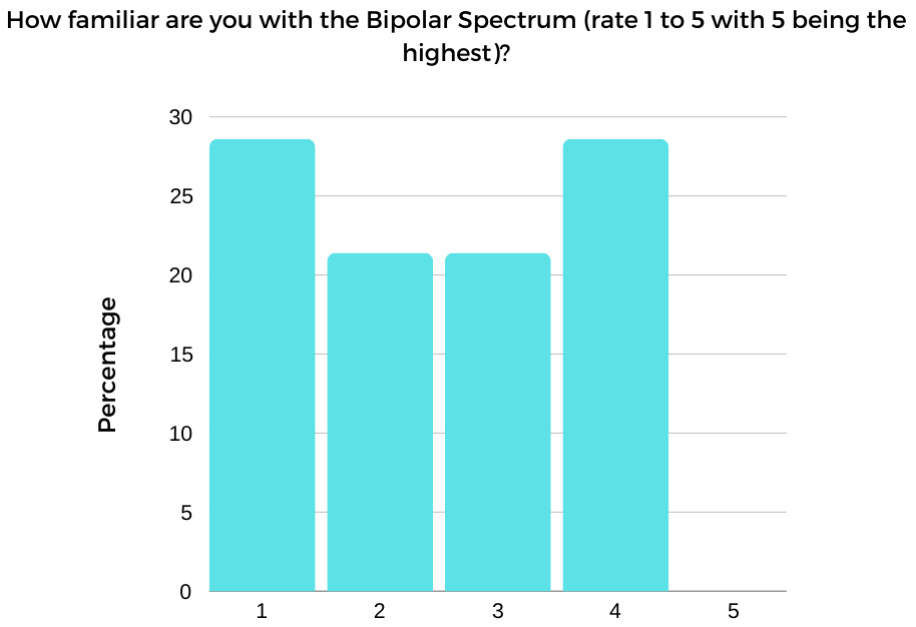

Figure 20: Chart 1

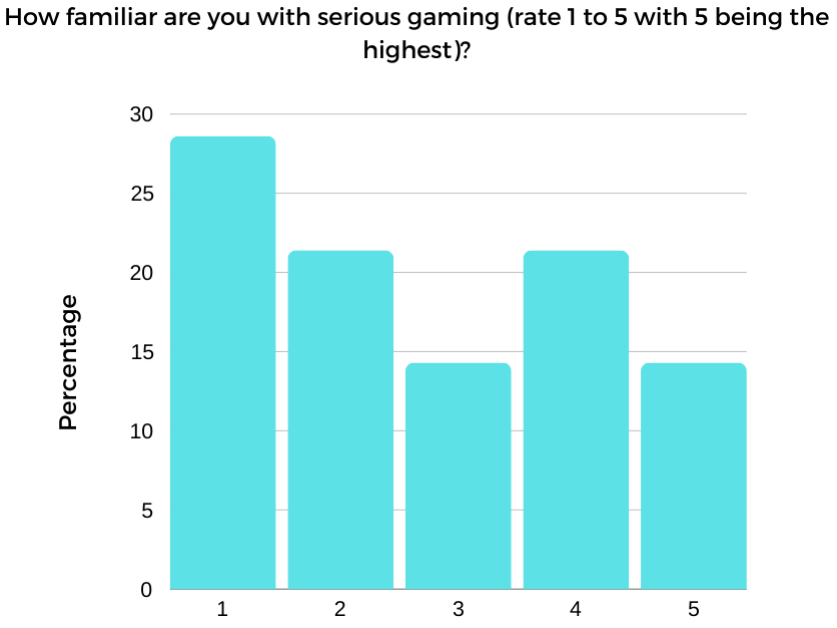

Figure 21: Chart 2

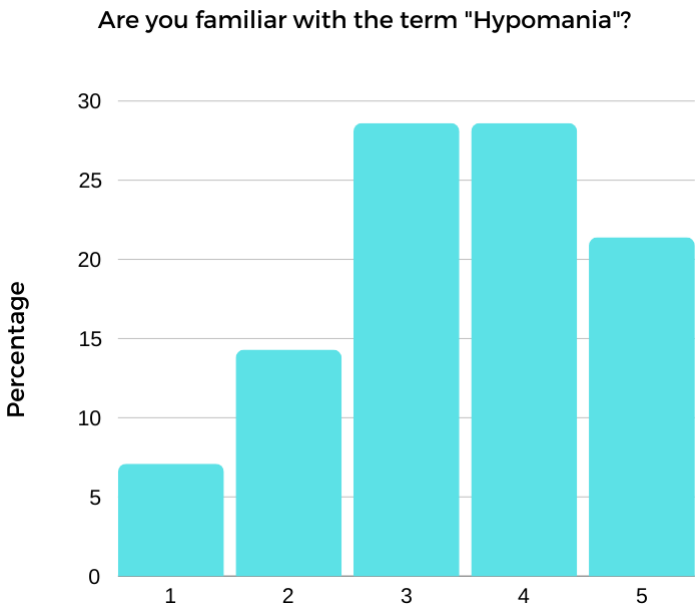

Figure 22:Chart 3

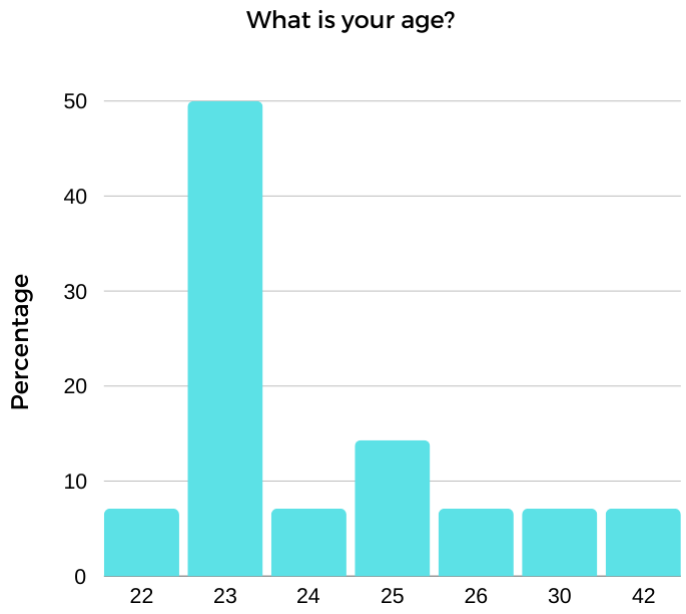

Figure 23:Chart 4

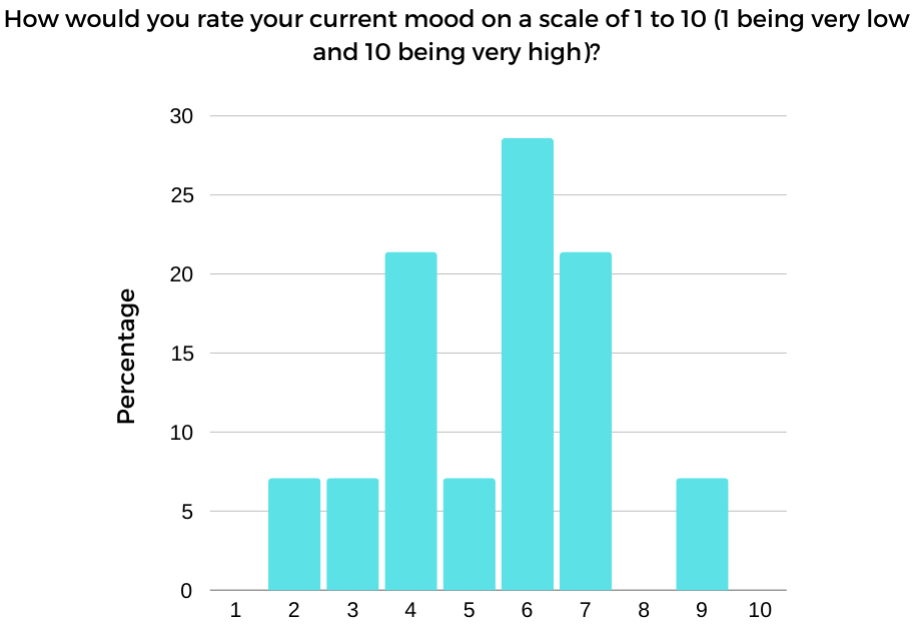

Figure 24:Chart 5

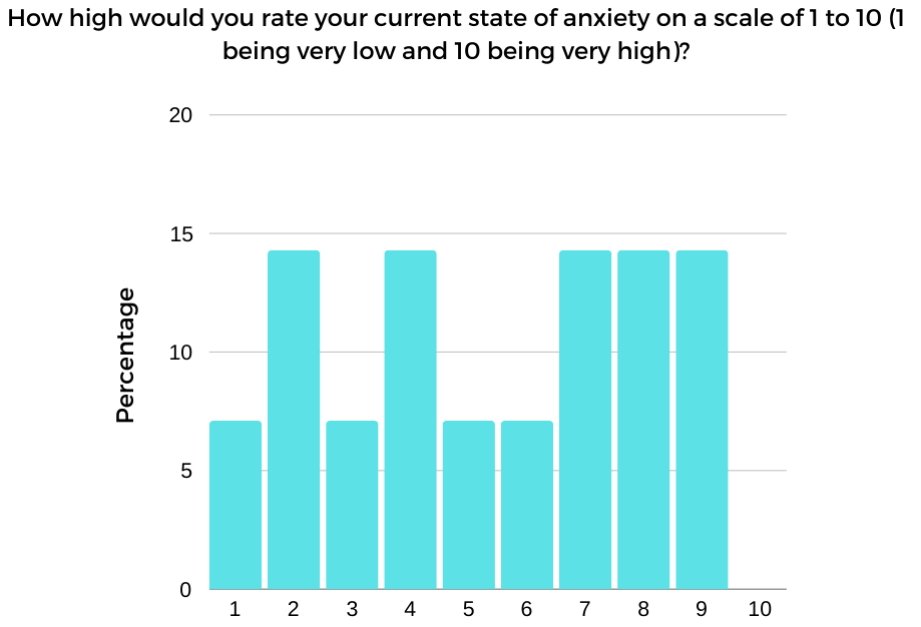

Figure 25:Chart 6

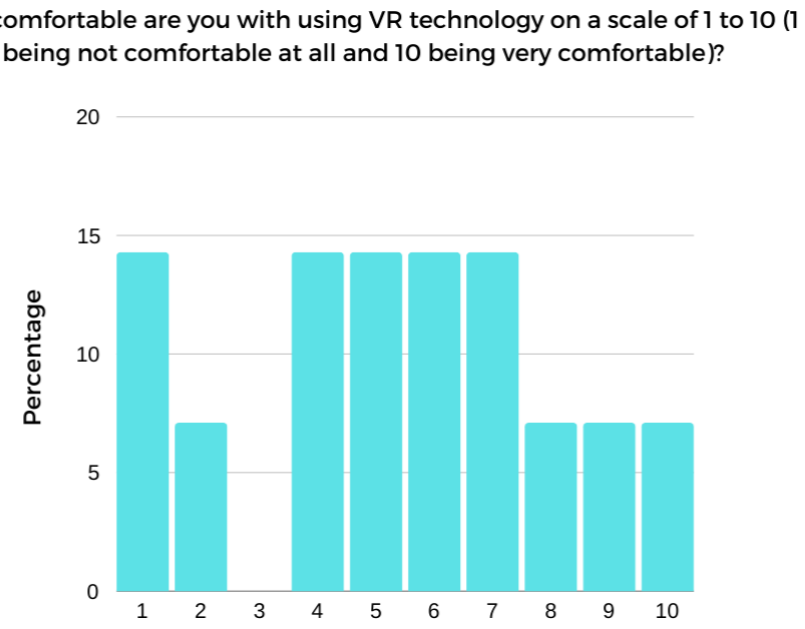

Figure 26:Chart 7

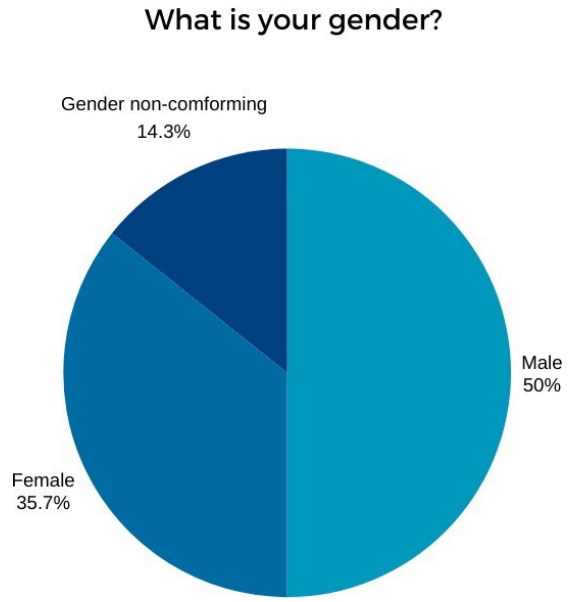

Figure 27:Chart 8

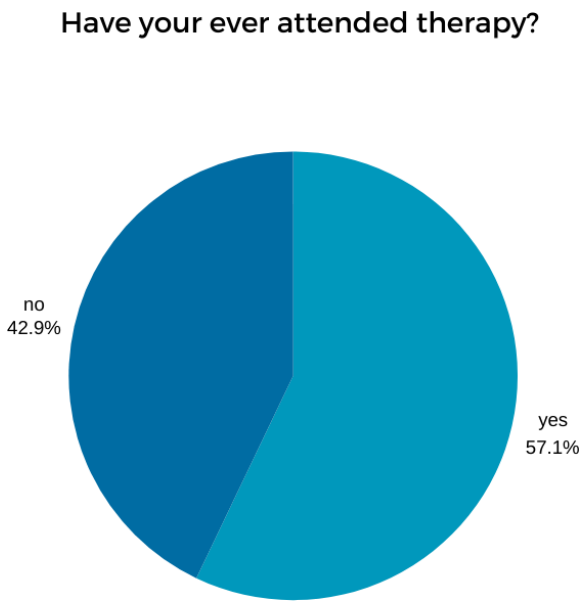

Figure 28:Chart 9

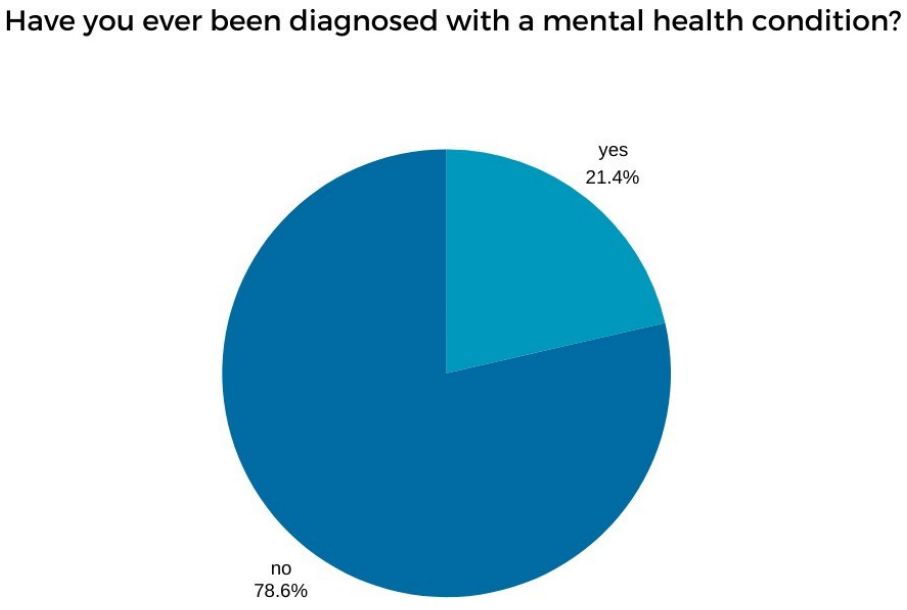

Figure 29:Chart 10

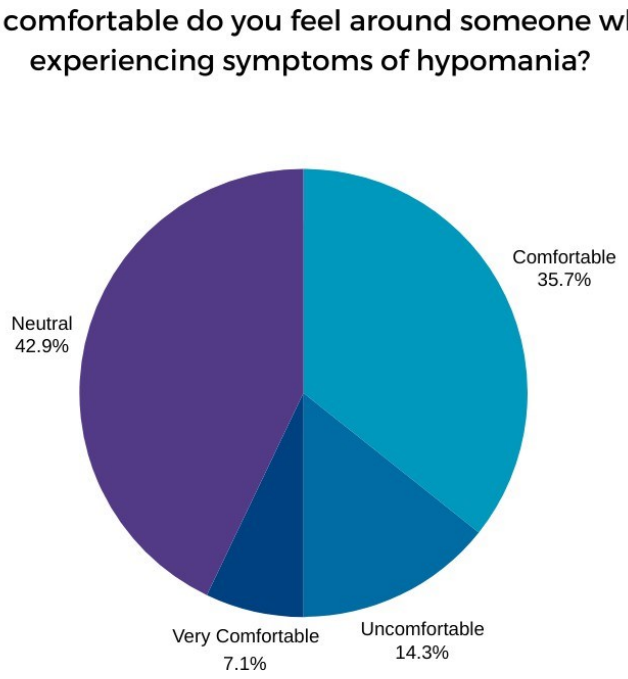

Figure 30:Chart 11

B. The results of the <u>second questionnaire</u> are summarized as follows:

The participants were initially asked to rate the experience on a scale of very poor, poor, average, good, and excellent. Of the participants, 75% rated the experience as good, while the remaining 25% rated it as excellent. Additionally, 66,7% describing it as very engaging with the remaining 33,3% found the experience extremely engaging. Participants reported that their understanding of hypomania and its symptoms greatly improved after the trial, leading to increased empathy towards individuals battling the disorder. When asked to what extent they considered the overall experience informative, the majority of participants said they considered it extremely educational and would recommend it to others to