



**Francisco
Quadrado Lopes**

**Exploração De Realidade Virtual Para Suporte Ao
Laboratório De Psicologia Experimental**

**Exploring Virtual Reality To Support The
Experimental Psychology Lab**



Universidade de Aveiro
2021

**Francisco
Quadrado Lopes**

**Exploração De Realidade Virtual Para Suporte Ao
Laboratório De Psicologia Experimental**

**Exploring Virtual Reality To Support The
Experimental Psychology Lab**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor Samuel de Sousa Silva, Investigador do Instituto de Engenharia Eletrónica e Informática de Aveiro, e da Professora Doutora Sandra Cristina de Oliveira Soares, Professora Auxiliar do Departamento de Educação e Psicologia da Universidade de Aveiro.

Dedico este trabalho à minha família pelo incansável apoio durante todo meu percurso académico.

o júri / the jury

presidente / president

Prof. Doutor Joaquim Arnaldo Carvalho Martins

Professor Catedrático da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor José Casimiro Nunes Pereira

Professor Adjunto do Instituto Politécnico de Tomar

Doutor Samuel de Sousa Silva

Investigador do Instituto de Engenharia Eletrónica e Informática de Aveiro

**agradecimentos /
acknowledgements**

Em primeiro lugar, quero deixar um agradecimento especial ao meu orientador Doutor Samuel Silva pelo acompanhamento exemplar durante toda a realização desta dissertação, que sempre me encaminhou pelo rumo certo. Quero ainda agradecer à Prof. Sandra Soares pela partilha dos seus conhecimentos da Psicologia que me proporcionaram um entendimento sucinto sobre os conceitos psicológicos abordados nesta dissertação. Gostaria também de agradecer ao Prof. Paulo Dias e ao Doutor Bernardo Marques por me disponibilizarem o equipamento de Realidade Virtual necessário para a realização desta dissertação.

Palavras Chave

realidade virtual, unity, continuous flash suppression, processamento de ameaça, psicologia experimental.

Resumo

O processamento de ameaça é uma área da psicologia que estuda como é processada a resposta emocional a estímulos de medo sob inconsciência. Os investigadores estudam esta área apresentando estímulos ameaçadores a participantes em condições de inconsciência visual, usando paradigmas de supressão visual, em laboratórios de psicologia experimental. Estes paradigmas exigem que duas imagens diferentes sejam mostradas a cada olho para criar o estado de inconsciência. Um desses paradigmas é a "breaking-Continuous Flash Suppression" (bCFS). Os métodos tradicionais usam equipamento especial para exibir uma imagem diferente para cada olho. No entanto, estes têm vários factores insatisfatórios, como o custo ou a eficácia.

A Realidade Virtual (RV) é uma tecnologia cujo uso tem vindo a aumentar em estudos recentes e tem-se mostrado promissora. Nesse sentido, em colaboração com o Departamento de Educação e Psicologia, propõe-se avaliar se a tecnologia RV será uma alternativa adequada para substituir os métodos tradicionais actuais através de sessões experimentais de recolha de dados. É desenvolvida uma aplicação para realizar experiências de processamento de ameaça com o paradigma bCFS que permite a recolha de dados, suportando RV e os métodos tradicionais atuais, satisfazendo uma lista de requisitos criada para orientar o seu desenvolvimento.

A aplicação desenvolvida é capaz de realizar as experiências necessárias e fazer a recolha dos dados resultantes para a análise, suportando RV e os métodos tradicionais. A configuração da experiência é feita através de um ficheiro de configuração que permite definir vários parâmetros que produzem diferentes condições na experiência. O sistema da aplicação é capaz de exibir estímulos com precisão, incluindo outras funcionalidades, como a calibração de imagem e recursos de resolução de problemas em tempo de execução.

A aplicação de software desenvolvida permite, no seu estado actual, a recolha de dados experimentais para validar a Realidade Virtual como uma tecnologia adequada para o laboratório de psicologia experimental.

Keywords

virtual reality, unity, continuous flash suppression, threat processing, experimental psychology.

Abstract

Threat processing is a psychology field that studies how the emotional response to fear stimuli is processed under unawareness. Researchers study this area by presenting to participants threat inducing stimuli in visual unawareness conditions, using visual suppression paradigms in the experimental psychology labs. These paradigms require that two different images are shown to each eye to create the unaware state. One such paradigm is breaking-Continuous Flash Suppression (bCFS). The traditional methods use special equipment to display a different image to each eye. However, these have several unappealing factors, such as expensiveness or effectiveness.

Virtual Reality (VR) is a technology that has seen its use increase in recent research studies, showing promising results. In this regard, in collaboration with the Department of Education and Psychology, it is proposed to evaluate VR technology as a suitable alternative to current traditional methods through experimental data collection sessions. An application system is developed to perform threat processing experiments implementing the bCFS paradigm that allows the data collection, supporting VR, and the current traditional methods, satisfying a list of requirements created to guide its development.

The application developed is capable of performing the experiments required and collecting the resulting data analysis, supporting VR and the traditional methods. The experiment setup is achieved via a configuration file that allows setting various parameters to produce different experiment conditions. The system is capable of displaying stimuli accurately, containing other functionalities such as image calibration, and runtime troubleshooting features.

The software application developed allows, in its current state, the experimental data collection to validate Virtual Reality as a suitable technology for the experimental psychology lab.

Contents

Contents	i
List of Figures	v
List of Tables	vii
Glossary	ix
1 Introduction	1
1.1 Context	1
1.2 Background to the Research	1
1.3 Challenges	2
1.4 Objectives	3
1.5 Document Structure	4
2 Background	5
2.1 Virtual Reality Technology	5
2.1.1 VR Accessibility and Usability	6
2.1.2 Advantages and Disadvantages of VR	7
2.1.3 Developing VR Experiences	8
2.2 Virtual Reality Research in Experimental Psychology	9
2.3 Threat Processing Research	10
2.4 Discussion	11
2.5 Chapter Conclusions	12
3 Experiment Protocol	13
3.1 Experiment Context	13
3.2 Protocol	13
3.2.1 Methodology	13
3.2.2 Stimuli	14
3.2.3 Experiment Procedure	16

3.3	Computational Requirements	17
3.4	Equipment	18
3.4.1	HTC Vive	18
3.4.2	Desktop Monitor	20
3.5	Chapter Conclusions	23
4	Experiment Software Platform	25
4.1	Conceptual Software Architecture	25
4.1.1	Configuration Parser	26
4.1.2	Pre-Task Manager	27
4.1.3	Experiment Engine	28
4.2	Application Development With Unity	30
4.2.1	Unity's Development Framework	30
4.2.2	Experiment Engine Scripts Hierarchy	31
4.2.3	Stimuli Presentation	33
4.2.4	Scene Objects Scaling Methodology	35
4.2.5	Data Management	38
4.3	Configuration File	38
4.3.1	XML File Structure & Properties	38
4.3.2	Configuration Parameters	39
4.4	Application Usage & User Interface	43
4.5	Runtime Troubleshooting Features	46
4.6	Chapter Conclusions	48
5	Conclusions & Future Work	49
5.1	Conclusions	49
5.2	Future Work	50
A	Experiment Object Presentation Notes	53
A.1	Virtual Reality Head Mounted Display Estimations	53
A.1.1	VR HMD Screen Dimensions	53
A.1.2	VR HMD Eye-to-Screen Distance	54
A.2	VR Object Presentation Observations	56
A.3	Unity Object Presentation Process	56
B	Configuration File Usage Notes	59
B.1	Configuration File Manual	59
B.2	Configuration File Compatibility Discussion	67
B.3	Future Improvements	67

C	Application Usage Notes	69
C.1	Application Input Mapping	69
C.2	System Requirements	73
C.3	Future Improvements	73
	References	75

List of Figures

1.1	Common threatening species to humans.	2
2.1	"The Sword of Damocles", Ivan Sutherland, 1968	6
2.2	A typical modern VR setup.	7
2.3	Screen Door Effect and Chromatic Aberration Examples.	8
2.4	Unity Logo.	8
3.1	Stimuli images selected for the experiments.	14
3.2	Frame dimensions and stimuli position presentation for the experiment.	15
3.3	Frame RGB channel filter processing for the Red-Blue anaglyph glasses.	16
3.4	Experimental trial stimuli presentation procedure.	17
3.5	HTC Vive system.	19
3.6	Desktop monitor selected for the experiments.	21
3.7	Red-Blue Anaglyph 3D Glasses.	21
3.8	Representation of a mirror apparatus as a four-mirror stereoscope.	22
3.9	Head-chin rest equipment example.	23
4.1	Diagram of the application's conceptual architecture.	26
4.2	Configuration Parser component diagram	26
4.3	Pre-Task Manager component diagram.	27
4.4	Experiment Engine component diagram.	29
4.5	Unity GameObject representing a cube.	31
4.6	Unity Sprite GameObject inspector window.	31
4.7	Hierarchy of Experiment Engine scripts of all supported platforms.	32
4.8	Desktop platform experiment frame example.	33
4.9	Binocular (VR and desktop with mirror apparatus) platform experiment frame example.	34
4.10	Example demonstration of overlapping images with and without Additive Blending.	34
4.11	Right Triangle of visual angle.	35
4.12	Two right triangles complement the length of an object.	36
4.13	Demonstration of the image deformation of a resized stimulus in both dimensions.	37

4.14	Participant information data page example.	43
4.15	Pre-Calibration page example.	44
4.16	Calibration phase in the Desktop with mirrors platform.	45
4.17	Trial of the "bCFS Fovea Periphery" experiment in the Desktop with mirrors platform.	46
4.18	Error message window.	46
4.19	Warning message window.	47
4.20	Information message window.	47
A.1	Screen dimension representation of the VR HMD	54
A.2	Human eye FOV representation.	55
A.3	Unity Canvas GameObject inspector window.	57
A.4	Unity Canvas and Camera of the desktop platform in 3D perspective view.	58
C.1	Vive controller input diagram.	69

List of Tables

3.1	Experiment session methodology.	14
3.2	HTC Vive HMD specifications.	19
3.3	Desktop monitor specifications.	20
B.1	Implemented experimental paradigm parameter values.	60
B.2	Implemented display platform values.	61
B.3	Implemented experiment's color filtering values.	63
B.4	Implemented size/distance unit values.	64
B.5	Implemented reference dimension values.	64
B.6	Implemented sort type values.	65
B.7	Implemented page name values.	66
C.1	Single frame calibration phase (desktop platform using anaglyph glasses) input action mapping.	70
C.2	Dual-frame calibration phase (VR and desktop using mirror apparatus platforms) input action mapping.	71
C.3	Experiment (and training) phase input action mapping.	72

Glossary

2D	Two-Dimensional
3D	Three-Dimensional
AMOLED	Active-Matrix Organic Light-Emitting Diode
API	Application Programming Interface
AR	Augmented Reality
bCFS	breaking-Continuous Flash Suppression
CFS	Continuous Flash Suppression
CSV	Comma-Separated Values
FOV	Field of View
HMD	Head Mounted Display
IPD	Interpupillary Distance
LED	Light-Emitting Diode
RGB	Red, Green, and Blue
SDK	Software Development Kit
TN	Twisted Nematic
VR	Virtual Reality
XML	eXtensible Markup Language

Introduction

"It is not the strongest of species that survives, not the most intelligent... it is the one that is the most adaptable to change." - Charles Darwin

1.1 CONTEXT

Throughout evolution, humans and their ancestors have faced potentially lethal dangers, most notably through predation. As reviewed by Isbell [1], snakes have been lethal since the origin of primates. Their survival depended on how quickly their visual system would detect threats before they strike. Those whose visual system was more responsive had better chances for survival. This responsiveness to fear and threat became an unconscious, instinctive reaction.

Charles Darwin was one of the first scientists to study our unconscious reactions to threat. In 1872, Darwin went to a reptile house in the zoo and stared at a poisonous viper behind glass [2]. He pledged not to move or flinch when the snake tried to strike. Darwin had never experienced the bite of a poisonous snake, and yet, without fail, he "jumped a yard or two backwards with astonishing rapidity" every time the snake lunged at the glass even though in his conscious mind he knew the snake could not hurt him. Darwin concluded that our fear reflexes are guided by an evolutionary system deep in the human brain.

People who feared the right dangers survived to pass on their genes. In passing on their genes, the trait of fear and the response to it were selected as beneficial to the race. Fear is an act of survival, as it promotes escape and avoidance when life is at stake.

1.2 BACKGROUND TO THE RESEARCH

Threat processing is a psychology field that, essentially, studies how the emotional stimuli that humans perceive as fear are processed under unawareness. Research of these matters is actively performed in Experimental Psychology, addressing the different cognitive and behavioral aspects of threat processing.

Researchers study human and nonhuman primates’ cognitive responses to threats in the experimental psychology labs, presenting threatening stimuli, such as snakes, arachnids, and fearful faces, in different visual conditions of unawareness. Subjects are also shown non-threatening (neutral) stimuli to compare cognitive response versus threatening stimuli. Researchers display these stimuli to participants using computer monitors in the psychology lab experiments.

Several studies support that threatening stimuli elicit faster cognitive activation in humans [3] and primates [4] [5] when presented in visual unawareness states. Visual unawareness is achieved in the experimental lab by presenting stimuli to subjects in visual taxing conditions, such as in the peripheral visual field, with brief exposure duration and cluttered environments. Some researchers studied a novel interocular suppression paradigm, Continuous Flash Suppression (CFS) [6], creating an absence of visual awareness, and measured the time needed for stimuli, such as faces, to break the suppression and gain access to awareness [7], breaking-Continuous Flash Suppression (bCFS).

Recently, Gomes et al. studied the advantage of threat stimuli, such as arachnids [8] and snakes [9], in accessing awareness using bCFS. The data gathered in these studies presented consistent preferential processing for threatening stimuli, showing more prevalence for snakes than other threatening species under visual taxing conditions. These results align with Isbell’s review [1] that primates’ visual systems evolved to be more responsive to snakes, their prime threat for survival.



Figure 1.1: Snakes and arachnids are among the most common threat inducing species to humans. Images retrieved from here¹ and here².

1.3 CHALLENGES

Lab experiments to study threat processing allow scientists to study specific cognitive and behavioural aspects of emotional stimuli that humans perceive as threats in a controlled setting. Creating such environments that immerse the participant is a challenging task:

- The lab is far from an ecological setting for studying threat and elements in the lab environment might result in sources of distraction for the participants;
- The characteristics of the stimuli and their location’s spatial perception can be affected by the adopted experimental paradigm [10];

¹https://www.pinclipart.com/pindetail/bxomwi_we-foster-support-for-snake-conservation-cobra-silhouette/

²https://www.pinclipart.com/pindetail/ibombwb_spider-web-silhouette-drawing-dessin-d-araigne-a/

[last accessed: 28/02/2021]

- The experimental paradigms often involve implementing complex visual stimuli techniques (e.g., masking stimuli to preclude their access to consciousness) or using expensive equipment (e.g., mirror stereoscope).

For example, Gomes et al. conducted experiments [8] [9] involving a different technique that required manipulating the Red, Green, and Blue (RGB) channels to display the stimuli with anaglyph glasses using the bCFS experimental paradigm. This manipulation of the stimuli's natural colors limits the experiment's ecology validity by compromising the proper color tone. In this case, the experiment results are mostly tied to the stimuli's shape.

In this regard, proposing and using methods and technologies that might immerse the participants in the experimental setting, providing a richer manipulation of visual stimuli (e.g., showing a threat stimuli in more realistic conditions), might benefit the evolution of threat processing research.

In recent years, Virtual Reality (VR) and Augmented Reality (AR) have been considered for implementing more realistic experiment environments for participants. Foerster et al. [11] [12] used a VR Head Mounted Display (HMD) to assess human visual processing capabilities with such devices, showing that VR provides reliable results similar to current traditional methods. VR devices also benefit from enclosing the entire visual field, eliminating any visual distractions for experiment participants. Korisky et al. [13] showed that using AR goggles to suppress real-world objects using CFS provides suppression durations comparable to those obtained in experiments using on-screen CFS. Using AR goggles in experiments can provide realistic environments as there can be, using real-world stimuli.

Nevertheless, the use of these technologies is still in its infancy, and much needs to be done to increase its value further and use it in Experimental Psychology.

1.4 OBJECTIVES

VR is an up-and-coming technology that has a high potential for Experimental Psychology. This dissertation's overall goal is to introduce VR technology into Experimental Psychology research by evaluating VR as a suitable option for experiments in the threat processing field while exploring its different requirements and features. To this end, it was decided to address the implementation of a current and challenging experimental paradigm, bCFS, since it requires special presentation techniques in which VR technology could excel.

Therefore, this work's objectives that are intended to be achieved are listed in the following points:

- Review current methods and advantages of VR in Experimental Psychology along with the approaches considered to study threat processing (e.g., bCFS);
- Define an experimental protocol, in collaboration with the Department of Education and Psychology, supporting the validation of VR based paradigms for threat processing research;
- Explore VR frameworks that allow using the potential of VR technology to implement the devised experimental paradigms;

- Develop proofs-of-concept for the proposed methods by supporting the implementation of new experimental methods to be used in Experimental Psychology studies considering bCFS;
- Validate the proofs-of-concept according to the experimental protocol established with the Department of Education and Psychology.

1.5 DOCUMENT STRUCTURE

This Dissertation is divided into five chapters, of which chapter 1, the Introduction, has already been presented. The rest of the document is structured as follows:

Chapter 2: Background context on the subject in question. It describes VR technology's evolution history and its proliferation in the modern computer era, along with an overview of VR's usage in Experimental and Clinical Psychology. Then, it discusses the research in the threat processing field and the standard methods currently used to perform the psychological experiments. The chapter is concluded by associating VR's potential in Experimental Psychology.

Chapter 3: Describes the experiment protocol designed, in collaboration with the Department of Education & Psychology of the University of Aveiro, to conduct the experiments to evaluate VR technology for psychological experiments. It is also delineated the requirements list that guides the software platform development and clarifies the equipment selected for each experimental setting.

Chapter 4: Details the software platform development, presenting its designed architecture and development solutions to support the experiment protocol and requirements. It is also explained the procedures and features for the creation of experiments and application program usage.

Chapter 5: This last chapter presents a conclusion of all the work developed and a summary of what was accomplished. Also, suggestions for future work are presented to continue this project.

Appendices: Finally, and to provide additional detail on some of the aspects of the work carried out, not presented in main chapters for the sake of brevity, this document also includes 3 Appendices.

Appendix **A** explains the estimations calculated for the VR HMD's display sizes and eye-to-display distance required for the stimuli presentation process. It discusses some of the nuances of using VR headsets for stimuli presentation and finalizes by explaining the in-app stimuli presentation methodology.