raise awareness and understanding of the Bipolar Spectrum. The VR experience evoked a wide range of reactions and insights from the participants, disclosing both the engaging power of Virtual Reality technology and its potential to raise awareness and create empathy for individuals dealing with hypomania. Many participants described initial feelings of unease and disorientation as they adjusted to the virtual reality environment, with some mentioning the intensity of the visuals and sounds as contributing factors. Despite these challenges, most participants found the experience highly engaging and were able to acclimate easily, with many expressing a desire to continue exploring the virtual world created. The experience was particularly impactful in altering perceptions of hypomania, with several participants gaining a more visceral understanding of the altered sensory phenomena occurring and the chaotic nature of a hypomanic episode. They described the simulation as a vivid portrayal of the overwhelming stimuli that individuals with hypomania might face, with the rapid changes in colors, images, and sounds creating a sense of disturbance and difficulty in concentration. As a result, a deeper understanding and empathetic feelings towards those suffering with the disorder was mentioned, as participants could better appreciate the challenges faced even in simple, everyday tasks. Also, it was noted by many how the VR experience brought them closer to the emotional and sensory reality of hypomania, offering more than just a narrative but also a tangible sense of what it might be like. The

experience also sparked reflections on the strength and resilience of those living with hypomania, with one participant expressing admiration for their ability to find balance amidst such tension. Additionally, the VR simulation initiated reflections about the broader implications of such technology in breaking down stereotypes and providing more profound insights into mental health conditions. Participants were also impressed by the detail and personal touch in the virtual environment, which added to the overall impact of the experience. The combination of emotional connection, increased understanding, and appreciation for the technological execution made this VR experience a meaningful and transformative event for the participants. Regarding the technological aspects of the VR Experience, 50% reported feeling comfortable during the trial, while 33.3% felt uncomfortable and 16.7% felt very comfortable. Overall, 58.3% characterized the program as neutral in terms of handling, 8.3% found it easy, 8.3% considered is difficult, while 8.3% found it very easy. Finally, 50% reported no dizziness, 25% experienced a slight feeling of dizziness, 8.3% felt a moderate level of dizziness, and the remaining 16.7% reported a stronger feeling of dizziness.

Here, all charts occurring from the data above are depicted:

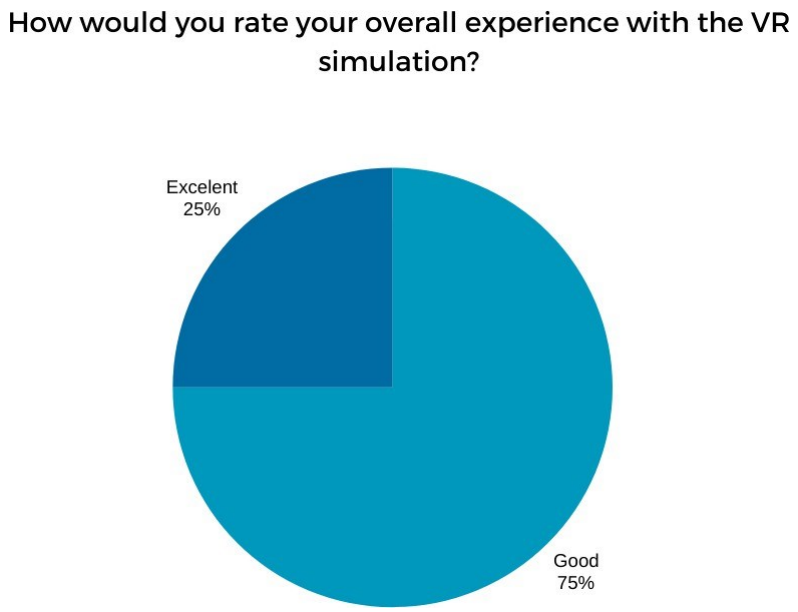

Figure 31:Chart 12

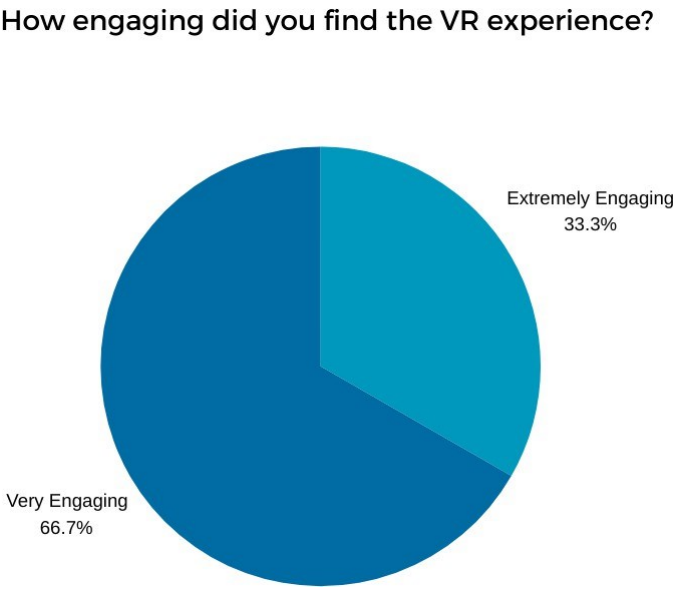

Figure 32:Chart 13

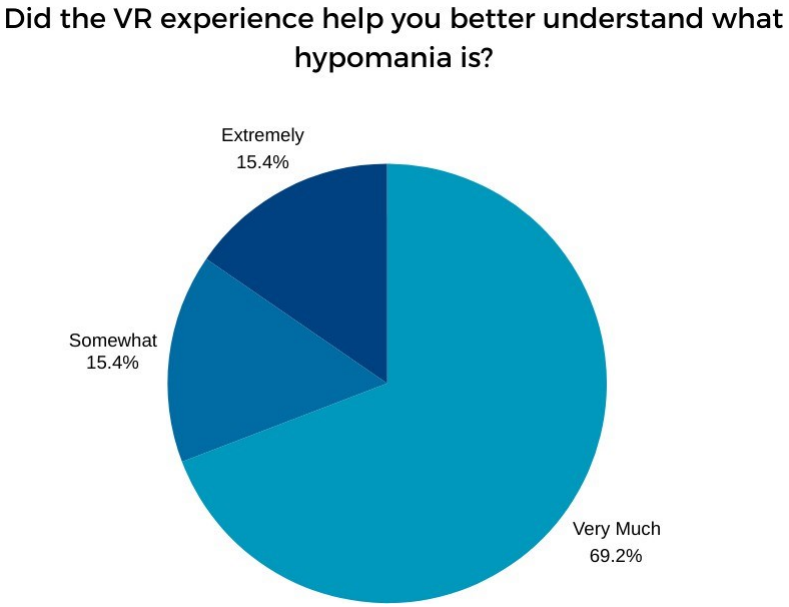

Figure 33:Chart 14

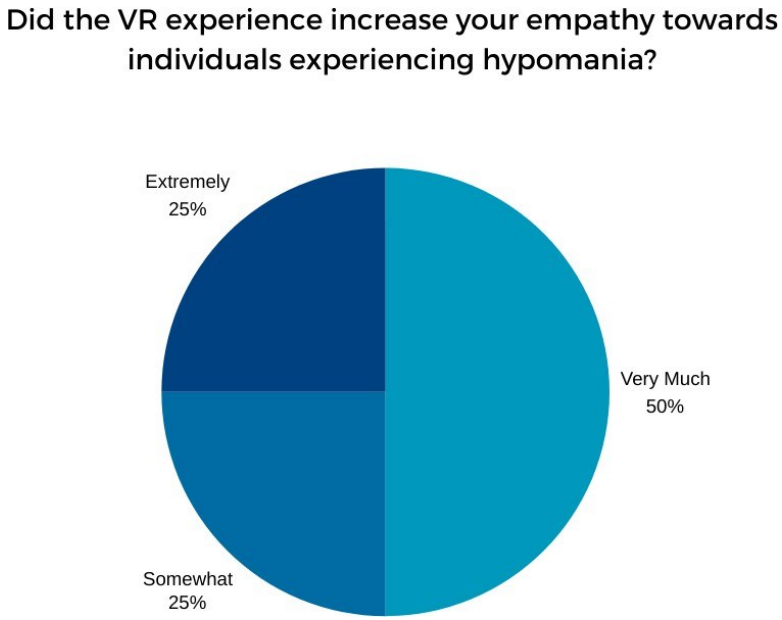

Figure 34:Chart 15

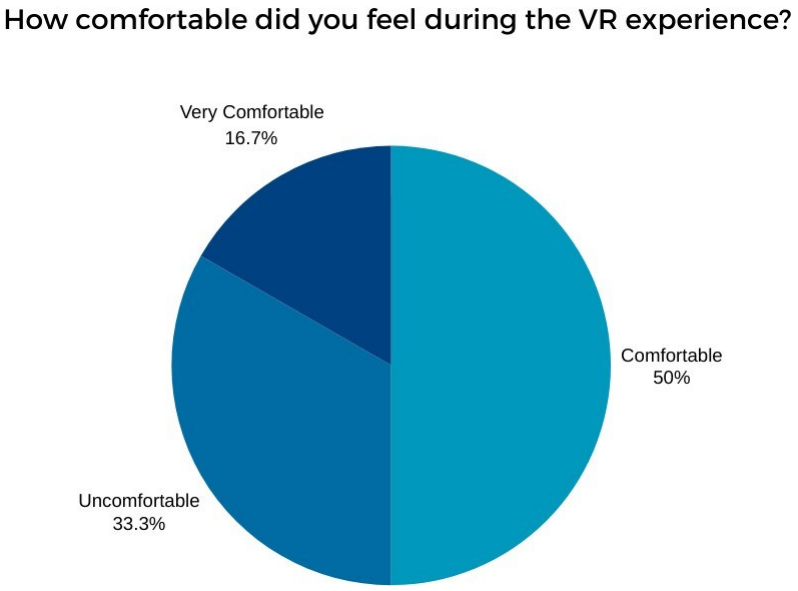

Figure 35:Chart 16

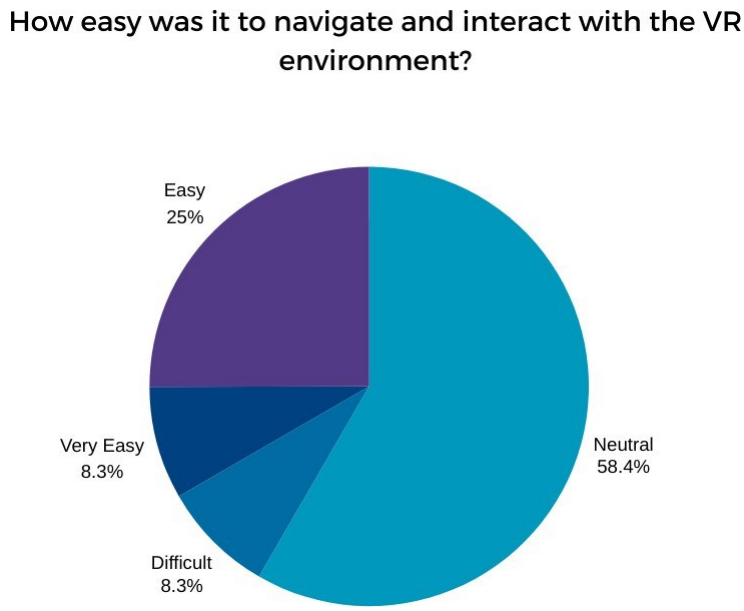

Figure 36:Chart 17

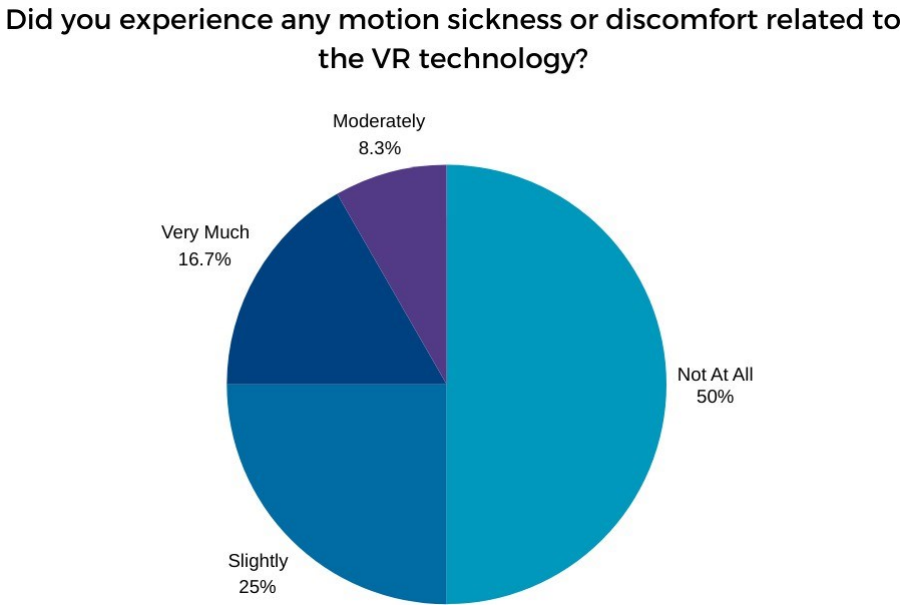

Figure 37:Chart 18

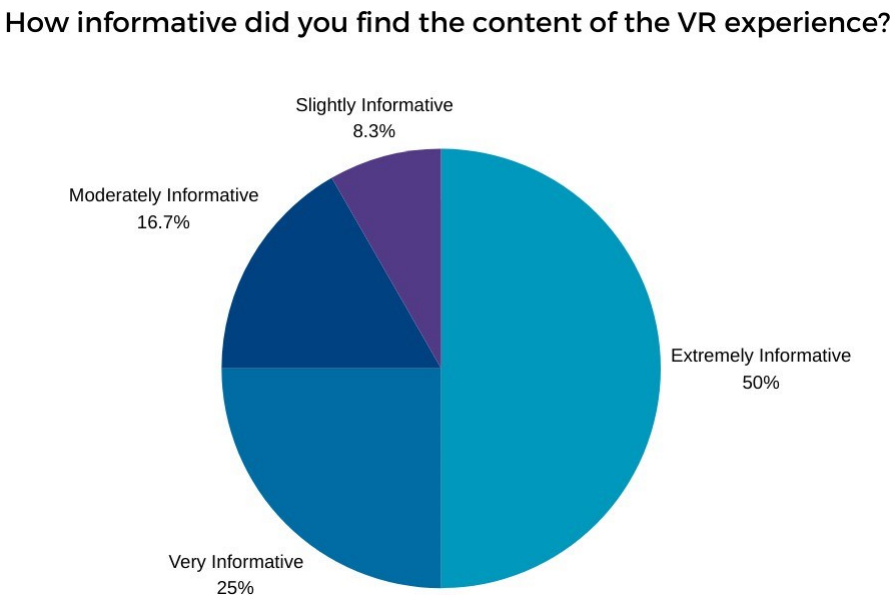

Figure 38:Chart 19

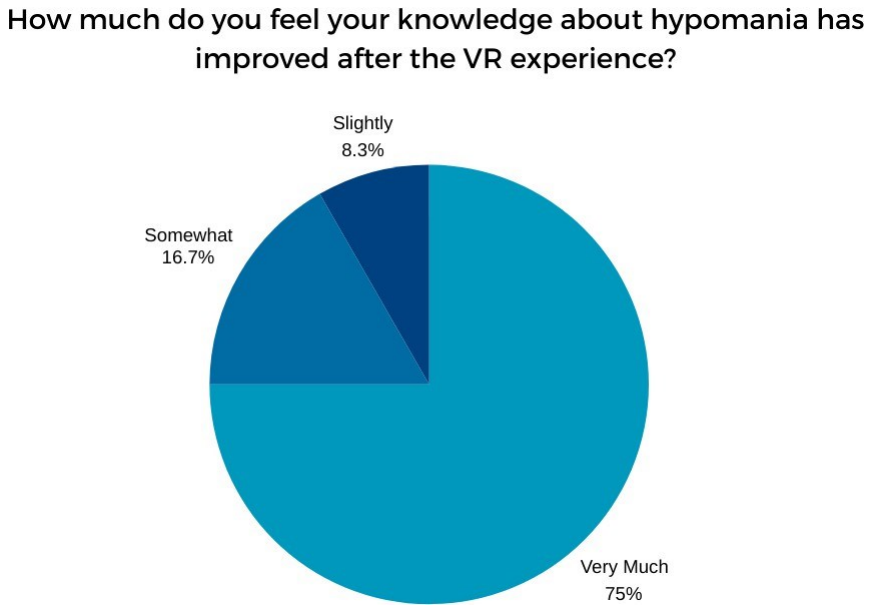

Figure 39: Chart 20

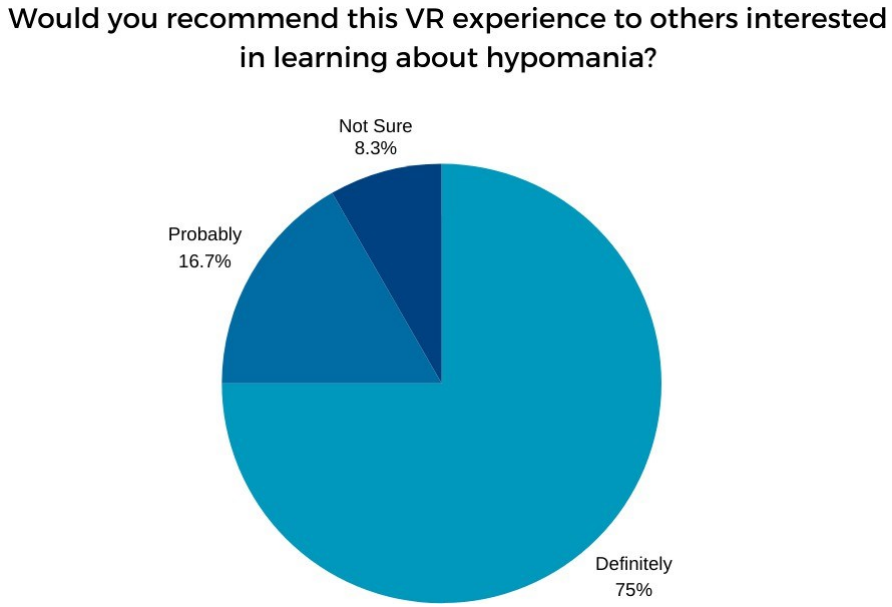

Figure 40: Chart 21

XIX. ACKNOWLEGMENTS

In this particular chapter I would like to pay tribute to the authors and researchers whose work, though not included as sources or directly used in the writing of this thesis, have significally helped my research and process of better understanding hypomania and serious gaming. In particular, I found the articles by BJPsych Bulletin, Carmona Torres, Adolfo J. Cangas Diaz, Alvaro I. Langer Herrera, Wang Sa, Mao Zhengli, Zeng Changhai, Gong Huili, Li Shanshan, Chen Beibei, Arthur H. Crisp, Michael G. Gelder, Susannah Rix, Howard I. Meltzer and Olwen J. Rowlands.

XX. CONCLUSION