Appendix **B** presents a manual on the configuration file usage to prepare experiments, discussing the parameters and values used and their role for experiment set up. It also addresses its compatibility with other types of experiments and future improvements.

Appendix **C** explains the keys and buttons implemented to control the program and input responses during experiments. It also mentions the system requirements and presents possible improvements for application usage.

Background

Experimental Psychology seeks to explore and better understand cognitive behavior through experimental research methods. Visual stimuli have a substantial importance in Experimental Psychology, which prompts the improvement of techniques and quality of researchers' methods. A prominent area for improvement is using technology that better engages the participants' attention to the experiments. Virtual Reality promises improvements in this aspect. In recent years, experimental psychologists have actively explored this technology, showing encouraging results in Experimental Psychology usage.

This chapter details Virtual Reality technology and briefly reviews its usage in recent Experimental Psychology studies. Then, it discusses threat processing studies, outlining some of the leading experimental paradigms used. Finally, an association between virtual reality usage in threat processing research is made, exhibiting the advantages the technology promises, comparing them to traditional methodologies.

2.1 VIRTUAL REALITY TECHNOLOGY

Virtual reality is a technology that has been around for decades. The initial versions were bulky and expensive, only used in high-end industrial and military facilities. Ivan Sutherland invented the widely considered first HMD system in 1968 [14]. The device was so heavy it had to be tethered to the ceiling, as shown in figure 2.1. Nowadays, improvements in computing power and manufacturing massification of computer hardware allow technologies such as VR to be more compact, portable, and comfortable to use. These technology improvements result in more accessibility at the consumer-level, increasing VR's popularity.

In general, VR is the computer-generated simulation of a Three-Dimensional (3D) environment using special electronic equipment. The objective is to achieve a strong feeling of being present in the virtual environment by simulating as many senses as possible, such as vision, hearing, touch, and even smell [15]. VR involves wearing an HMD (such as goggles) to view the 3D environment and look around by moving your head and walk using hand controls

or motion sensors. Hand controls also allow interaction with the virtual world's objects and performing other specific actions.

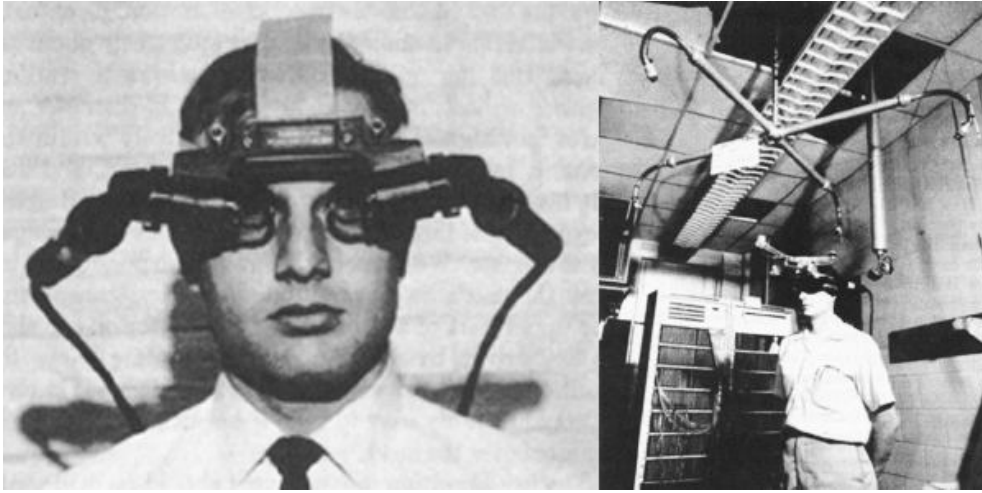


Figure 2.1: "The Sword of Damocles", Ivan Sutherland, 1968. With his students' help, Ivan Sutherland created the first virtual reality and augmented reality head-mounted display system. Image retrieved from here¹.

2.1.1 VR Accessibility and Usability

Nowadays, VR HMDs are available through different manufacturers. The Oculus Rift² and HTC Vive³ are considered the first consumer-level VR systems to popularize the VR market. These devices include integrated displays in the goggles and motion sensors to track them virtually (as illustrated in figure 2.2). The affordable prices, simple installation, and ease of use made VR an appealing choice for new experiences, both at a consumer and non-consumer levels.

VR's usage is currently more prevalent among gaming applications, but its use in non-gaming applications can prove beneficial by providing immersive experiences. Allowing people to virtually travel to other places [16], train specialized personnel for specific practice exercises [17], and aid in remote therapy for mental health patients [18] are among the many possible VR uses.

¹<https://alchetron.com/Ivan-Sutherland> [last accessed: 28/02/2021]

²Oculus home page (<https://www.oculus.com>).

³Vive home page (<https://www.vive.com>).

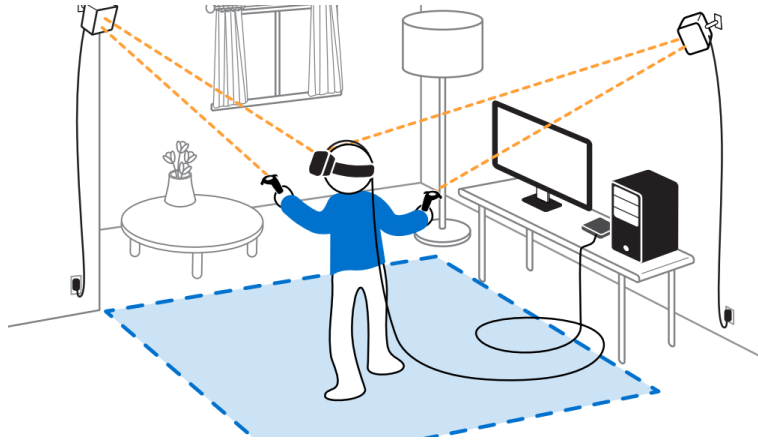


Figure 2.2: A typical modern VR setup. The VR HMD is connected to a computer that processes the VR environment and sends it to the HMD. The motion sensors (both at the top corners of the image) track the HMD and hand controllers in line of sight (dashed orange lines) to simulate their real-world movement in the virtual environment. The blue area represents a marked zone where the person can move freely. Image retrieved from here⁴.

2.1.2 Advantages and Disadvantages of VR

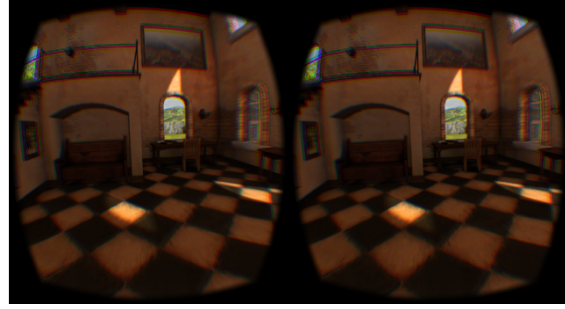
The best asset of VR systems is the level of immersion in a 3D virtual world it introduces by stimulating, most frequently, three of our senses: vision, touch, and hearing. The vision is the most stimulated by this technology, having modern display techniques inside the goggles that provide a wide Field of View (FOV) of the virtual space. The goggles also provide a shielded enclosure to prevent possible visual distractions from the outside. Touch is stimulated through haptic feedback, although still very limited, in the hand controllers and hearing through sound from headphones or speakers. VR devices are also very portable and light-weight, making them easy to transport and install for experiments and comfortable for people to use.

However, in some circumstances, the technology can not achieve a complete immersion level given some limitations. Due to the eyes being very close to the displays and the resolution not being significantly high, a person can notice a screen door effect [19] [20]. Simultaneously, a form of chromatic aberration [21], caused by lens dispersion, can be detected (see figure 2.3 for an illustration on these factors). Some people may be prone to cybersickness [22], similar to motion sickness, in particular circumstances that involve virtual locomotion. However, existing research shows that some techniques [23] [24] can reduce cybersickness, such as sitting down and assuring good VR experiences by improving any latencies or erratic movement in the application.

⁴<https://learnvr.org/3-dimensional-space-and-virtual-reality-understanding-x-y-z-coordinates/>
[last accessed: 28/02/2021]



(a) Screen Door Effect.



(b) Chromatic Aberration.

Figure 2.3: Screen Door Effect and Chromatic Aberration Examples. On the left, screen door effect caused by low pixel density compared to viewing distance through the VR lenses. On the right, lens view of the chromatic aberration caused by lens dispersion. Images retrieved from here⁵ and here⁶.

2.1.3 Developing VR Experiences

The growing popularity of VR technology and the demand for better VR experiences has motivated the creation and improvement of VR development tools. The most popular VR development frameworks are Unity⁷ and Unreal Engine⁸. While both software frameworks are game development engines at heart, they also allow developing non-gaming applications for many sectors^{9,10}. Unity is arguably the most popular and easy to use software development framework, widely adopted by researchers to create VR experiences for their studies (see section 2.2).

Unity

Unity is a cross-platform game engine developed by Unity Technologies [25] and is one of the most prevalent for VR development. It provides the tools to develop VR applications quickly while being reasonably easy to use. Unity supports C#¹¹ to develop scripts that implement the application logic and behavior [26]. Unity's vast popularity comprises a large community that shares resources and documentation to aid in development.



Figure 2.4: Unity Logo.

⁵<https://byteside.com/2020/10/samsungs-new-displays-to-eliminate-screen-door-effect-in-vr/>

⁶<https://forums.oculusvr.com/community/discussion/28380/chromatic-aberration>

[last accessed: 28/02/2021]

⁷Unity home page (<https://unity.com/>).

⁸Unreal Engine home page (<https://www.unrealengine.com>).

⁹Unity has been used in sectors such as Automotive, Filmmaking, and Architecture, among others.

¹⁰Unreal Engine has been used in sectors such as Architecture, Live Events, and Simulation, among others.

¹¹C# documentation (<https://docs.microsoft.com/en-us/dotnet/csharp/>) .

Unity has a wide variety of supported platforms [27], including practically every VR platform [28], making it simple to later implement other VR systems. For VR development, Unity contains tools such as SteamVR and the OpenVR Software Development Kit (SDK).

SteamVR & OpenVR SDK

SteamVR¹² is a runtime platform developed by Valve¹³ for VR systems [29], including a set of tools and services to run and create VR experiences [30]. One of such tools is the OpenVR SDK¹⁴. The OpenVR SDK is implemented in SteamVR and acts as an Application Programming Interface (API) between VR software and hardware, providing a way to develop VR experiences for different VR systems [31].

Unity SteamVR support is made via a plugin released by Valve called SteamVR Unity Plugin [32] [27].

2.2 VIRTUAL REALITY RESEARCH IN EXPERIMENTAL PSYCHOLOGY

With the rise of VR accessibility, researchers started adopting the technology to study its effectiveness to provide reliable results and improve the quality of the experimental methods used in their research. Experimental Psychology is a scientific field that started using VR more frequently in experiments to assess its benefits in various Psychology fields.

The enhanced visual perception and sense of existence in the virtual world induced by virtual reality can revolutionize current treatment forms for psychiatric disorders in psychotherapy. Researchers have reviewed VR to evaluate its efficacy in exposure therapy to treat phobias [33] [34], effectively showing that VR exposure is preferred over traditional methods such as In Vivo exposure [35]. VR exposure therapy has been evaluated over many studies for different phobias, such as social anxiety disorder [36], acrophobia [37], animal phobias [38] [39], and others [40] [41] [33], finding that VR is a promising therapy innovation.

Social acceptance of new technologies, such as VR, is essential for its standardization in psychological experiments and procedures. Older adults usually reveal more negative attitudes towards new technologies [42] [43] than younger adults, eliciting more age-sensitive designs for easier acceptance. However, researchers found that older adults' attitude towards VR was positive [44] after the first exposure to immersive virtual reality. Furthermore, older adults, having more sensory and cognitive frail conditions, completed VR experiments without reporting cybersickness [44] or adverse side-effects [45]. Using a newer generation of VR HMDs can significantly reduce adverse symptoms [46], which is essential to ensure the reliability of results and health and safety standards.

VR's enhanced visual perception and enclosed visual field are the most common features researchers look for with this technology. Foerster et al. studied the advantage of using VR to assess visual processing capabilities [11], which depend on visual testing conditions that threaten reliability and visual processing speed. They found that VR produced reliable results

¹²SteamVR main page (<https://www.steamvr.com/>).

¹³Valve home page (<https://www.valvesoftware.com>).

¹⁴OpenVR SDK Repository (<https://github.com/ValveSoftware/openvr>).

as the standard method and improved visual processing speed and visual working memory. The same authors [12] also found very similar results in a related study focusing more on visual selective attention and processing capacity. They commented on VR’s convenience for bedside testing (i.e., the performance of clinical diagnostic testing at the site of patient care) on patients who cannot be transported or sit upright when using HMDs for clinical purposes. Studies also observed improved visual search performance on real scenes using VR compared to Two-Dimensional (2D) displays [47], replicating results successfully when translating traditional paradigms to VR [48]. These findings further cement VR as a suitable contender to improve experimental paradigms for visually focused experiments and provide higher quality ecological conditions.

Brookes et al. have created the Unity Experiment Framework (UXF) [49] to assist in designing behavioural experiments using Unity. This framework contains features that help behavioural scientists implement several forms of data collection and configuration settings to create their experimental programs, hiding potential shortcomings for scientists unfamiliar with programming. This framework supports Unity projects in any platform but was designed specifically for virtual reality behavioural experiments. Although still in early development, UXF elicits interest to be used for Unity developed experiments.

VR technology has a lot to offer for Experimental Psychology, and researchers should further explore this technology to improve and expand the conventional methods and procedures commonly used. The sense of involvement in an environment is paramount for experimental studies that benefit from visual fidelity and ecologically valid settings. One such Psychology field that can benefit from VR’s previously mentioned advantages is threat processing study.

2.3 THREAT PROCESSING RESEARCH

Threat processing is a Psychology field that studies human cognitive response to fear (threat) stimuli. Research in this field studies human processing of ecologically-relevant stimuli and their susceptibility in accessing visual awareness, molded by our ancestors’ evolutionary survival [1] against predators. Over the years, threat processing research has implemented different experimental paradigms to assess preferential access to awareness from specific stimuli.

Researchers found that threatening stimuli detection prevails over neutral stimuli using experimental paradigms such as visual search in visually taxing conditions. For instance, presenting stimuli in the visual periphery [50], stimuli camouflaged in a cluttered environment [51] [52] [50], and brief stimulus exposures [52] [50], which are conditions that may have been critical for survival, shared similar results favouring threatening stimuli detection. The results show more preference of snakes in accessing visual awareness, which is more of an ecologically-relevant threat [1] than other phobia-inducing stimuli such as arachnids. Several experimental paradigms have been considered for this field of research, and, recently, a novel technique has been proposed with exciting results: bCFS, a toned-down version of CFS [6]. CFS is an interocular suppression technique that allows stronger visually demanding

conditions by suppressing static images with dynamic noise. It has awakened more interest by researchers on how threatening stimuli are processed under these conditions.

CFS is a variant paradigm of binocular rivalry [53] in which each image, the stimulus, and dynamic noise, is displayed to each eye. The traditional methods to get this binocular effect requires a monitor to display the stimuli and special equipment used to display the images to each eye. Researchers commonly use a mirror apparatus, such as a mirror stereoscope, that reflects an image to each eye or a Red-Blue 3D anaglyph glasses, manipulating the RGB channels to filter the stimuli to each lens, as demonstrated in Gomes et al.'s experiments [8] [9]. The experiments require participants to place their head in a fixed position, usually using a head-chin rest. The anaglyph glasses method's issue is that the images are constrained to the red and blue channels, thus degrading the stimuli's ecological value by precluding their presentation in their actual colors. The mirror apparatus is the preferable method of the two, although it is an expensive and delicate item.

Nevertheless, researchers have explored CFS, studying the strong suppression effect of this technique and that threatening stimuli, such as fearful faces [54] [10], angry body postures [55], spiders [56] [8] and snakes [8] [9], have preferential access to visual awareness and break suppression (bCFS [7] [57] [58]) faster than neutral stimuli. The bCFS paradigm shortens the time for the stimuli to break suppression, which would take minutes compared to CFS. In order to control how long it takes to break suppression, the bCFS paradigm is implemented, introducing a progressive fade out of the dynamic noise to reduce the suppression effect. This variant of CFS has been adopted by several studies, including Gomes et al.'s [8] [9] experiments.

2.4 DISCUSSION

Many studies have analysed the potential VR in Experimental Psychology due to the high level of immersion and enhanced visual perception it provides to experiments. CFS is an experimental paradigm that would highly benefit from a more ecological setup, namely favoring the binocular aspect, so it would be an interesting point to demonstrate VR usage in the lab.

Abandoning 3D anaglyph glasses adds the possibility of applying natural colors to stimuli using the VR display, providing a better ecological setting. Although a better option than the anaglyph glasses, the mirror apparatus is an expensive and delicate item compared to a VR headset. The traditional methods require using head-chin rests to fix the participant's head in place, which is also very expensive. The VR HMD provides the same binocular functionality as the mirror apparatus while being more comfortable to use in the experiment tasks, as the head-chin rest is not required. The VR headset displays a screen to each eye, making it possible to show different images to each eye, as is required with the CFS paradigm. The visual isolation from the surrounding environment provided by the HMD goggles helps to maintain the participant focused on the experiment tasks by blocking any visual distractions from the outside.

Establishing the suitability of VR to implement such a challenging technique as bCFS can positively contribute to improving research on threat processing and can serve as grounds for continued evolutions to improve the experiments' ecology in the future. To this end, Unity's popularity and potential capability to develop a variety of experiences elicit interest to explore it for core aspects that fulfill this work's objectives. Unity also supports frameworks built by the community, such as UXF, that help create applications to support experimental protocols. It may be interesting to analyze whether such tools are relevant to the current work.

There is currently a lack of research for CFS using VR for threat processing, and it would be interesting to see more research with these settings. For this reason, this dissertation will focus on implementing VR for threat processing experiments using the CFS paradigm.

2.5 CHAPTER CONCLUSIONS

This chapter provides a short overview of Virtual Reality technology and its use in recent Psychology research, with a greater focus on the advantages it brings for experimental binocular paradigms used in threat processing research.

To better understand Virtual Reality, a first introduction is done by briefly discussing its technological evolution and describing the features such technology provides compared to desktop monitors. Some of the recent studies that evaluate VR usage in Experimental Psychology are then discussed to get a better idea of VR's benefits. It is then discussed threat processing research focusing on the leading experimental paradigms used, such as CFS, an experimental binocular paradigm.

Finally, it is discussed the advantages that the VR technology could provide to threat research, comparing to current methodologies. Consequently, this brings the focus of this dissertation, which is to implement VR for threat processing experiments using the CFS paradigm as a demonstration. These experiments will allow evaluating VR's capabilities in the Experimental Psychology lab. The next chapter will discuss the elaboration of an experimental protocol to conduct these experiments.

Experiment Protocol

This chapter contextualizes the objectives, protocol methodology, and procedure for the experiments that support VR equipment validation to implement the bCFS technique. Based on the defined experimental protocol, the experiments' requirements are delineated, which inform the computational system's development to support it. Additionally, the available equipment deemed relevant for this work is described, which further establishes a set of requirements for the software platform to adopt.

3.1 EXPERIMENT CONTEXT

According to this dissertation's objectives, the focus is to develop a computerized system that allows assessing the viability of using a VR HMD in the experimental lab for threat processing experiments considering the bCFS paradigm as a demonstration given the challenging characteristics of the technique as a visual stimulus. In collaboration with the Department of Education and Psychology of University of Aveiro, an experiment protocol was designed to study the cognitive response to threat stimuli. It aims to test VR's reliability to produce similar results to previous works that successfully used other platforms to implement bCFS (desktop monitor). The next section presents the devised protocol methodology and procedure for the experiments.

3.2 PROTOCOL

In the following section, the main aspects that the experiment protocol encompasses will be discussed. First, the experiment sessions' methodology is described, followed by the stimuli used in the experiments, and finally, the list of requirements the application should incorporate.

3.2.1 Methodology

The participants perform the experiments on both platforms (desktop monitor and VR), separated a week apart. They are randomly divided into two groups, each group with the

same amount of participants. Each group starts the first session on a different platform than the other, i.e., group 1 uses the desktop monitor, while group 2 uses VR, and swap platforms in the following week, as shown in table 3.1.

Group	Platform	
	Session 1	Session 2
1	Desktop Monitor	Virtual Reality
2	Virtual Reality	Desktop Monitor

Table 3.1: Experiment session methodology. Participants are separated into two groups. In each group, participants start the first experiment using the opposite platform than the other group. In the following week, participants swap platforms for the session.

Establishing a week of separation between sessions allows the participants not to start the second experiment already accustomed. Allowing time off the experiment is essential to provide reliable results. If done incorrectly, the results could show improvements in the second experiment due to participants already accustomed to the CFS effects. This methodology grants more reliable results for the assessment of VR’s viability in this experiment.

3.2.2 Stimuli

The stimuli selected for this experiment consists of five images of snakes as threat stimuli and five images of birds as neutral stimuli (see figure 3.1 for an illustration). All images used are grayscale and were chosen from those used in Gomes et al. (2017) experiments [8], snake images initially from Soares et al. (2014) Experiment 4 [50]. The images are trimmed of any white background space to the smallest dimension that encompasses the stimuli to ensure coherence usage and processing.



Figure 3.1: Stimuli images selected for the experiments. The top row comprises the neutral stimuli (birds) and the bottom row the threat stimuli (snakes). Images adapted from the Gomes et al. (2017) experiments [8].

The literature often uses visual angles to define the stimuli dimensions since they define the size of the stimuli in our field of view instead of on the display referential (e.g., pixels). For instance, the perceived size of a line with 100 pixels length depends on the screen resolution, size, and the distance from the viewer to the screen. The specification in visual angles enables that, for each case, the size of the stimuli in display coordinates is computed when the experimental setup is fully defined and aims to ensure consistency across setups.

Participants are presented with a $16^\circ \times 16^\circ$ frame with a 0.5° white noise pattern border. The stimuli appear in the foveal and peripheral vision, distancing 1° and 6° of visual angle, respectively, both at the left and right sides of a fixation cross located at the center of the frame (see figure 3.2 for an illustration of the frames).

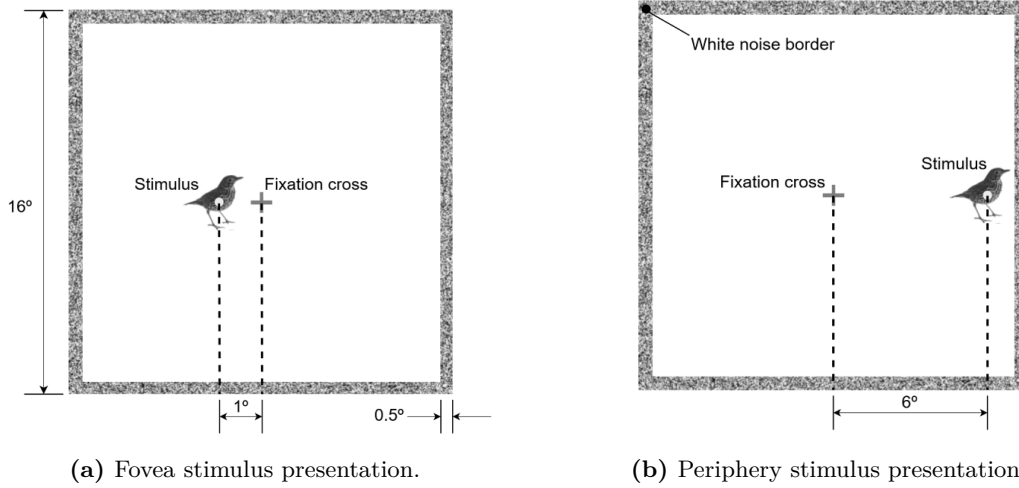


Figure 3.2: Frame dimensions and stimuli position presentation for the experiment. The left image shows the fovea position, while the right image shows the periphery position for the stimuli. Each position is presented both at the left and right sides of the frame.

The CFS effect is created using Mondrian pattern mask images in the experiments (see figure 3.3 for a reference). These masks are composed of randomly arranged grayscale circles of varying diameters and presented at a rate of 10 frames per second to create the CFS effect. These mask images were generated using a Python¹ script². Due to not having a mirror apparatus display available, to allow a binocular viewing of an image to each eye, enabling CFS requires using "Red-Blue anaglyph glasses" (e.g., Gomes et al., 2017 [8] & 2018 [9]) on the desktop experiment. Stimuli are presented with a blue RGB channel filter and masks with a red RGB channel filter.

When overlapping the stimuli and masks in the frame, each eye of the participants will only see the images in the same color as the glasses' corresponding lens (for an illustration of this process, see figure 3.3). For the sake of equivalence and testing all setups, it was considered the implementation using anaglyph glasses and extended the same technique to the VR setup. This approach deteriorates the experiments' ecological validity by using unnatural colors but maintains stimuli coherence on both platforms for a more reliable evaluation. This approach is also supported on the mirror setup for future-proofing. Nevertheless, the same procedure to generate and show the stimuli in the VR and mirror setups can be done adopting the natural colors on the stimulus and mask frames.

According to the literature, the most common choice is to present the stimulus to the dominant eye. While, for the sake of illustration, the stimulus is shown as being presented to

¹Python main page (<https://www.python.org/>).

²The Python script was retrieved from this repository (<https://github.com/AnoAn/Mondrian-generator>) [last accessed: 28/02/2021].

the dominant eye (see figure 3.4), it can also be presented to the non-dominant eye. It depends on the experiment’s research questions addressed; for instance, presenting the stimulus to the non-dominant eye can make it even harder to detect.

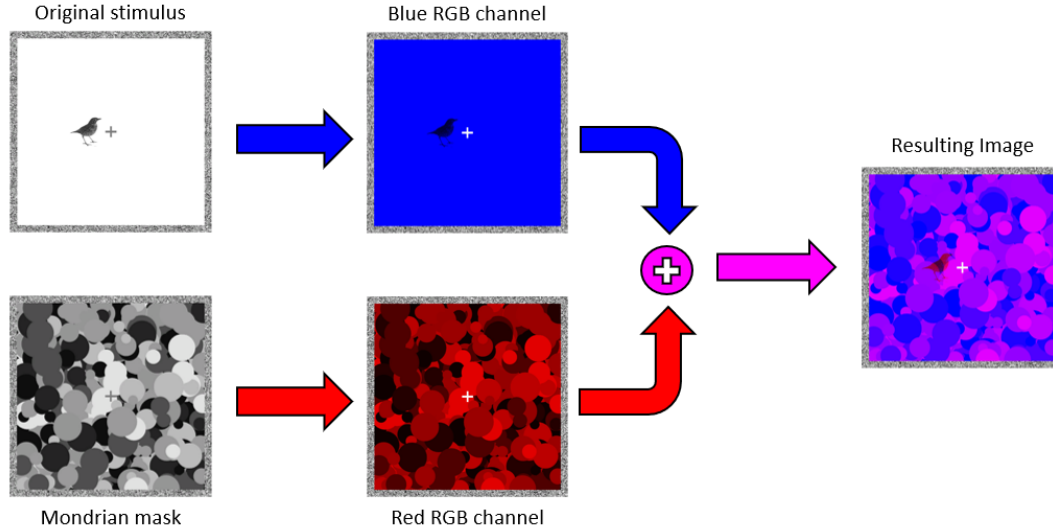


Figure 3.3: Frame RGB channel filter processing for the Red-Blue anaglyph glasses. From left to right, the original stimulus and Mondrian mask in the frame, their filtering to the blue and red RGB channels, respectively, and the frame with the resulting RGB channel filtering

3.2.3 Experiment Procedure

The procedure for the experiments is similar to those used in Gomes et al. experiments [8] [9]. Participants’ eye dominance is evaluated using the Miles test [59]. They are seated down either in front of a computer monitor or with a VR HMD. Participants are positioned 50 cm away from the screen for the monitor experiments, with the screen at eye level, using a head-chin rest to fix the head in the same position. Using VR, participants place the HMD in their head and adjust the head straps to sit comfortably, positioning the goggles to the closest distance to the eyes to obtain the widest FOV achievable. There is an opportunity for the participant to adjust the VR goggles’ Interpupillary Distance (IPD) before the experiment and a calibration phase to adjust the frame’s position in the display. For reference, the participant follows the same desktop platform procedure for the mirror apparatus, placing the device in an appropriate position that equates to 50 cm from the monitor. When the participant is comfortable and ready, the experiment starts.

Each trial in the experiment (see figure 3.4 for an illustration) starts by presenting a blank frame for a second, containing only a fixation cross. Then, CFS masks appear, to the non-dominant eye, interchanging every 0.1 seconds (10 frames per second) to create the CFS effect. During this process, the stimuli image starts to fade in for 1.1 seconds until fully visible. Then, the CFS effect is reduced for 4 seconds until it is no longer visible, to break suppression (bCFS). The trial ends when the participant inputs a response or after 7 seconds.

All participants are instructed to identify, as quickly as possible, which side in the frame the stimulus, or any part of it, became visible. All experiments start with a training session, consisting of 20 trials, including stimuli not shown in the main experiment. This session allows participants to understand the task better and clear any doubts that may surface. The main experiment consists of 120 trials ($10 \text{ stimuli} \times 2 \text{ sides} \times 2 \text{ positions} \times 3 \text{ repetitions}$). All trial responses (including the training) are stored accordingly (reaction times, side responded, if the participant responded to the correct side or not). The start, duration, and stop times of each experiment's phase, together with relevant information about the participant, such as age, gender, and dominant eye, are also stored in the experiment's results.

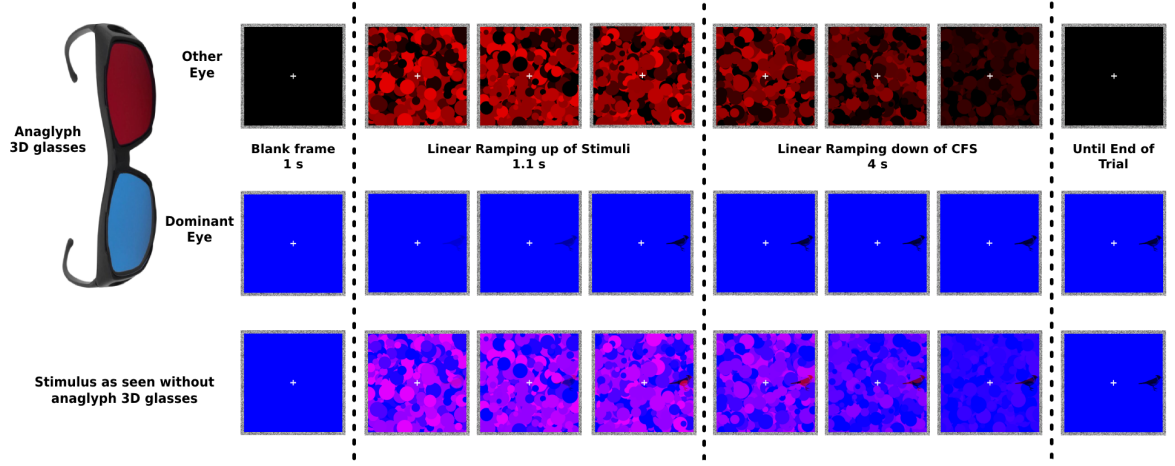


Figure 3.4: Experimental trial stimuli presentation procedure. The image includes the timings of each trial phase and illustrative frames for the content seen through the anaglyph 3D glasses lenses from the content on screen (bottom row). The VR platform is only presented the separated frames (first and second rows) to each eye. The presentation procedure is equal on all platforms.

3.3 COMPUTATIONAL REQUIREMENTS

The experimental protocol described above requires that the computational methods support stimuli creation, implement the different experimental steps, and allow experimental data collection. A list of requirements was established from an analysis of the protocol, which directed the software platform's development to satisfy them successfully. The list of requirements is divided into three main parts, which are as follows:

- **Overall experiment implementation:**
 - Support VR technology and standard methods in the experiments to compare results;
 - Support visual stimuli based experiments allowing independent stimuli, per eye (e.g., bCFS);
 - Support multiple ways for participants to input responses, and standardize one solution for all experiment platforms.

- **Stimuli presentation:**
 - Implement the trial methodology specific to the scenario of bCFS (as exemplified in figure 3.4);
 - Support the definition of display parameters in visual angles.
- **Configuration and data storage:**
 - Support experiment setup (e.g., select stimuli images, configure stimuli sizes) through a customizable human-readable configuration file;
 - Register and store participant responses in an easy to process file format.

Application support for VR is, naturally, an essential part. The main goal is to support the evaluation of VR's viability for threat processing in the experimental lab using CFS paradigms. To this end, it was agreed that the experiment would benefit from having a standardized method, between all platforms, for participants to input their responses. Using the hand controllers, bundled with the VR system, in each experiment would allow more coherence in the responses and offer participants to be in a relaxed position. Developing the application infers another goal: to establish the overall grounds that enable the implementation of new experimental methods relying on visual stimuli presentation for Experimental Psychology. Without loss of generality and given the relevance and characteristics of bCFS (e.g., profiting from binocular rivalry), the proposed platform is designed to fully endorse the experimental protocol for its testing in VR and validations towards other platforms.

The more technical aspects of this list of requisites, such as the configuration file, visual angle display definition support, participant response processing, and all the requisites mentioned above, are discussed in more detail in chapter 4.

3.4 EQUIPMENT

The protocol designed, in collaboration with the Department of Education and Psychology, was made considering the validation of a set of technologies to implement the bCFS experimental paradigm. Each experiment setting requires specific equipment to interact with the application system and display the stimulus in its way. In this sense, the existing equipment available at the lab was considered. The equipment required for the experiments consists of a VR system (HMD, hand controllers and base stations), a computer monitor, and anaglyph 3D glasses. The mirror apparatus equipment is also included for reference. The following section describes the equipment's specifications and intended use.

3.4.1 HTC Vive

HTC Vive, or Vive for short, is a VR headset developed by HTC³ and Valve. This VR system is composed of an HMD, controllers, and base stations (see figure 3.5 for an illustration). The Vive was chosen due to being one of the two VR units available in the laboratory and its widespread adoption in previous studies and among VR enthusiasts.

³HTC home page (<https://www.htc.com>) .



Figure 3.5: HTC Vive system. From top to bottom: Vive base stations, HMD, and hand controllers. Image retrieved from here⁴.

Head-mounted Display

The Vive headset is the display used for the VR experiment. It possesses two Active-Matrix Organic Light-Emitting Diode (AMOLED) displays that project different images, one for each eye, to produce more realistic 3D virtual environments. The specifications⁵ for the Vive HMD are presented in table 3.2.

Specifications	
Feature	Technical Specification
Display Device	Dual AMOLED panels
Resolution	1080x1200 pixels per eye, 2160x1200 pixels combined
Display Size	91.44mm diagonal
Display Dimensions	67.9mm width; 61.1mm height (estimated)
Refresh Rate	90 hertz (Hz)
Field-of-View	110 degrees

Table 3.2: HTC Vive HMD specifications.

The VR HMD serves as a dual-screen that projects two different images for each eye in the experiments, allowing the CFS effect. The displays are enclosed, in a shielded way, covering any possible distractions for participants during the experiment. The possibility of displaying different images to each eye can benefit experiments by showing images in natural colors, rather than using specific techniques that require special equipment and changes in the RGB channels, as is the case in the desktop experiment. Nevertheless, the VR experiment will use the same color scheme required for the desktop.

The official Vive specifications do not mention the display size. As such, these values had to be estimated. For an in-depth review of these measures, see Appendix A.

⁴<https://blog.vive.com/us/2016/02/21/unveiling-the-vive-consumer-edition-and-pre-order-information/> [last accessed: 28/02/2021]

⁵For more specification details, the reader is forwarded to the Vive Specs & User Guide (<https://developer.vive.com/resources/vive-sense/hardware-guide/vive-specs-user-guide/>) [last accessed: 28/02/2021].

Hand Controllers

The Vive VR system contains hand controllers that are used to interact with the virtual environment. For both VR and desktop experiments, the participant holds a controller in each hand and uses it to interact with the application, inputting reaction responses during the trials. Maintaining the hand controllers' usage for both platforms provides a standard method for participants to input their responses, assuring more coherence in the results. This approach takes advantage of the hand controllers, allowing participants to be in relaxed positions instead of constraining their hands on a keyboard, which may feel uncomfortable.

As a side note, the hand controllers require that the Vive HMD is connected to the computer and detected by the tracking system. Therefore, in the desktop experiment, the HMD needs to be present to use the hand controllers for inputs. There is a workaround to make the hand controllers work alone using Vive trackers or fiddling with configuration files, but, due to lack of such equipment and time constraints, it was not explored.