Summarizing the aforementioned, the Bipolar Disorder Spectrum was first noticed during the 19th century by Parisian alienist Jules Bailarger, due to its distinct symptomatology of combining both euphoric and depressive phases and in the late 1970s was successfully divided into 4 categories based on the symptoms exhibited: 1. Bipolar Disorder Type 1, 2. Bipolar Disorder Type II, 3. Cyclothymia and 4. Bipolar Disorder Not Otherwise Specified. The main symptoms characterizing the disorder are Major Depressive Episodes, Mania, and Hypomania, depending on the severity and duration. Since the symptomatology closely resembles Major Depressive Disorder (MDD) and most patients lack insight to their disorder, the rate of misdiagnosis is quite elevated, reaching around 60% of all cases. Bipolar Disorders can be attributed to several causes, with the most prominent being genetic factors, chemical imbalances concerning the patients' neurotransmitters, and lastly, triggers rooted in early childhood, such as child sexual abuse. It should also be noted that in some instances, substance abuse, such as alcohol and opioids, can lead to Manic/Hypomanic episodes. Even though BP is considered incurable, research has displayed promising results in combating the disorder, particularly when patients adhere to their medication, with multiple different agents being used, and psychotherapy attendance, even when in remission. An aspect of the Bipolar Spectrum that still lacks research is the alterations patients experience in their sensory perception, as their vision deteriorates, auditory hallucinations and errors commonly take place and their overall consciousness is affected. Considering that fact, individuals with BP should be approached as a high-risk group, not only because of their psychological turmoil but

also due to the physical difficulties they endure. A modern-day approach that has shown promising results is the use of serious gaming, a gaming experience used to educate while simultaneously entertaining the player. Up until recently, only two serious games considering the Bipolar Spectrum have been developed: 1. BIPOLIFE and 2. Stigma-Stop. Both games focus strictly on the psychological aspect of the disorder, surrounding a character that has to make prudent choices to maintain their mental stability. This is the exact point where we would like to bring forward our proposition; a serious game that prioritizes the physical symptoms an individual with BP experiences and highlights the daily danger they face. The