Base Stations

Vive base stations are the primary devices of Vive's tracking system. The HMD and controllers require them to track their position and communicate. The base stations benefit from being very simple to set up, only needing to plug them into the power outlet and pressing a synchronize button after being mounted in place.

3.4.2 Desktop Monitor

The monitor used for the desktop experiment is a Dell Professional P2212H⁶ (see figure 3.6 for an illustration), and its specifications can be seen in table 3.3.

Specifications	
Feature	Technical Specification
Display Device	Light-Emitting Diode (LED) Twisted Nematic (TN) panel
Resolution	1920x1080 pixels
Display Size	546.8mm diagonal
Display Dimensions	476.64mm width; 268.11mm height
Refresh Rate	60 hertz (Hz)

Table 3.3: Desktop monitor specifications.

Since the VR and desktop displays are of different types, some variables are introduced that need consideration. The nature of TN panels may induce incorrect color presentation when viewed from a specific angle. However, the participant is always in the correct position to view the display's correct color scheme granted by the experiment protocol. The TN panel may also present different color tones than the VR AMOLED screen. The screen door effect is also much more noticeable in the VR HMD. These are minor details that should be acknowledged when evaluating both platforms, as they can influence results.

⁶Dell Professional P2212H specification page (<https://www.dell.com/ae/business/p/dell-p2212h/pd>) [last accessed: 28/02/2021]



Figure 3.6: Desktop monitor selected for the experiments. Image retrieved from here⁷.

Red-Blue Anaglyph 3D Glasses

Anaglyph glasses have special lenses that filter a specific RGB color channel (see figure 3.7 for an illustration). These glasses are used to separate images that are on the same color channel as the lens. The anaglyph glasses used in the experiments contain lens filters for the red and blue RGB channels. The desktop experiment relies on these glasses to separate overlapping red and blue channel images so that the participant can only detect one image on each eye and successfully produce the CFS effect. The stimuli are always presented using the blue RGB channel to the dominant eye. The lens of the glasses cannot be changed, so in the cases where the dominant eye is on the opposite side of the blue lens, for lack of a better solution, the glasses are used upside down in order to match the participant's dominant eye correctly.



Figure 3.7: Red-Blue Anaglyph 3D Glasses. Image retrieved from here⁸.

It is noteworthy that stimuli can sometimes be marginally noticed in the red lens due to the color blending. To make matters worse, changing the stimuli to the red lens does not solve the problem, as the masks are much more noticeable for occupying more space in the frame and being continuously changed. Although this solution has been proved successful in other studies, it is far from optimal. VR makes for a better case for providing the possibility to use natural colors, considering the mirror apparatus can be unattainable.

⁷<https://www.dell.com/support/home/en-ae/product-support/product/dell-p2212h/overview>
[last accessed: 28/02/2021]

⁸<https://www.amazon.com/-/es/Visoína-película-formato-Anaglyph-piezas/dp/B0848VWL3B>
[last accessed: 28/02/2021]

Mirror Apparatus

The mirror apparatus allows a binocular visualization of a monitor’s reflected images. A participant could view the monitor’s separate images through this device, allowing experiments without the techniques used to implement binocular rivalry using just the desktop display. Although this device is expensive and unavailable in the laboratory to be used for the experiments, the application system supports it. The experiment stimuli are displayed in two separate frames: one frame presents the stimulus, while the other presents the CFS effect. The mirror apparatus reflects each frame to the corresponding eye, as shown in figure 3.8. This support was developed as a proof-of-concept for future implementations of experiments that might need to use such devices.

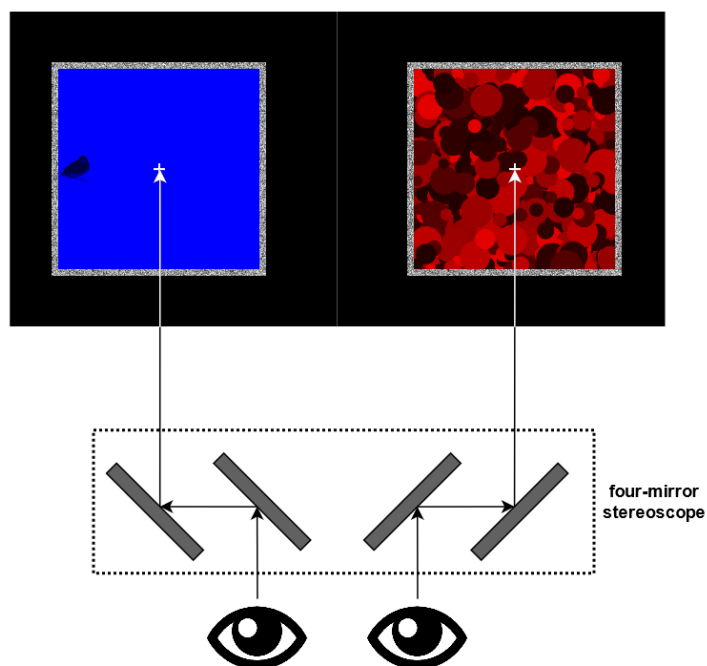


Figure 3.8: Representation of a mirror apparatus as a four-mirror stereoscope.

Head-Chin Rest

All the experimental setups for the bCFS paradigm (and, overall, whenever visual stimuli are presented) require that the participant’s head is in a fixed position relative to the stimuli, which is kept throughout the experiment. While this is not required for the VR goggles, an HMD, action needs to be taken to ensure it for the other two desktop monitor setups. This fixed positioning is accomplished using a head-chin rest (see figure 3.9 for an illustrative example). This equipment is expensive and can be uncomfortable to use during experiments, as the participant is constrained to a fixed position for a prolonged time.



Figure 3.9: Example of a head-chin rest used to fix the participant’s head position for experiments. Image retrieved from here⁹.

3.5 CHAPTER CONCLUSIONS

This chapter described the experiment protocol with detail designed in collaboration with the Department of Education and Psychology, to conduct experiments that allow assessing VR’s technology viability as a suitable alternative for threat processing experiments.

The protocol was designed for an experiment using the bCFS experimental paradigm as an example demonstration to compare VR and standard methods. It was described the specific equipment that was available to conduct the experiments, which elicits a set of requirements to accomplish. In this regard, it was established a list of requirements that guides the application system development to satisfy them accordingly.

The next chapter will detail the software platform design and development that meets the requirements established.

⁹<https://lafayetteevaluation.com/products/head-chin-rest> [last accessed: 28/02/2021]

Experiment Software Platform

The current chapter aims to describe the application system design and development solution to support the experiment protocol, considering the requirements identified in the previous chapter. It is presented a conceptual vision of the different blocks that compose the system, followed by some details regarding its development and features that fulfill the requirements proposed.

4.1 CONCEPTUAL SOFTWARE ARCHITECTURE

The application developed for this dissertation was driven by the experiment protocol requirements listed previously in chapter 3. To summarize, the application is required to support VR technology, and standard technology methods, for experiment paradigms based on CFS. It should process a configuration file to set up experiments and register experiment data in a simple data format. The objective is to build an application that serves as a foundation to implement other experiment paradigms that benefit from the binocular properties of VR technology, such as the CFS paradigm. To do this, the conceptual architecture for the application was designed to implement different experiments based on the presentation of visual stimuli while fulfilling all the established requirements for the particular experiment protocol planned for this project.

As can be seen in figure 4.1, the application architecture contains three principal components. A *Configuration Parser* reads and processes the configuration file data, providing that information to the other system modules. A *Pre-Task Manager* reads the information regarding the experimental paradigm and platform specified in the configuration data and loads the corresponding experiment scene that supports those specifications. Finally, an *Experiment Engine* performs all the actions regarding the experiment set up, execution, and data collection. These components will be discussed in more detail in the following sections.

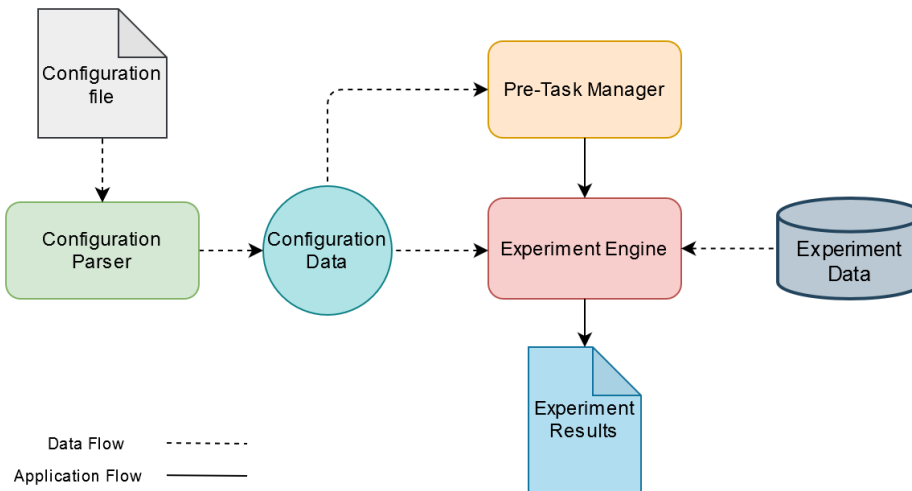


Figure 4.1: Diagram of the application's conceptual architecture.

4.1.1 Configuration Parser

The *Configuration Parser* component is responsible for processing the configuration data provided in the configuration file to simplify access by other system components (see figure 4.2 for an illustration). A *Data Parser* module reads the configuration file and processes all the information to create a single instance (object) through deserialization¹. This process occurs when other components request access to the configuration data object if it has not been processed already. The components of the system can then obtain the configuration details directly.

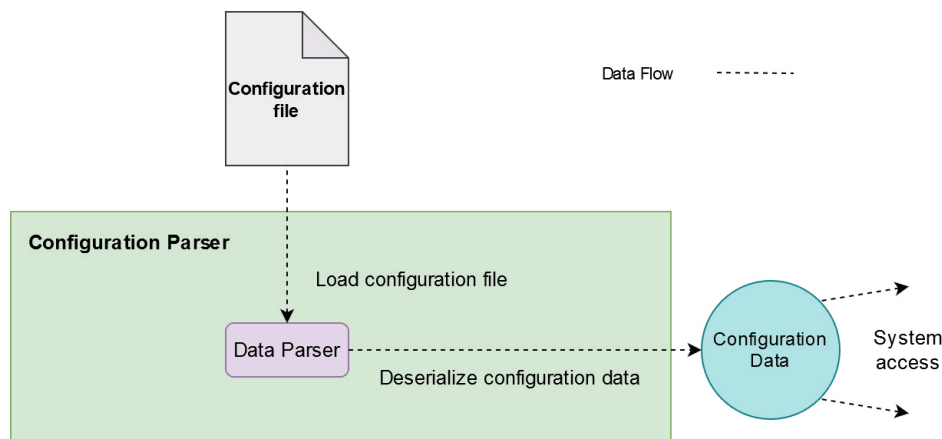


Figure 4.2: Configuration Parser component diagram. A Data Parser module loads the configuration file and deserializes it to a configuration data object, accessible by other components.

¹For more details on serialization, and for the sake of brevity, in this document, the reader is forwarded to Microsoft's documentation here (<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/serialization/>) [last accessed: 02/03/2021].

Processing the configuration file is required to simplify its access by the system. Files stored in persistent storage, such as computer hard disk drives, cannot be directly understood by computer language. Computer systems need to deserialize the file's data, converting it to a data object of the same state that the system can understand. When the system needs to access the configuration data, it only needs to access the data object created instead of reading and processing it every time.

4.1.2 Pre-Task Manager

The *Pre-Task Manager* handles the initial section of the application before the experiment trial section starts (an illustration for this component can be seen in figure 4.3). At the beginning of the experiment, the participant is shown descriptive pages about the experiment, customized through the configuration file. A *Page Manager* module reads the configuration data to fill the pages with the corresponding text information. After the introductory pages, the Pre-Task Manager starts preparing the information to start the experiment. The Page Manager loads a page to insert relevant participant information, such as age, gender, and dominant eye, where a *Participant Data Manager* validates these values to proceed. Finally, an *Experiment Loader* module reads the specified platform in the configuration file, loading the corresponding scene that supports the experiment in that platform.

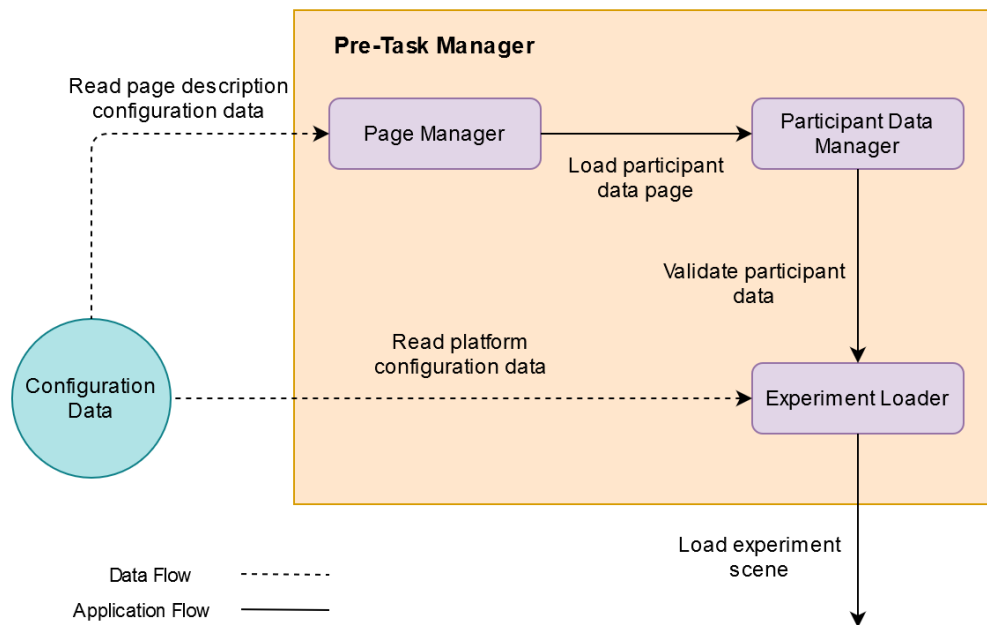


Figure 4.3: Pre-Task Manager component diagram. This component precedes the experiment tasks and introduces the participant to the experiment, collecting relevant participant information for the experiment.

The dominant eye entered in the participant information page directs the experiment objects' set up to display the stimuli to the corresponding dominant eye. All participant information is stored at the end of the experiment, along with the experiment results.

4.1.3 Experiment Engine

The *Experiment Engine* is the core component of the application, controlling all the experiment's details and performance (see figure 4.4 for an illustrative diagram). This component starts setting up the experiment scene by preparing the objects that are displayed to the participant. An *Experiment Assembler* module reads the configuration data to set up the specified sizes of scene objects and obtain information regarding the folder locations for stimuli to be used in the experiments. This module then loads the images from local storage to create the experiment and training sets. These sets are assembled by another module (Experiment Manager, which will be explained later). After the experiment scene objects are ready, their positions can then be calibrated, if necessary, using an *Object Position Calibrator* module. This process is more useful using platforms that separate the objects to each eye directly, i.e., in VR, still being possible on all supported experiment platforms.

Nevertheless, the Object Position Calibrator reads inputs from the controllers (or keyboard), moving the objects in the display to better accommodate the images to the participants' IPD. While the Vive headset contains an integrated IPD adjuster, it may also be necessary to the image distances in the application. After setting up the experiment and calibrating the objects, the experiment training and evaluating session can start.

The *Experiment Executer* module controls the experiment execution, respecting the experimental paradigm specified in the configuration file. The module obtains each trial's phase duration for stimuli control and trial set repetitions, reading the configuration data. This module listens for participant input during the trials and uses an Experiment Manager to evaluate that response.

The *Experiment Manager* is a module that identifies the experimental paradigm specified in the configuration data to perform distinct operations to satisfy it. As previously mentioned, the Experiment Assembler provides the stimuli images to the Experiment Manager to create a set containing the stimuli with all their position iterations for the experiment and an identifier name to distinguish each iteration in the results. At the start of each trial, the Experiment Executer calls the Experiment Manager to update the trial with the next stimulus to be presented and its associated position. The Experiment Manager is also responsible for evaluating the participant's trial response by comparing the given response with the experiment's defined inputs and creating a trial result to be registered. Having a separate module to handle paradigm-specific operations streamlines the implementation for different experiments.

Finally, when the experiment is complete, the results are stored in a file for analysis. A *Data Manager* module collects the trials' results, given by the Experiment Manager, throughout the experiment. It registers the participant's information, entered on the participant data page at the experiment's pre-task, and the display platform and experimental paradigm used for the experiment. The Data Manager also records the start, duration, and ending times for each section (pre-task, training, and experiment times). The module saves the experiment results in the location specified by the configuration file.

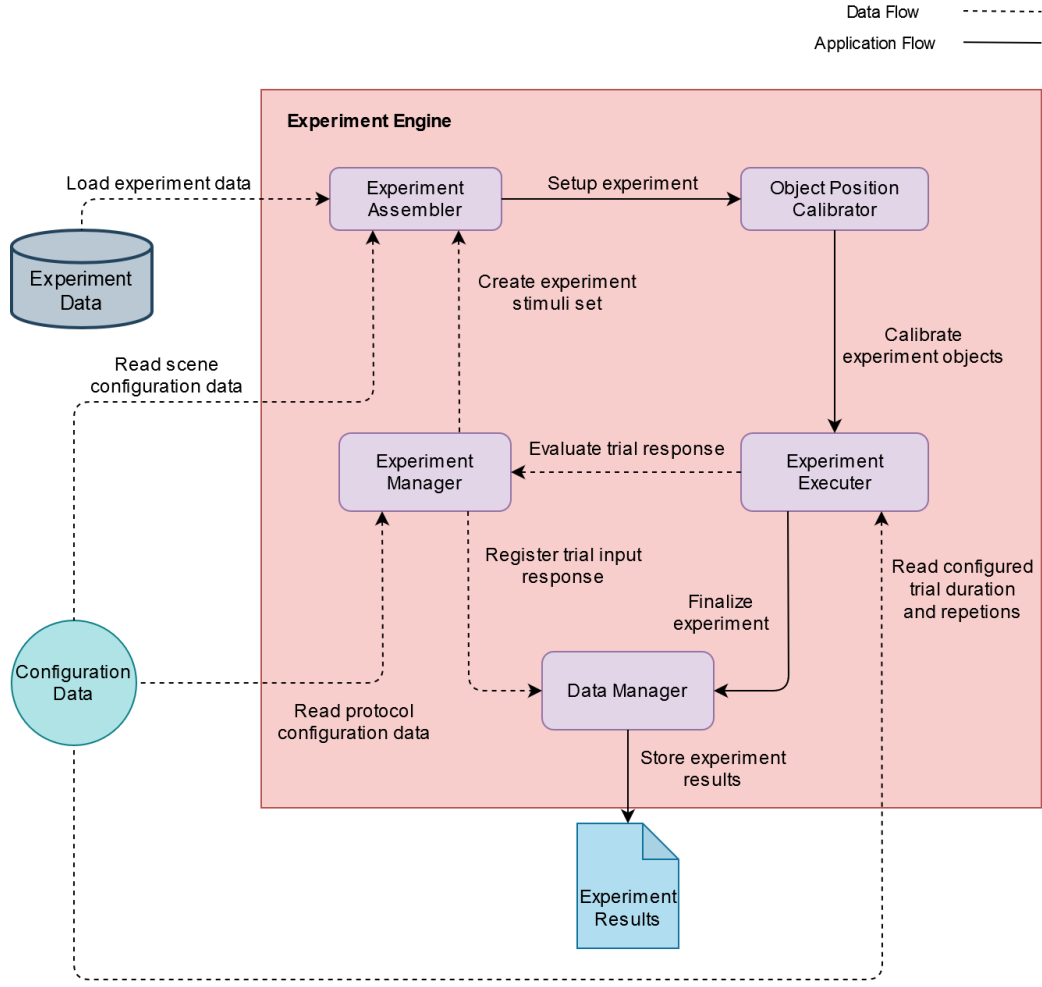


Figure 4.4: Experiment Engine component diagram. This component takes care of the experiment setup, execution, and data collection.

Experiment Manager Role

It is important to note that designing a specific experiment entails implementing a specific Experiment Manager variant. For example, for this dissertation's experiment, an Experiment Manager was developed to handle the "bCFS Fovea Periphery" experimental paradigm and perform the required actions. In an effort to support different experimental paradigms, two other variants that perform generic actions were implemented, allowing an experimental data collection that does not require specific actions or behaviours, such as a particular reaction input or evaluation of trial response. The Experiment Manager's variant used for the experiment is directly tied to the experimental paradigm defined in the configuration file.

If an experiment requires unique behavior for an experimental paradigm that is not supported, the only implementation necessary would be a variation of the Experiment Manager to support those conditions. Since the other Experiment Engine's components already take care of the experiment setup, execution, and data collection, the implementation would be set.

4.2 APPLICATION DEVELOPMENT WITH UNITY

Unity was the tool selected to develop the computational system, after successful testing on its support for core aspects of the requirements previously listed in chapter 3, and its support for multiple platforms, such as VR, most notably. Having selected Unity, the UXF framework was explored for its potential to assist in the development of the system. However, despite being a tool that can be very useful in the future, the current work requirements did not justify the investment in implementing this tool, as most of its essential features would not be used. The application only explored 2D aspects, considering the requirements established.

The next section will briefly discuss Unity's development framework² to get some insight into some of the terms mentioned and then elaborate on the application's development.

4.2.1 Unity's Development Framework

A Unity engine application is the collection of one or more scenes. Everything that runs in the application exists in a scene. Each scene can be thought of as a section in the application to perform a specific task. Virtually everything in a scene is composed of *GameObjects* that serve different purposes, depending on their type.

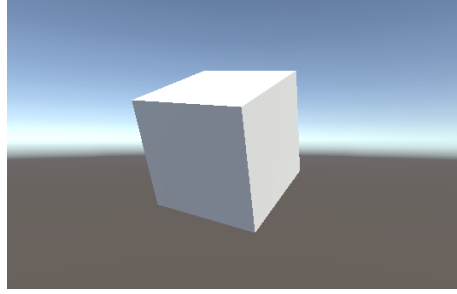
A *GameObject* is the base type for all objects in a Unity scene. Every *GameObject* is composed of components that add functionalities to them. For example, a text displayed in a scene is simply a *GameObject* with a "Text" component attached. The default component of a *GameObject* is the *Transform*, which handles the *GameObjects*' position, rotation, and scale in the virtual environment. The combination with components can make *GameObjects* act as lights, 3D/2D objects, shadows, among others. The scene is rendered into the display by a "Camera", which is also a *GameObject*.

Behaviour can be implemented by developing a C# script and attaching it to a *GameObject*. These scripts must derive from the *MonoBehaviour* base class, which provides access to Unity's event functions³, such as "Awake", "Start", and "Update", and other features necessary to create a Unity application. As a demonstration of these concepts, figure 4.5 illustrates an example of the components that form a cube *GameObject*, with a script attached that makes the cube rotate given a speed.

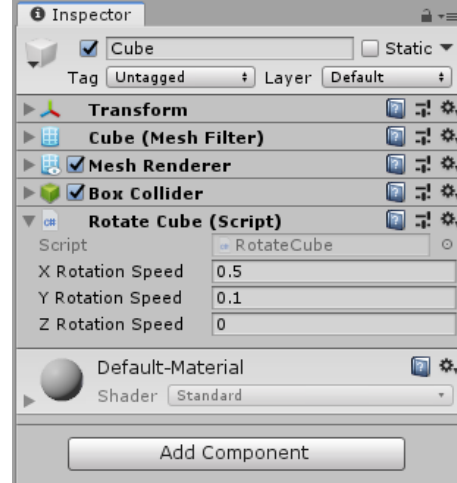
A component that is going to be mentioned frequently throughout this section is *SpriteRenderer*. As the name suggests, this component renders a *Sprite* into view. A *Sprite* is a 2D graphic object that represents an image in a virtual environment. The *Transform* and the *SpriteRenderer* components contain properties to manage the *Sprite*'s presentation, most notably the size, position, color, and material (see figure 4.6 for an illustration on those properties inside Unity), which will be proved useful further down in this section.

²To get a better insight on the full features of Unity, the reader is forwarded to the corresponding online resources (<https://docs.microsoft.com/en-us/archive/msdn-magazine/2014/august/unity-developing-your-first-game-with-unity-and-csharp>) [last accessed: 02/03/2021]

³For more details, see Unity's event function order of execution documentation (<https://docs.unity3d.com/Manual/ExecutionOrder.html>) [last accessed: 02/03/2021]



(a) Unity default GameObject cube.



(b) Unity cube inspector window.

Figure 4.5: Unity GameObject representing a cube. (a) shows the cube representation in the scene; (b) shows the Unity inspector window containing all the components representing the GameObject as a cube. The "Rotate Cube" script attached adds behaviour to the cube, rotating it in its axis, given a rotation speed.

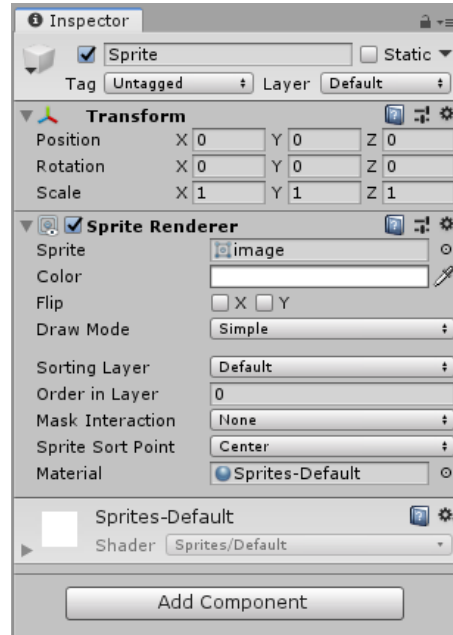


Figure 4.6: Unity Sprite GameObject inspector window. The Transform component can control the Sprites's position and size, while the SpriteRenderer component controls the image texture, color, and material.

4.2.2 Experiment Engine Scripts Hierarchy

The main goal for the considered experimental protocol is to implement the bCFS paradigm. As explained previously (see section 4.1.3), the experiments are performed by the Experiment Engine component. To this end, a specific Experiment Engine type had to be created that supports visual stimuli presentation experiments based on the CFS technique. Each platform supported is implemented in different Unity scenes, containing different versions of

an Experiment Engine script to support the execution of experiments for that platform. The Experiment Engine's core functionality is the same for all platforms, the only difference being the setup of the scene, which requires different procedures.

The Experiment Engine scripts inherit the MonoBehaviour base class to interact with the scene's GameObjects and use Unity event functions. Using inheritance, the "Engine" scripts for each platform derive from preceding base scripts that implement shared system functionalities. With this hierarchy, each "Engine" class implementation focuses on specific functionalities instead of implementing everything in a single script. Any alterations done in a base script are immediately applied to the derived scripts, making maintenance of the scripts more manageable. Figure 4.7 illustrates the hierarchy for all supported platform Experiment Engine scripts.

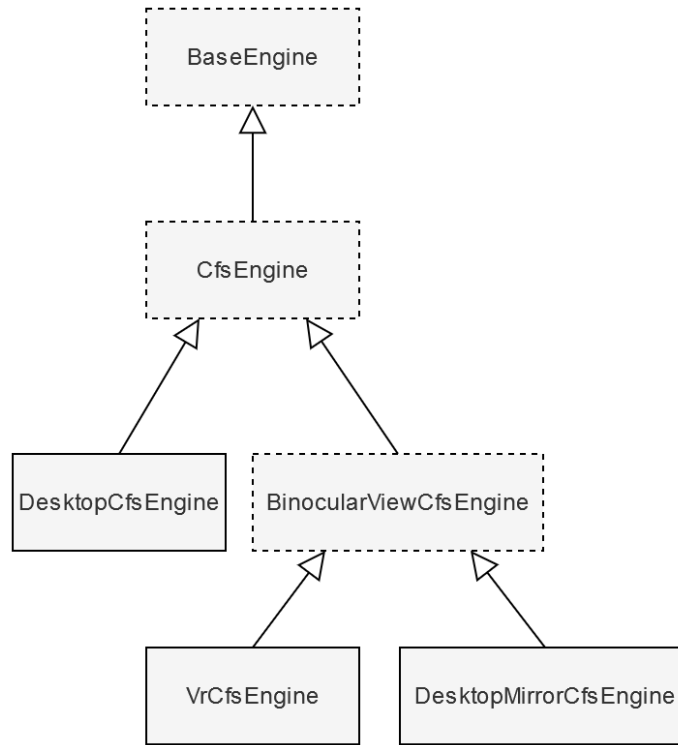


Figure 4.7: Hierarchy of Experiment Engine scripts of all supported platforms. Dashed lines are scripts that implement shared functionalities, while the solid lines represent the final "Engine" script versions.

The "BaseEngine" implements basic functionalities that any experiment type should use, such as instantiating a Data Manager and getting a reference to the configuration data. The "CfsEngine" implements functionalities specific to the CFS paradigm and the setup of some scene objects. This script implements the Experiment Executer that performs the CFS paradigm used by all platforms. It also instantiates an Experiment Manager variant for the type of experimental paradigm specified in the configuration file. For the scene setup, the "CfsEngine" sets the stimuli to be displayed to the specified dominant eye and sets up the

stimuli objects' sizes to the configured amount.

The final scripts in the hierarchy perform specific actions to the platform they will manage. These actions are mostly tied to the way the scene is set up. The "BinocularView" scripts have to manage two separate frames since the platform they control needs to display an image to each eye, while the "Desktop" script only needs to manage one. The actions performed are for the scaling, positioning, and calibration of the scene objects.

4.2.3 Stimuli Presentation

The stimuli used in experiments are images and are presented into the experiment frame. The frame presented in the experiment scene comprises a group of SpriteRenderers, i.e., the white noise, fixation cross, frame background, stimuli, and the Mondrian pattern CFS masks, as shown in figure 4.8. The binocular platforms, i.e., VR and desktop with mirror apparatus, have two separate frames: one frame displays the stimulus to one eye, while the other produces the CFS effect to the other eye (see figure 4.9 for an illustration). All frame objects are manipulated to the same extent, only the masks needing special handling for the desktop experiment presentation.

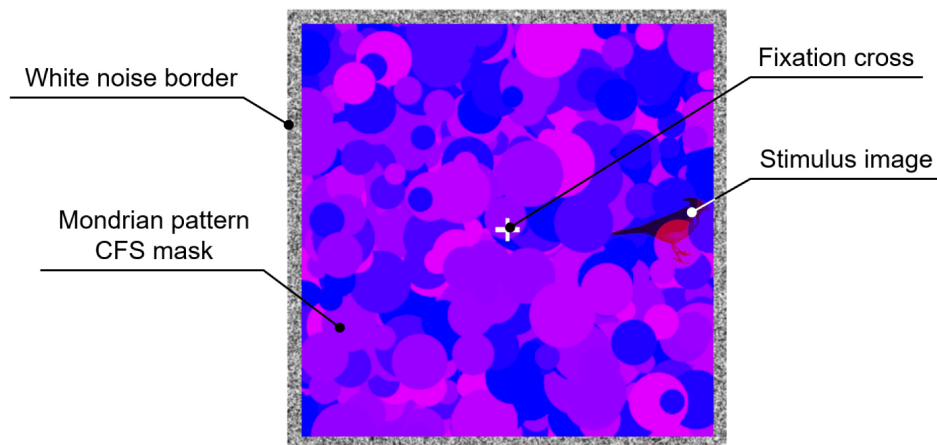


Figure 4.8: Desktop platform experiment frame example. This figure shows the blending process to display both the stimuli and CFS masks (at full opacity). The viewable objects that compose the frame are labeled.

All frame objects have their size manipulated to fulfill the specified size in the configuration file. The frame background, stimuli, and masks also have their color changed to the red or blue RGB channels to comply with the specifications imposed using the "Red-Blue anaglyph glasses". Otherwise, the color could be set to use the natural colors of the images. However, for the desktop experiment, the masks require changes on their material. In Unity, SpriteRenderers are organized on layers, with the one sitting in the highest being shown primarily. The default material of Sprites are opaque and hide the Sprites in the lower layers.

For the desktop experiment, only one frame is shown, and the masks sit on the layer above the stimuli images. During the first step of the trial, raising the stimulus opacity into view, the mask completely hides it and it cannot be seen, which hinders the CFS paradigm. Using

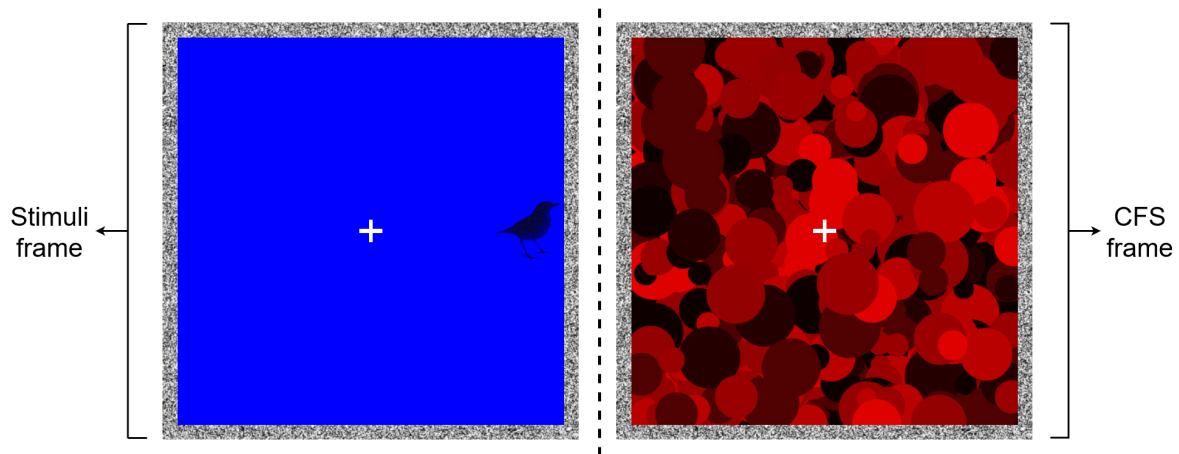
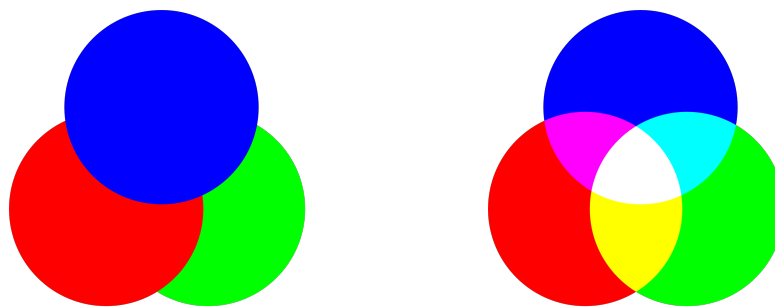


Figure 4.9: Experiment frame example for the binocular platforms (VR and desktop with mirror apparatus). These platforms display an experiment frame to each eye: one frame presents the stimulus, while the other produces the CFS effect. The dashed line represents the binocular isolation, done by the display platform, between each frame.

"Additive Blending" can solve this issue (see figure 4.10 for an illustration) by allowing the masks to blend with the Sprites on the layers below. This blending effect can be produced by modifying the mask's SpriteRenderer material's Shader file⁴. The mask SpriteRenderer with the additive blending material blends itself with the stimulus Sprite in the layer below, which can then be pickup up by the anaglyph glasses so the participant can see the stimulus and mask separately. The VR and desktop with mirrors experiment scenes do not require this manipulation of the Sprite's material since two separate frames are displayed directly to each eye. One frame displays the stimulus, and the other the CFS masks.



(a) Normal presentation (no blending).

(b) Additive Blending.

Figure 4.10: Example demonstration of overlapping images with and without Additive Blending.

The stimuli presentation procedure is identical for all platforms, independent of the special treatment discussed above. The presentation is done in the Experiment Engine, using the Experiment Executer and Experiment Manager modules. The stimuli images are read from an accessible storage directory, whose path is specified in the configuration file, containing only

⁴For a better reference of this modification, see the "Details" section in this Unity documentation page (<https://docs.unity3d.com/2019.3/Documentation/Manual/SL-Blend.html>) [last accessed: 02/03/2021].

the images intended for the experiment. The same process is used for the Mondrian masks, which are loaded into memory, so the mask SpriteRenderer uses them to rapidly change them (10 frames per second) and create the CFS effect. The experiment set is created, using the experiment's specific Experiment Manager, containing the generated Sprites and their position information for each trial. The Experiment Executer reads the next stimulus to be presented for the trial, and uses the Experiment Manager to interpret the position information. The stimulus object is placed accordingly, proceeding with the trial procedure. This process is repeated until the end of the experiment.