serious game is developed using Unity3D and will be available as a VR experience. By including the physical aspect of the Bipolar Spectrum, as well as the difficulties it can cause, the general public will be informed, while we concurrently raise awareness around the spectrum, in an attempt to eliminate the generational trauma that has led to social exclusion and discrimination.

XXI. REFERENCES

[1] L. Tondo, G. H. Vázquez, and R. J. Baldessarini, "Depression and Mania in Bipolar Disorder," *Curr Neuropharmacol*, vol. 15, no. 3, p. 353, Jun. 2017, doi: 10.2174/1570159X14666160606210811.

[2] Y. C. Shen, "Treatment of acute bipolar depression," *Tzu-Chi Medical Journal*, vol. 30, no. 3, p. 141, Jul. 2018, doi: 10.4103/TCMJ.TCMJ_71_18.

[3] M. J. Weintraub, M. M. Van De Loo, M. J. Gitlin, and D. J. Miklowitz, "Self-Harm, Affective Traits, and Psychosocial Functioning in Adults with Depressive and Bipolar Disorders," *J Nerv Ment Dis*, vol. 205, no. 11, p. 896, Nov. 2017, doi: 10.1097/NMD.0000000000000744.

[4] U. Mccormick, B. Murray, and B. Mcnew, "Diagnosis and treatment of patients with bipolar disorder: A review for advanced practice nurses," *J Am Assoc Nurse Pract*, vol. 27, no. 9, p. 530, Sep. 2015, doi: 10.1002/2327-6924.12275.