4.2.4 Scene Objects Scaling Methodology

Scene object setup is guided through the settings included in the configuration file. These settings control the object's size and position in the scene. In Experimental Psychology, the size of the objects observed in experiments is usually measured using visual angles, focusing on the perceived size of a stimulus instead of its real size. If the object is moved further away, the visual angle decreases and is perceived as smaller⁵. The distance from the observer's eyes to the object influences its visual angle calculation.

The visual angle calculation can be decomposed into a simple trigonometry task, using a right triangle, as illustrated in figure 4.11. Knowing that the tangent is given by the opposite and adjacent sides' ratio, the angle (θ) is obtained using the arctangent (arctan) function as in the following equation:

$$\theta = \arctan\left(\frac{\textit{Opposite}}{\textit{Adjacent}}\right) \quad (4.1)$$

Since the visual angle is influenced by the size of the stimulus (opposite side) and its distance to the eyes (adjacent side), the resulting visual angle can be obtained using this method.

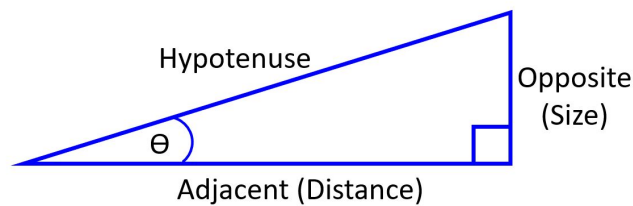


Figure 4.11: Right Triangle of visual angle. The tangent value of the angle is defined to be the ratio of the opposite side length (size of the stimulus) to the adjacent side length (distance from the eye to the stimulus). Image retrieved from this page⁶.

Although the opposite side of the right triangle represents the size of the stimulus, it only represents half of its length, and thus half of the visual angle, as can be seen in figure 4.12.

⁵For more detailed information and an interactive example, visit this page (<https://elvers.us/perception/visualAngle/va.html>) [last accessed: 02/03/2021]

⁶<https://www.sr-research.com/eye-tracking-blog/background/visual-angle/> [last accessed: 02/03/2021]

Two mirrored triangles can be used to cover the distance from the stimulus' center to each side, which doubles the angle value. The visual angle is calculated using the arctan equation 4.1 by doubling the angle obtained using the distance and half the stimulus's size. The following equation gives the visual angle (θ):

$$\theta = 2 \times \arctan\left(\frac{Size/2}{Distance}\right) \quad (4.2)$$

The "Size" and "Distance" have to be in the same measurement unit. The θ angle unit is radians, which can then be converted to degrees by multiplying $\frac{180^\circ}{\pi}$ (approximately 57.296).

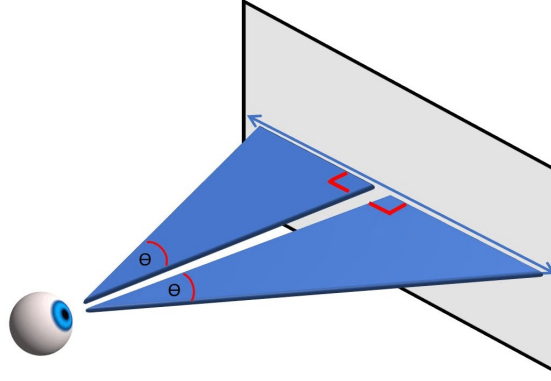


Figure 4.12: Two right triangles complement the length of an object. A right triangle only completes the length from the center to one of the sides of the object. A mirrored right triangle is added to complete the length of the other half. Image retrieved from this page⁶.

Presenting the scene objects with the correct visual angle size requires more calculations to translate this value into the number of pixels the display should present. This number is obtained by calculating the visual angle of one pixel (angle per pixel) of the display and dividing the object's visual angle size by this value. The display's angle per pixel is obtained by dividing its visual angle by the resolution. Using equation 4.2, where "Size" is the display's size, and dividing by the display's resolution, its angle per pixel is found:

$$AnglePerPixel = 2 \times \frac{\arctan\left(\frac{Size/2}{Distance}\right)}{Resolution} \quad (4.3)$$

Finally, the object's pixel size is obtained with:

$$ObjectPixelSize = \frac{ObjectVisualAngleSize}{AnglePerPixel} \quad (4.4)$$

For positioning the object in the display, the pixel values are obtained similarly. Only in this case, the values are interpreted as the pixel distance from the screen's center.

It is important to note that resizing scene objects is done using only one of its dimensions as a reference, specified in the configuration file. Since the scene objects are images, resizing them horizontally and vertically could disrupt their aspect ratio and appear deformed (see figure 4.13 for an illustrative example). Resizing only one dimension of the image prompts the system to resize the other dimension accordingly, maintaining the aspect ratio. For an object

to have the same width and height, such as the frame's noise border, the image should have an aspect ratio of 1. The system calculates the object's corresponding pixel amount using each pixel's angle of the referenced dimension.

The display size specifications should be defined with the most accuracy possible to provide precise pixel amounts for the object's size. If the specifications are inaccurate, the object could be represented with a greater or smaller size than the effective one due to the angle per pixel being influenced by the incorrect display size. Using the angle per pixel angle of a given dimension also helps if the display used does not have perfectly "squared" pixels. If an overall angle per pixel value of the display is used, it could present differing sizes for the same visual angle when resizing an object on the width and another on the height. For instance, if resizing an object with 2° of visual angle on a display without "squared" pixels, its apparent width size could represent the effective 2° , when resizing with the width as the reference, while when resizing it on the height, it could present a different size than the effective 2° it should have. Using the angle per pixel of a reference dimension to resize an object ensures a higher standard of reliability, as this approach accounts for the factors mentioned above.

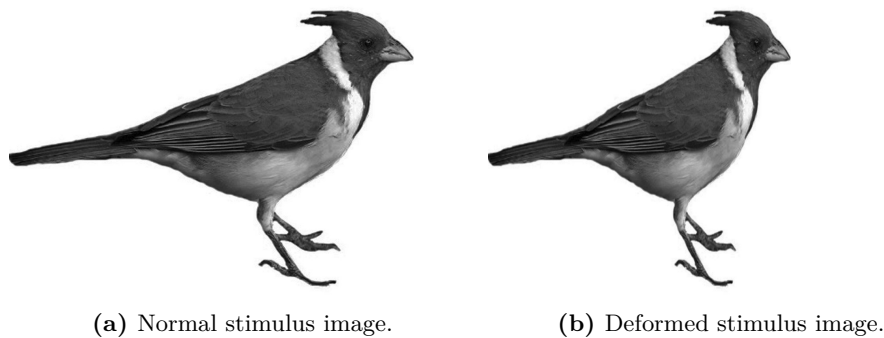


Figure 4.13: Demonstration of the image deformation of a resized stimulus in both dimensions. (a) presents a stimulus image resized using only one dimension as a reference; (b) shows a stimulus resized to have the same width and height size.

Scene object setup methodology is identical on all platforms by using the configured scene object size values, display specifications, and the methods described above. For the desktop experiments, the display specifications are detailed with all the information needed. However, for the VR experiments, not all specifications needed about the HMD are listed (equipment specifications can be found in section 3.4).

Specifications for the Vive do not explicitly include the headset's display dimensions (height and width) and lack information about the distance from the eyes to the screen. Therefore these values had to be estimated. The width and height were calculated using the diagonal size provided and the aspect ratio of the screen. The distance from the eyes to the display was estimated using the display's known FOV (110°) and the calculated display size, substituting these values in equation 4.2 and solving for "Distance". The values obtained were 61.1mm width, 67.9mm height, and a distance of approximately 51.2mm.

This methodology for VR display object scaling seems plausible, although its effectiveness is debatable due to the fisheye lens distortion preventing the exact determination of the

stimuli pixel size. However, this methodology aims to standardize a system that performs scene object scaling on all platforms. Display dimensions and eye-to-display distance can be specified in the configuration file and later configured when obtaining more accurate information that validates the effective screen dimensions and eye-to-display distance of a VR HMD. Appendix A contains an in-depth review of the metrics estimated for the VR headset and discusses some of the challenges of using this scaling method using the VR HMD. It also explains the methodology adopted to achieve the correct pixel representation of stimuli to the display using Unity.

4.2.5 Data Management

The data generated throughout the application lifecycle is collected using a Data Manager. The experiment data is stored in a Comma-Separated Values (CSV) file format at the end of the experiment. The Data Manager assembles the experiment data to be stored in a single CSV line of the experiment results file. By storing all the participant's data in a single line enables that the data for additional participants to be appended to the same file and, later, opened with software tools such as Excel⁷ or SPSS⁸, commonly used for the analysis.

The data collected into the results file ranges from relevant information about the participant: age, gender, and dominant eye; a unique alphanumeric ID to distinguish each result, the platform and experimental paradigm used in that experiment, timestamps for the start, end, and elapsed times of each session of the experiment: introductions, training, and experiment; and all the data of each trial of the training and experiment sessions.

*CsvHelper*⁹, a community-made .Net library for reading and writing CSV files, was used to streamline the manipulation and handling of experiment results in one line and into the CSV files.

4.3 CONFIGURATION FILE

The configuration file contains all the parameters necessary to configure and set up an experiment. It is written in eXtensible Markup Language (XML) as it promotes an organized structure for each specific parameter. The information regarding the configuration file's properties and parameters is discussed in more detail in this section.

4.3.1 XML File Structure & Properties

XML is a markup language that defines rules to encode documents in a readable format for humans and systems. *Elements* and *attributes* compose an XML document (see listing 4.1 for an example of an XML document file structure). Elements mark up a section of the XML document and hold a value or nest other elements, while attributes add descriptive information to an element. The XML document starts with a root element that encloses all elements that form the document's structure.

⁷Microsoft Excel main page (<https://www.microsoft.com/pt-pt/microsoft-365/excel>).

⁸IBM SPSS main page (<https://www.ibm.com/analytics/spss-statistics-software>).

⁹CsvHelper main page (<https://joshclose.github.io/CsvHelper/>).

```
<root_element>
  <element attribute="value">
    <nested_element>Value</nested_element>
    <nested_element other_attribute="other_value">Value</nested_element>
  </element>
  <element>Value</element>
</root_element>
```

Listing 4.1: Example of an XML document structure.

The information contained in an XML document can be parsed to a system through deserialization. Deserialization reconstructs the XML document information into a single machine-readable object representing that information and maintaining its structure. For the application, the object created represents the XML configuration data that can be accessed by the system to set up the experiment.

The XML format was chosen to create the configuration file due to its high verbosity and structure. Compared to other alternatives, it provides more readability and better appeal for researchers unfamiliar with this concept. This assumes a particular relevance for the context of this work since it enables researchers at the Psychology lab to more easily customize configurations, e.g., to define the dimensions of a new desktop monitor to use or add more stimuli to the list. XML has the cost of creating larger files, though it is a negligible factor for its purpose.

4.3.2 Configuration Parameters

The experiment set up is done by setting different values to the parameters in the configuration file. These parameters were organized to generalize their use so that experiments with different experimental paradigms can use the same parameter set. Listing 4.2 presents a minimal configuration file that complements the proposed experimental paradigm, labeling every parameter to facilitate identification. A "configurations" element nests all parameters, holding all the information of the configuration file. This section describes the primary function of each parameter and its settings. Appendix B presents an extended version of this section, providing a better insight into using the configuration file to set up experiments.

Experimental Paradigm

The experimental paradigm, labeled with "Experimental Paradigm Setting" in listing 4.2, is specified using the "experimental_paradigm" element. This element contains a unique value that indicates the experimental paradigm to be used, instructing the system to set up and perform the experiment accordingly. The given configuration example shows the experimental paradigm for this dissertation's experiment, with the codename *bCFS_F_P*. The "bCFS" portion appoints to an experiment that uses the bCFS paradigm. The "F_P" portion indicates Fovea and Periphery, respectively, which are the positions for the stimuli in the experiment frame, creating the visual angles intended to be studied using this method. Another two

codenames were implemented to add support for other experimental paradigm settings, which are detailed in appendix B.

```

<configurations>

  <!-- ## Experimental Paradigm Setting ## -->
  <experimental_paradigm>bCFS_F_P</experimental_paradigm>

  <!-- ## Display System Settings ## -->
  <display_system platform="Desktop">
    <display pixel_width="1920" pixel_height="1080" width="476" height="267"/>
    <distance_to_screen>500</distance_to_screen>
  </display_system>

  <!-- ## Data Settings ## -->
  <data participant_id_length="8">
    <csv csv_delimiter=";">
      <save_path base_folder="Desktop">Experiment Results</save_path>
      <filename>Experiment_Results.csv</filename>
    </csv>
  </data>

  <!-- ## Experiment Settings ## -->
  <experiment_settings color="Red_Blue" training_set_size="20" trial_repetitions="3">
    <frame size="16" size_unit="Degree" reference_dimension="Height">
      <stimuli size="2" size_unit="Degree" reference_dimension="Height" sort_type="Random">
        <positions distance_unit="Degree">
          <position horizontal_distance="1" vertical_distance="0" label="fovea"/>
          <position horizontal_distance="6" vertical_distance="0" label="periphery"/>
        </positions>
      </stimuli>
      <masks cfs_step="100" size="15" size_unit="Degree" reference_dimension="Height"/>
    </frame>
    <duration raise_stimulus="1100" lower_mask="4000" stimulus_exposed="3000"/>
    <experiment_data_folders>
      <experiment_folder>Experiment Stimuli</experiment_folder>
      <training_folder>Training Stimuli</training_folder>
      <mask_folder>Masks</mask_folder>
    </experiment_data_folders>
  </experiment_settings>

  <!-- ## Pre-Task Pages Settings (content omitted) ## -->
  <pre_task_pages>[...]</pre_task_pages>
</configurations>

```

Listing 4.2: An example of a configuration file that complements the bCFS experimental paradigm proposed. Each parameter is separated and labeled for clarity. The Pre-Task pages parameter is omitted for the sake of brevity.

Display System

The display system parameter is specified with the "display_system" element and contains the settings concerning the experiment's platform. The platform itself, i.e., desktop, desktop with mirrors, or VR, is specified here as a value. Each platform contains a display to present stimuli to the participant. Its dimensions and resolution need to be specified to calculate scene objects' pixel size using visual angles. The participant's view distance to the display

influences stimuli size calculation, which is specified in this parameter. Listing 4.2 shows an example of this parameter for an experiment using the Desktop platform.

Data

The data parameter, specified with the "data" element, contains all the settings necessary for the experiment results CSV file. It allows to specify the CSV delimiter, save path, and filename. There are two ways to specify the save path: combining a base folder, recognized by the operating system¹⁰, e.g., *Desktop*, *MyDocuments*, with a relative path (if needed), or writing the absolute path. If the system recognizes the base folder indicated, it will get its absolute path and join it with the relative path stated. This method makes it easy to relocate the experiment files to different computers or folders without affecting its functioning. Listing 4.2 shows an example of a data parameter using the "base folder and relative path" approach.

The results collected from each participant are identified by a randomly generated unique ID, whose length is also specified in this parameter.

Experiment Settings

The experiment settings parameter is responsible for all the values necessary to prepare the experiment scenery and execution, and it is specified with the "experiment_settings" element. Listing 4.2 presents an example of the settings used to set up the experiment for the experimental protocol designed.

The color used for the experiment can be specified to control the experiment's displayed color filtering. The experiment stimuli can be displayed using the natural colors or with the Red-Blue RGB filtering process through this setting. The number of trial repetitions for the experiment can be specified, and, for the training phase, the training set used can be limited to display only the indicated amount trials.

This parameter allows defining the sizes of the experiment frame, stimuli, and masks while indicating those values' measurement type. The sizes can be interpreted in visual angles, using degree and radians as possible measurements, and in pixels, directly using that value to display on the screen. In the example given, the stimuli size is defined as 2 and interpreted as degree units (2°). The reference dimension is also stated, which instructs the system to scale the stimuli to that size, according to their height or width.

The stimuli are also presented in various positions. Each position stated is defined by a horizontal and vertical distance from the frame's center, using a label to identify them. The positions' distance measurement, which has the same measurement types as discussed previously, is indicated to inform the system how to interpret each position. The stimuli's trial presentation order can be organized randomly, alphanumerically ascending, or descending. The listing 4.2 example demonstrates the positions used for the proposed experimental paradigm's experiments, organizing the stimuli randomly for presentation.

¹⁰This is obtained using a specific enumerator as described here (<https://docs.microsoft.com/en-us/dotnet/api/system.environment.specialfolder?view=netcore-3.1>) [last accessed: 02/03/2021]

The example also demonstrates the duration steps of each trial according to the stimuli presentation procedure of the experimental protocol designed. Each trial phase performs a specific action to be carried out during the amount of time specified. The CFS time step can be set to determine how many times per second the masks are interchanged, creating the CFS effect. The duration and CFS step values are all interpreted as milliseconds. The configuration example shows the CFS step to be 100 milliseconds, changing masks at a rate of 10 frames per second.

The experiment settings also allow setting the path for image folders used in the experiments. This path can either be an absolute path or a relative path. When using a relative path, the application will try to read it from a pre-defined folder under the application data directory. The training and experiment folders should contain stimuli images to present during the training and experiment phase, respectively. The mask folder should include pre-rendered Mondrian pattern images to display and create the CFS effect. Using this approach allows full control to the examiner to add or change stimuli without needing further assistance.

Pre-Task Pages

The pre-task pages parameter allows the system to set a customized text for the descriptive pages that appear before the experiment. The specific page to which the text and description should appear can be specified. Listing 4.3 shows an example of the structure for this parameter. Section 4.4 exemplifies figures on how the text set in this parameter is presented in the interface.

```
<pre_task_pages>
  <page name="Introduction">
    <title>[Introduction Title]</title>
    <description>[Introductory experiment description]</description>
  </page>
  <page name="Participant_Information">
    <title>[Participant Information Title]</title>
    <description>[Participant information instructions]</description>
  </page>
  <page name="Pre_Calibration">
    <title>[Pre-Calibration Task Title]</title>
    <description>[Calibration phase description]</description>
  </page>
  <page name="Pre_Training">
    <title>[Pre-Training Task Title]</title>
    <description>[Training phase description]</description>
  </page>
  <page name="Pre_Experiment">
    <title>[Pre-Experiment Task Title]</title>
    <description>[Experiment phase description]</description>
  </page>
  <page name="Experiment_End">[...]</page>
</pre_task_pages>
```

Listing 4.3: Pre-Task pages parameter. The title and description values are set between square brackets to imply the text that should be set. For the sake of brevity and demonstration, the values are not explicitly set.

4.4 APPLICATION USAGE & USER INTERFACE

The interface design's main focus was for it to be minimalist, without any aspects such as animations that might distract the participants. The application segregates experiments into five phases: pre-task, calibration, training, experiment, and final phases. This section discusses the application's usage and each phase's purpose, in agreement with the experimental protocol designed. Additional detailed information regarding the implemented input controls to use the application can be found in appendix C.