[5] ΣΤΥΛΙΑΝΝΗ ΜΟΥΤΣΑΝΑ, "ΔΙΠΟΛΙΚΗ ΔΙΑΤΑΡΑΧΗ ΚΑΙ ΝΟΣΗΛΕΥΤΙΚΕΣ ΔΙΕΡΓΑΣΙΕΣ," 2016.

[6] M. L. Phillips and D. J. Kupfer, "Bipolar disorder diagnosis: challenges and future directions," *Lancet*, vol. 381, no. 9878, p. 1663, May 2013, doi: 10.1016/S0140-6736(13)60989-7.

[7] J. Angst *et al.*, "Hypomania: a transcultural perspective," *World Psychiatry*, vol. 9, no. 1, p. 41, 2010, doi: 10.1002/J.2051-5545.2010.TB00268.X.

[8] M. Bauer and A. Pfennig, "Epidemiology of Bipolar Disorders," *Epilepsia*, vol. 46, no. SUPPL. 4, pp. 8–13, 2005, doi: 10.1111/J.1528-1167.2005.463003.X.

[9] G. Parker, A. Paterson, M. Romano, and R. Graham, "Altered Sensory Phenomena Experienced in Bipolar Disorder," *https://doi.org/10.1176/appi.ajp.2017.16121379*, vol. 174, no. 12, pp. 1146–1150, Dec. 2017, doi: 10.1176/APPI.AJP.2017.16121379.

[10] Y. Takaesu, "Circadian rhythm in bipolar disorder: A review of the literature," *Psychiatry Clin Neurosci*, vol. 72, no. 9, pp. 673–682, Sep. 2018, doi: 10.1111/PCN.12688.

[11] J. Orzeł-Gryglewska, "CONSEQUENCES OF SLEEP DEPRIVATION," *Int J Occup Med Environ Health*, vol. 23, no. 1, pp. 95–114, 2010, doi: 10.2478/v10001-010-0004-9.

[12] M. Latifian, K. Abdi, G. Raheb, S. M. S. Islam, and R. Alikhani, "Stigma in people living with bipolar disorder and their families: a systematic review," *Int J Bipolar Disord*, vol. 11, no. 1, p. 9, Dec. 2023, doi: 10.1186/S40345-023-00290-Y.

[13] Beth Hughes, Platon Issaias, and Yannis Drakoulidis, "ISLANDS OF EXILE: THE CASE OF LEROS," 2018.

[14] C. H. Au, C. S. M. Wong, C. W. Law, M. C. Wong, and K. F. Chung, "Self-stigma, stigma coping and functioning in remitted bipolar disorder," *Gen Hosp Psychiatry*, vol. 57, pp. 7–12, Mar. 2019, doi: 10.1016/J.GENHOSPPSYCH.2018.12.007.

[15] A. Nelson, "Ups and Downs: Social Media Advocacy of Bipolar Disorder on World Mental Health Day," *Front Commun (Lausanne)*, vol. 4, p. 446918, May 2019, doi: 10.3389/FCOMM.2019.00024/BIBTEX.

[16] L. Stege, G. Van Lankveld, and P. Spronck, "Serious Games in Education." [Online]. Available: www.iacss.org

[17] S. de Matos Lima and P. Otero, "Serious games are more than just games," *Arch Argent Pediatr*, vol. 122, no. 6, 2024, doi: 10.5546/aap.2023-10218.eng.

[18] F. Fernández-Aranda *et al.*, "Video games as a complementary therapy tool in mental disorders: PlayMancer, a European multicentre study," *Journal of Mental Health*, vol. 21, no. 4, pp. 364–374, Aug. 2012, doi: 10.3109/09638237.2012.664302.

[19] A. J. Cangas, I. Sánchez-Lozano, J. M. Aguilar-Parra, and R. Trigueros, "Combination of a Serious Game Application and Direct Contact with Mental Health Patients," *Int J Ment Health Addict*, vol. 20, no. 6, pp. 3274–3284, Dec. 2022, doi: 10.1007/s11469-022-00752-x.

[20] A. Dewhirst, R. Laugharne, and R. Shankar, "Therapeutic use of serious games in mental health: scoping review," *BJPsych Open*, vol. 8, no. 2, Mar. 2022, doi: 10.1192/bjo.2022.4.

[21] A. J. Cangas *et al.*, "Stigma-Stop: A Serious Game against the Stigma toward Mental Health in Educational Settings," *Front Psychol*, vol. 8, no. AUG, p. 1385, Aug. 2017, doi: 10.3389/FPSYG.2017.01385.

[22] K. D. Maurin *et al.*, "Use of a serious game to strengthen medication adherence in euthymic patients with bipolar disorder following a psychoeducational programme: A randomized controlled trial," *J Affect Disord*, vol. 262, pp. 182–188, Feb. 2020, doi: 10.1016/J.JAD.2019.10.008.

[23] "Bipolar Eyes: What It Is and More I Psych Central." Accessed: Feb. 22, 2024. [Online]. Available: https://psychcentral.com/bipolar/bipolar-eyes#eyes-and-mood-episodes

[24] K. Zajac, M. K. Ginley, R. Chang, and N. M. Petry, "Treatments for Internet Gaming Disorder and Internet Addiction: A Systematic Review HHS Public Access", doi: 10.1037/adb0000315.

[25] "Set up Meta Quest Link | Meta Store." Accessed: Jul. 21, 2024. [Online]. Available: https://www.meta.com/help/quest/articles/headsets-and-accessories/oculus-link/set-up-link/

[26] "Unity - Manual: VR overview." Accessed: Jul. 21, 2024. [Online]. Available: https://docs.unity3d.com/540/Documentation/Manual/VROverview.html

[27] "Unity - Manual: Oculus." Accessed: Jul. 21, 2024. [Online]. Available: https://docs.unity3d.com/550/Documentation/Manual/VRDevices-Oculus.html