Before starting the application, the experiment should be set up using the configuration file with the parameter values required. Then, the system reads the configuration file at startup to layout the correct scenery for the experiment.

The **Pre-Task phase** is the first event presented by the application. This segment contains a page that describes the experiment to be performed and an area to set experiment relevant information about the participant. The application navigation is done by pressing the button at the bottom of the screen.

The first page introduces the participant to the experiment. It can describe the tasks to be done, display a welcoming message for the participant, and so forth (all descriptive pages are similar, only changing the text content; see the calibration page example in figure 4.15). The text set in the configuration file will appear on the page. The next page is the information area that contains a small form to enter relevant participant information such as age, gender, and dominant eye (see figure 4.14 for an illustration). A smaller text can appear in red, alerting for invalid values or no values inserted and averting continuing the experiment with erroneous or missing participant information.

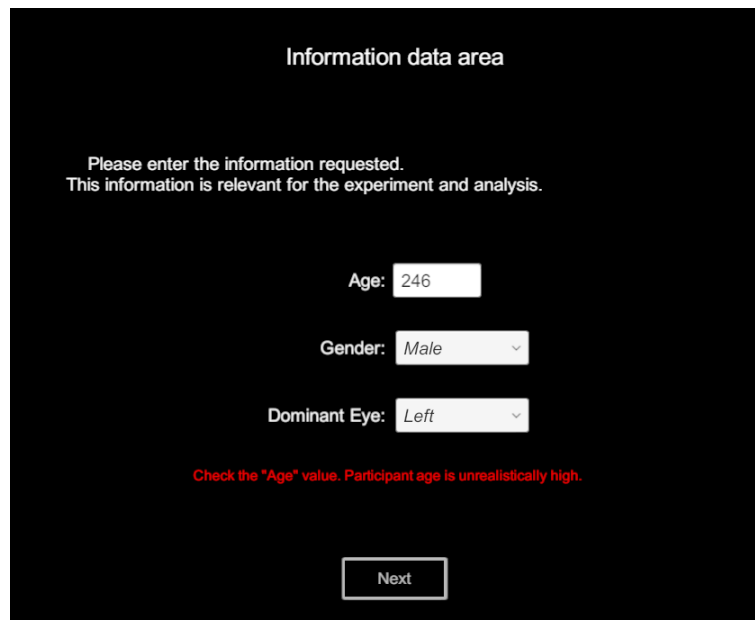


Figure 4.14: Participant information data page example. The information is entered in the input fields to use for analysis. The dominant eye inserted will instruct the application to display the stimuli to that eye. The text displayed can be edited through the configuration file. The displayed page has been cropped and zoomed for readability.

This phase is complete after entering the participant data. The next phases encompass the tasks to be performed by the participant. Each task contains an initial page that describes the task to be performed. Like the first page of the application, each page's text can be set in the configuration file.

The **Calibration phase** provides a chance to calibrate the scene objects' position. The displayed images for the calibration are the same for each eye. It is displayed a fixation cross in the center and a white noise border, having the same size as the frame, specified in the configuration file. The frame can be repositioned both vertically and horizontally using the Vive hand controllers. For binocular view displays, i.e., VR and desktop with mirrors, the horizontal movement is symmetrical between the images (moving the left image to the left makes the right image go to the right and vice-versa). There is the option to move them independently from each other (moving only the left or right image freely) by combining the movement keys with a specific key.

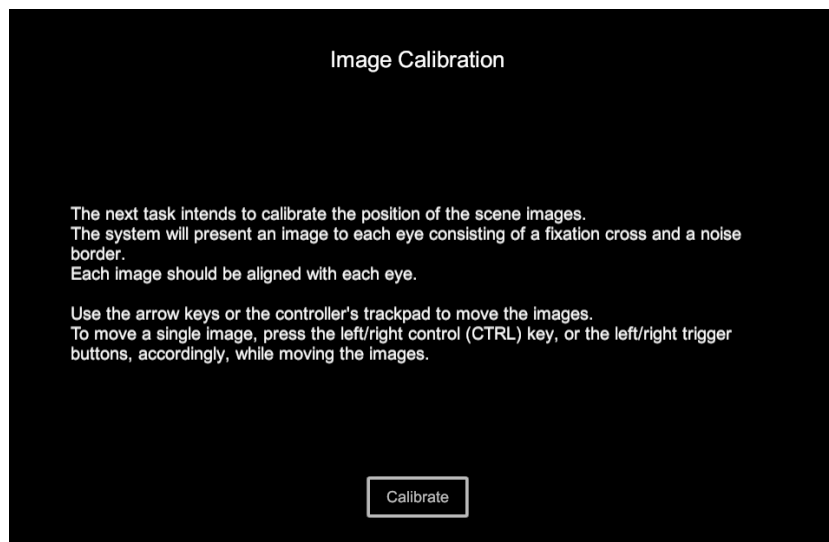
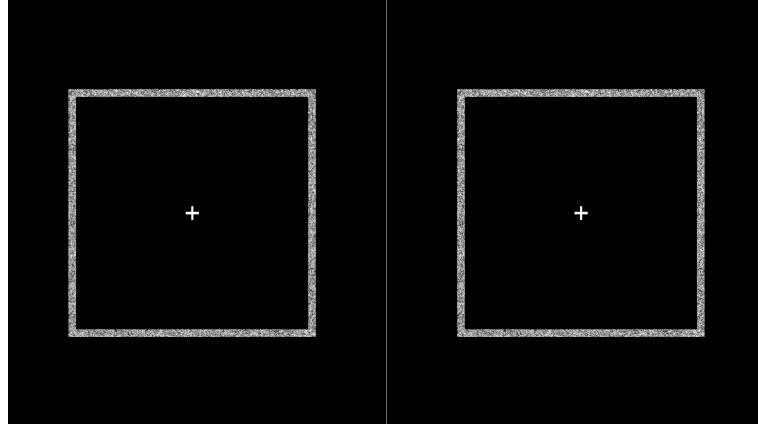


Figure 4.15: Pre-Calibration page example. It appears before the calibration task to describe to the participant details regarding the calibration process. The descriptive pages that precede the tasks have the same layout, only changing the text content. The displayed page has been cropped and zoomed for readability.

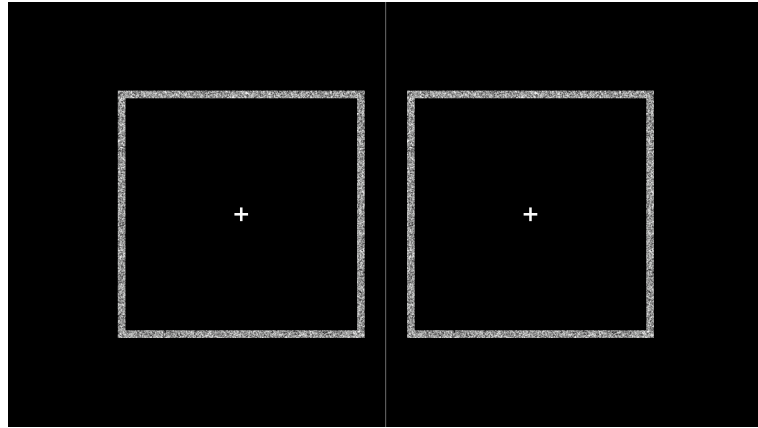
This phase is used on all platform displays but should be more useful for the platforms that project different images directly to each eye, as is the case in VR and desktop with mirrors platforms. Initially, the scene's objects are evenly centered; however, participants may feel that their eyes are forcefully converging to the frame's center because it is not perfectly aligned with their eyes. The frames can then be moved to calibrate their position and align with the participant's eye's center, relaxing eye focus (see figure 4.16 for an illustration of this process).

Since the IPD can vary between participants, this phase is mandatory. After the images are calibrated, or the calibration is not necessary, the images' position can be accepted to move on to the next task.

The **Training phase** is where the participant gets accustomed to the experiment task



(a) Initial image in the calibration phase.



(b) Image calibrated with a narrower distance in-between.

Figure 4.16: Calibration phase in the Desktop with mirrors platform. The images show the positioning of the scene frames before and after calibration. For the lack of a better representation of the VR calibration phase, the desktop with mirrors platform is shown.

and input controls. It simulates the experiment environment in a shorter number of trials, specified in the configuration file. The participant's trial response input is distinguished by the hand controller that responded; a left controller input would mean that the participant perceived the stimulus in the left of the frame and vice-versa. The results obtained in this phase are not considered for analysis but are stored to keep a record for the experiment session. The task is preceded by a page that explains the training phase's purpose and details.

After setting everything up for the experiment and giving the participant the chance to practice, the **Experiment phase** begins. A page describes to the user the experiment task will begin and that the results obtained in this phase will be evaluated.

The participant goes through all the trials of a full-length experiment. The results obtained are stored and saved to a CSV file for analysis.

The **Final phase** displays a simple page to announce the experiment's end, thanking the participant for the participation in the experiment, and a button to exit the application.

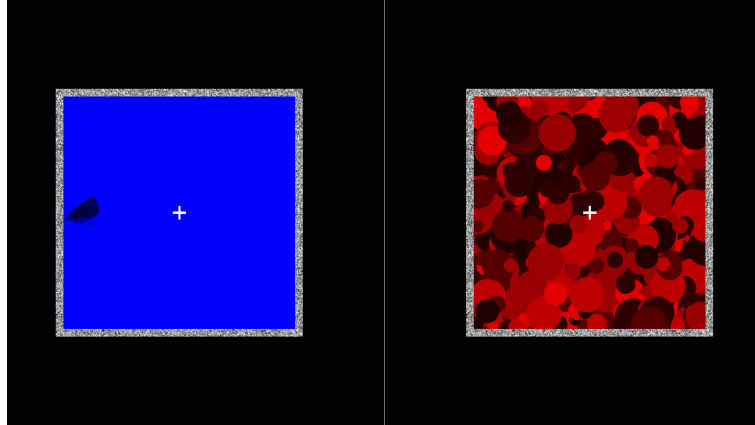


Figure 4.17: Trial of the "bCFS Fovea Periphery" experiment in the Desktop with mirrors platform. The image shows a phase of the trial where the stimulus is at full opacity and the CFS effect is about to reduce. For the lack of a better representation of a VR experiment trial, the desktop with mirrors platform is shown.

4.5 RUNTIME TROUBLESHOOTING FEATURES

Using a configuration file to set values to build an experiment is prone to some errors happening. A format and content validation feature was implemented for the application to help identify these errors and fix them. The purpose of this feature is to provide helpful information about issues that may emerge. Its messages are shown in a window, created for this purpose, and may be of informative, warning, or error type.

Error Messages

The most common errors may happen when setting up the configuration file for the experiment. For example, one of the configuration parameters' values is incorrectly set up, which prompts an error that stops application execution. The window shows the error message (see figure 4.18), so it is possible to identify that error's origin. The error messages are identified by a red cross icon and stop the application's execution.

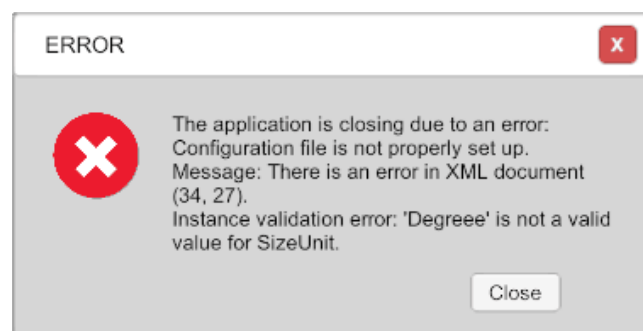


Figure 4.18: Error message window. The message shows an error in a configuration file's parameter value. The value "Degreee" is not valid; it should be "Degree".

Warning Messages

The warning messages happen when a critical operation has been interrupted. An example of this behaviour is when saving the experiment data in the results file. A program, such as Microsoft Excel, can lock the access to the file, which interrupts the saving process. A window displays a message that warns of this occurrence and suggests closing the program locking file access. The saving process can be started again by pressing a "backup save" key, displayed in the window (see figure 4.19 for an illustration of this event). A yellow exclamation mark icon identifies the warning messages.

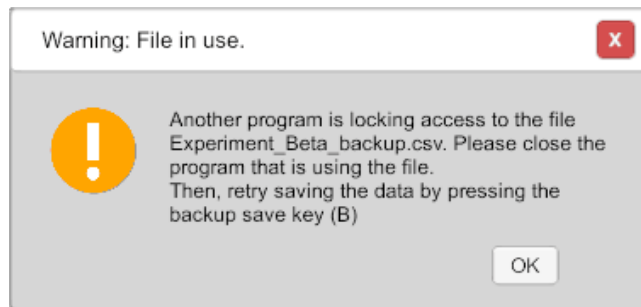


Figure 4.19: Warning message window. The message displayed warns of the existence of another program locking access to the results file. The program should be closed, and the saving process should be retried by pressing the key indicated.

Information Messages

The informative messages are self-explanatory. These messages often appear when exiting the application, prompting confirmation to exit the application. When clicking to exit the application at the end of the experiment, the window will present a message with the results file's save location and the ID attributed to that experiment's participant (see figure 4.20 for an example). A blue circle icon with a lowercase "i" identifies the information messages.

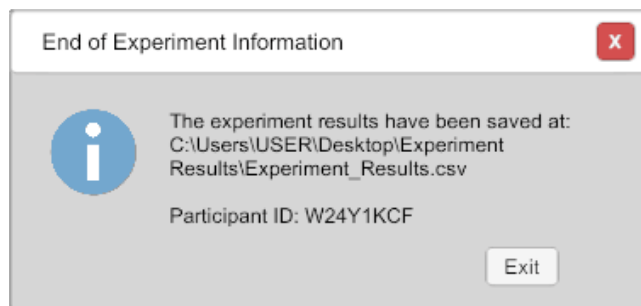


Figure 4.20: Information message window. This message is displayed at the end of the experiment. It shows the save location of the experiment results file and the participant's ID of that session.

4.6 CHAPTER CONCLUSIONS

This chapter described the design and development of an application system to support the experiment protocol requirements. To this end, it was presented the conceptual architecture and its different blocks that compose the application, followed by the application's development and its features that fulfill the requirements proposed

Firstly, it was explained the system's architecture, which is essential to get a better grasp of the application workflow. It was then explained the application's development with Unity and methodology to support the experiments' protocol requirements, which details the processes that occur to set up experiments. The configuration file concept detailed how it drives the experiment set up and its potential to provide different experiment iterations.

Finally, it was explained the application's usage and shown its user interface, displaying some of its features and robustness concepts.

Conclusions & Future Work

5.1 CONCLUSIONS

This dissertation approached several aspects regarding the evolution threat processing research in the Experimental Psychology lab. Threat processing is a Psychology field that studies the emotional stimuli that humans perceive as fear processed under unawareness. As in many works in Experimental Psychology, visual stimuli play a pivotal role in this research field. Recent experimental paradigms have considered increasingly complex methods for stimuli creation and presentation, e.g., bCFS. This complexity, along with a need to move into more ecologically valid experimental settings, prompt the consideration of novel methods and technologies. In this respect, VR has a strong potential, and interest in this technology has been steadily increasing for Experimental Psychology research. This work proposes the creation of a system that allows a study of VR's benefits to provide a richer manipulation of visual stimuli.

This dissertation's general objective was to propose a platform that supports the validation of VR for threat processing research using protocols based on the presentation of visual stimuli such as bCFS. To this effect, and collaborating with the Department of Education and Psychology of the University of Aveiro, an experimental protocol was elaborated to carry out data collection experiments that support the validation of VR based paradigms for threat processing study. The protocol focused on the bCFS experimental paradigm as its complex methodology is well suited for VR. Then, based on the requirements obtained from the previous step, different frameworks were explored. Unity was tested for its support on core aspects of the work to carry out, e.g., create and present bCFS stimuli, gather results, with success, which prompted its selection to develop the platform for experiments.

Subsequently, it was developed the system platform that supports the experimental protocol. The system implements the computational methods for the requirements established, supporting multiple platforms, registering user inputs, supporting display parameter definition in visual angles, storing the experimental data, and so forth, having the versatility to configure different experiment settings through a configuration file. Additionally, it was added support to

calibrate the visual stimuli to align with the participant’s eyes and implemented troubleshooting features that help handling system errors that may occur. Regarding the goal of validating the proposed methods, this was only addressed to a minimal extent. It was demonstrated that the methods support the execution and data collection of experiments in the different setups. However, an extensive validation, supported on the collection of participant data, as was initially intended, was not performed due to constraints evoked by the current pandemic situation. At this time, there is no chance to conduct an experimental data collection.

The application system developed still has some limitations that should be noted for future utilization. Specific experiments that require different inputs or outcomes to what is supported need to be explicitly implemented. The application was developed with a comprehensive design to comprise many possible experiments. However, not every possible experiment variation can be accounted for, and some experiment types may require features that were not developed. The application does not support every VR system. Some VR headsets operate on different frameworks that will not be compatible with the one implemented in this application version (*SteamVR*). VR usage was tested using the HTC Vive, which was the only VR system available.

Despite not having met the goal to evaluate VR through experimental data collections, the application is prepared to configure and run experiments for this purpose when the time comes.

5.2 FUTURE WORK

There are several expectations for future work, whether it is to improve the current version of the application system or take the positive aspects and evaluate the negatives to develop a new, improved version.

Since the VR platform’s assessment as a suitable option was not carried out, an experimental data collection should be the first task to do in the future, when the circumstances allow it. After the assessment is carried out, if proved successful, another study exploring the 3D perception advantages could be explored in this field, creating scenes that render a 3D environment, simulating threatening situations in real-life scenarios.

The application would benefit from implementing various VR frameworks to have better support for other VR headsets. Currently, the VR framework used is *SteamVR*, which is not directly supported by all VR headsets. This implementation would allow better VR displays than the HTC Vive, such as the newly released Oculus Quest 2¹. HTC Vive is a tethered headset that requires a tracking system to be installed. VR systems such as the Oculus Quest 2 streamlines the system setup and portability. It is a wireless VR headset with inside-out tracking, which does not require installing a standalone tracking system. This headset also bears a high-resolution display that improves image quality and reduces the screen door effect significantly, making for a great candidate for the Experimental Psychology lab.

Translating visual angle sizes to the corresponding pixel size displayed is paramount for experiments. The current methodology for this process had the purpose of creating a standard

¹Oculus Quest 2 home page (<https://www.oculus.com/quest-2/>)

process for all platforms. However, this methodology is not straightforward to use in VR displays. VR headsets' display process is complex and draws many variables that seem to influence the sizes displayed. Appendix A.2 contains a detailed discussion on this topic. This methodology would benefit from an eye-to-display distance that better represents the distance to the perceived image in the HMD. A thorough investigation of how the VR display mechanics work would help estimate this distance, replacing the eye-to-display distance used previously. If this distance is estimated, it could then be specified in the configuration file to provide more accuracy in the visual angle translation process.

Although the application is not heavy on performance, it could benefit from performance optimizations if needed in the future. Unity provides a Data-Oriented Technology Stack (DOTS)² containing features that can improve performance, such as dedicated multi-threaded code support. For example, in the current version of the application, these features could be leveraged to reduce the processing time of images when converting them to Sprites.

UXF is an appealing framework to consider for the design of behavioural experiments. While the application's requirements did not justify its integration, a newer version of UXF (2.0)³, containing additional and improved features, has been released during the development of this work. It introduces new and user-friendly features that can enhance the application, such as the new User Interface system and Data Handler methodology. In case the experiment data collection gets more resource-intensive, UXF includes a multi-threaded file Input/Output feature to boost performance that can be implemented if required.

Despite the limitations and methods with room for improvement, the work carried out provides an insight for future endeavours.

Additionally, some noteworthy observations and discussion of possible improvements regarding the experiment object presentation (especially using VR), configuration file usage and compatibility, and application usage are mentioned in appendices A, B, and C.

²Unity DOTS main page (<https://unity.com/dots>)

³UXF 2.0 main page (<http://immersivecognition.com/unity-experiment-framework/>), and documentation (<https://github.com/immersivecognition/unity-experiment-framework/wiki/UXF-2.0>).
[last accessed: 02/03/2021]

Experiment Object Presentation Notes

This appendix serves as a supplementary section focusing on detailing the VR HMD measurements' estimation, namely the width and height for the display screens and the overall distance of the eye to the screen. These are measurements required to calculate the pixel size of stimuli to be displayed in the VR headset. It will also be discussed some of the nuances that the VR display type has. Finally, it explains the process of translating the stimuli size into the pixel representation on display using Unity.

A.1 VIRTUAL REALITY HEAD MOUNTED DISPLAY ESTIMATIONS

The VR HMD selected for experiments, the HTC Vive, lacks official specifications¹ by the manufacturer regarding the screen. There is no explicit mention of each screen's dimension size and about the overall eye-to-screen or even the lens-to-screen distances. This information is essential to calculate the stimuli's pixel size for experiments, so these values had to be estimated.

A.1.1 VR HMD Screen Dimensions

The display's screen dimensions were simple to estimate, as the specifications stated the screen's diagonal size. The width and height could then be calculated using the aspect ratio of the screens. The aspect ratio is given by the resolution of each screen, which is 1080×1200 pixels per eye, giving an aspect ratio of 9:10. The display's dimensions can be obtained using an online display dimension calculator² with these values. The dimensions obtained are approximately 67.9 millimeters (mm) in height and 61.1 mm in width. The calculations can be done without relying on online calculators by merely using the Pythagorean theorem since these measures form a right triangle on the screen (see figure A.1 for an illustration).

¹Vive Developers: Vive Specs & User Guide (<https://developer.vive.com/resources/vive-sense/hardware-guide/vive-specs-user-guide/>)

²Online display dimension calculator (<https://planetcalc.com/1890/>) [last accessed: 02/03/2021].

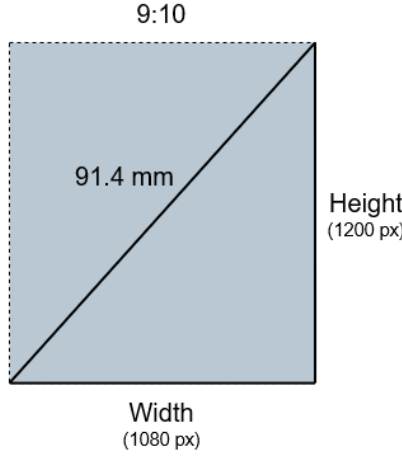


Figure A.1: Screen dimension representation of the VR HMD. The screen has an aspect ratio 9:10 given by the resolution of the screen. Knowing the diagonal size of the screen is 91.4 mm, the width and height dimension can be calculated using the Pythagorean theorem.

The Pythagorean theorem asserts that:

$$91.4^2 = Width^2 + Height^2$$

Knowing the screen's aspect ratio is 9:10, it can be determined that $Width = \frac{9}{10}Height$, and $Height = \frac{10}{9}Width$. The Width and Height can then be calculated by

$$91.4^2 = Width^2 + \left(\frac{10}{9}Width\right)^2$$

and

$$91.4^2 = \left(\frac{9}{10}Height\right)^2 + Height^2$$

After simplifying the results and discarding the negative square rooted value, it is determined that the width and height are, approximately, $Width \approx 61.1$ mm, and $Height \approx 67.9$ mm.

A brief search online has brought attention to a similar display panel³ for VR HMDs. This panel has the same diagonal size and aspect ratio as the HTC Vive's and presents similar display dimensions. Although the panel cannot be confirmed to be the one used in the HTC Vive, its similar dimensions support the notion that the dimensions calculated are plausible.

A.1.2 VR HMD Eye-to-Screen Distance

The eye-to-screen distance is a delicate value to obtain. The method used to obtain this distance is similar to the visual angle equation (4.2) explained in section 4.2.4. In this case, the equation is solved for "Distance" as

$$Distance = \frac{\frac{Size}{2}}{\tan(\frac{\theta}{2})},$$

³Similar VR HMD panel specifications (https://www.panellook.com/AMS361EP01_Samsung_3.6_OLED_overview_33237.html) [last accessed: 02/03/2021].

using the previously estimated display dimensions ("Size") and using the HMD's horizontal FOV as the visual angle (θ).

The FOV value considered was the whole 110° , specified by the manufacturer, for two main reasons. The first reason is that the VR HMD simulates stereoscopic vision, similar to the human eyes (see figure A.2 for an illustrated reference), producing images that simulate each eye's view of the environment that the brain can process into a single picture. In this regard, the FOV considered for calculation should include the whole vision provided by VR HMD. An object's visual angle is calculated in co-ordinance with its distance to both eyes in a natural environment. Similarly, the same concept should apply to a VR headset in a simulated virtual environment. The second reason is that the FOV provided by a single display, the monocular FOV, of the VR headset is an unknown value. The FOV of the HMD cannot be divided in half as the display images overlap, binocular FOV, and this value is influenced by how the manufacturer set up the VR displays. The FOV value is not explicitly stated if it is horizontal, vertical, or diagonal, so it was considered to be horizontal as it is the most plausible value.

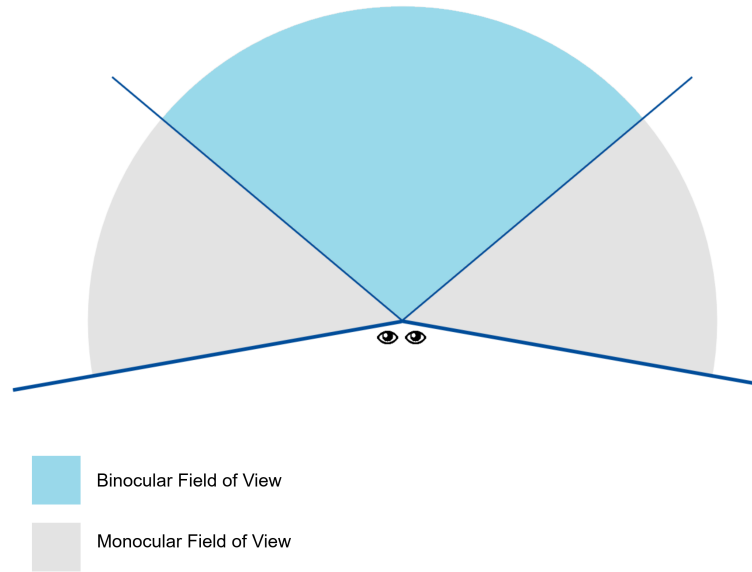


Figure A.2: Human eye FOV representation. The blue zone represents the visual area shared by both eyes, the binocular FOV. The grey areas represent the visual field viewed by each eye, the monocular FOV, the left grey area is view by the left eye and the right area by the right eye.

The eye-to-display distance using the horizontal FOV encompasses both VR screens, giving a total display of dimension $61.1 \times 2 = 122.2$ mm. The distance is calculated by replacing the FOV value (θ) and the total display dimension ("Size") in the previously stated equation:

$$Distance = \frac{\frac{122.2}{2}}{\tan(\frac{110}{2})} \approx 42.8\text{mm}.$$

The estimated 42.8 mm of eye-to-display distance represents a plausible value as it seems a realistic measure from the eyes to the VR HMD's screens. The scene objects scaled using

this value resemble similar perceived sizes as in the desktop monitor. This estimated distance has the assumption that the lens distance of the HTC Vive⁴ is at the closest position to the participant's eyes, which is the position that the VR HMD provides the widest possible FOV, i.e., 110°.

A.2 VR OBJECT PRESENTATION OBSERVATIONS

The nature of the VR device's display has many parameters that need to be taken into account. Using the process explained in section 4.2.4 to scale the scene objects to the correct visual angle is not as straightforward as using the desktop monitor.

The VR headset includes a fisheye lens to each eye, magnifying and distorting the image generated by the display. The display renders images of the virtual environment using inverse lens distortion, such that both distortions cancel out, producing a final image that is perceived naturally. This process enables a wider FOV on the VR HMD, which in the case of the HTC Vive is 110°. Therefore, the visual angles of presented stimuli after lens distortion cannot be directly determined and compared to the desktop monitor, as it influences the scaling process's accuracy.

The way each VR display is set up in the headset may include another factor influencing the scaling process. Each display presents a portion of the image that overlaps with the other display. VR headset manufacturers should test and evaluate this stereoscopic visualization to assert image correctness. The participant's cranium features can also influence the overall distance, as different people would have different distances to the lenses for the same lens distance position in the VR headset.

The scene object scaling process explained in section 4.2.4 intends to standardize a method that works for all display types. Although the VR HMD has many factors that influence the image scaling process, a thorough investigation of how the VR display mechanics work would help estimate the eye distance to the perceived image instead of merely using the eye-to-display distance. Analyzing every aspect that influences the displayed image on the headset: image and lens distortion, lens magnification, and image overlap; would allow an educated estimation of the distance to the VR display's perceived image. This value could then be configured on the configuration file to provide more accuracy in the image scaling process.

A.3 UNITY OBJECT PRESENTATION PROCESS

The experimental task has a 2D virtual environment, which eases the process of translating visual angles of the perceived stimuli into its pixel representation. In Unity, this process was achieved using a Canvas⁵, a GameObject composed, most importantly, by a Canvas component and a Canvas Scaler script. Its general purpose is to draw its containing elements,

⁴HTC Vive manual on adjusting the lens distance on the headset
(https://www.vive.com/us/support/vive/category_howto/adjusting-the-lens-distance.html).

⁵Canvas component Unity documentation
(<https://docs.unity3d.com/2020.1/Documentation/Manual/UICanvas.html>) [last accessed: 02/03/2021].

such as images (experiment frame), into view. Figure A.3 shows the Unity inspector window of the Canvas GameObject used to contain the desktop platform's experiment frame. The most significant fields to pay attention to are the "Render Mode" of the Canvas component, "UI Scale Mode" of the Canvas Scaler script, and the "Width" and "Height" of the Rect Transform component.

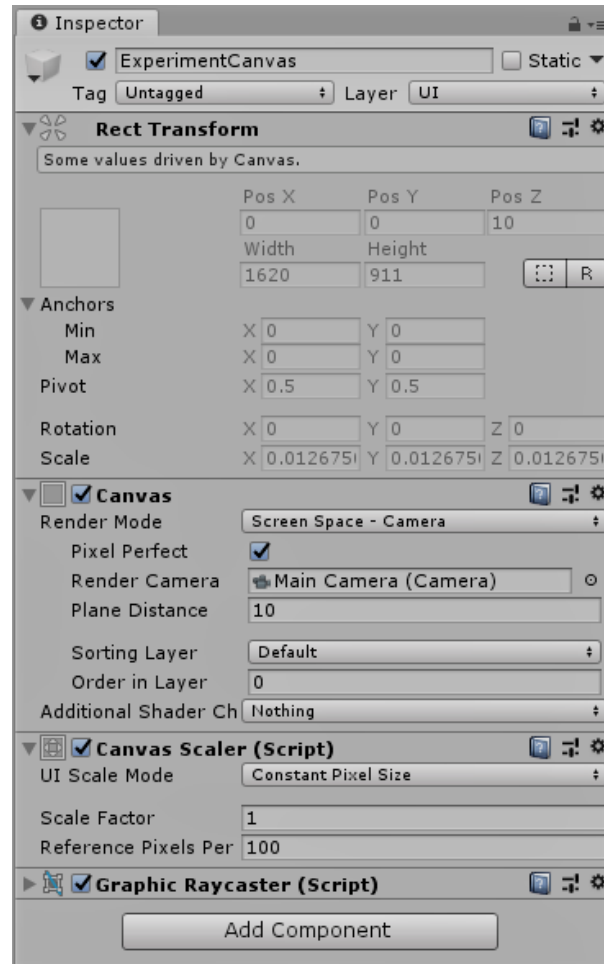


Figure A.3: Unity Canvas GameObject inspector window with the settings used to contain the experiment object images. The "Width" and "Height" field adjust according to the application window resolution. In full-screen the "Width" and "Height" would show the monitor's resolution's 1920 and 1080 respectively, Unity editor does not allow a complete full-screen.

The "Render Mode" field of the Canvas is set to *Screen Space - Camera*. This setting essentially makes the Canvas and its containing GameObjects to be rendered by a Camera GameObject, the "Main Camera", driving the Canvas to fill the screen space and accommodate the Camera's settings (see figure A.4 for an illustration). If the screen is resized or the resolution changes, the Canvas will automatically match those changes. In this case, the "Width" and "Height" of the Rect Transform component of the Canvas matches the same resolution of the application window.

The "UI Scale Mode" parameter of the Canvas Scaler script is set to *Constant Pixel Size*.

This setting makes the items displayed on the Canvas comply with the display's resolution, allowing to scale of the frame's objects in pixel size. This setting also allows the containing objects to maintain the same amount of pixels regardless of the application window resolution, which assures more consistency in the presentation of stimuli. If the application window is resized, the objects will not change to match the same resolution, keeping the same size. The "Pixel Perfect" field is enabled as, according to the documentation⁶, it forces the elements in the canvas to be aligned with pixels, making them appear sharper and prevent blurriness.

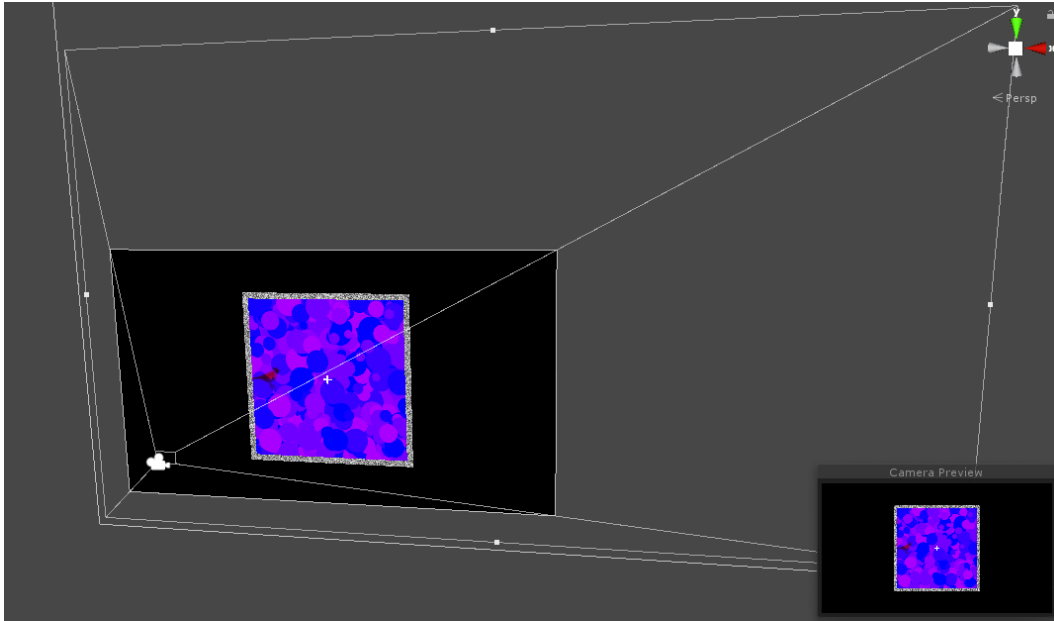


Figure A.4: Unity Canvas and Camera of the desktop platform in 3D perspective view. The white lines represent the camera perspective view. The Canvas matches the Camera's view, which encompasses the screen resolution. At the bottom right corner, a preview of the camera view as seen from the monitor.

This concept is also applied to the VR platform, only in this case, there is a Canvas rendered for each display that presents the corresponding experiment stimuli to each eye. The VR HMD's perceived FOV is not affected by this methodology, as both display's rendered Canvas have the same settings and background color, which makes the perceived image appear as if it was a single Canvas across both screens. Each frame is then presented on the corresponding display, overlapping them to perform the CFS effect used for the experiments.

⁶Canvas Pixel Perfect Unity documentation
(<https://docs.unity3d.com/ScriptReference/Canvas-pixelPerfect.html>) [last accessed: 02/03/2021].

Configuration File Usage Notes

This appendix thoroughly details information regarding the parameters' value types and their role in the experiment set up. It represents a manual explaining how to use the configuration file to set up experiments. It also discusses its compatibility to set up other types of experiments and possible improvements.

B.1 CONFIGURATION FILE MANUAL

Section 4.3.2 was given a brief explanation of each configuration parameter's different values and their primary function. The explanation was shortened to illustrate the configuration file's fundamental idea concisely. This section will keep a similar structure to section 4.3.2, carefully reviewing each value type, possible values, and default values of the parameters to fully explain how to use the configuration file to set up experiments.

The "configurations" element nests all parameters that compose the configuration file. After deserialization, this element is converted to an object that holds all the configuration data. The configuration file can be found in the application data directory under the *StreamingAssets/ExperimentData/Config* folder with the filename "config.xml".

```

<configurations>
  <experimental_paradigm/>      <!-- Experimental Paradigm Parameter -->
  <display_system/>             <!-- Display System Parameter -->
  <data/>                       <!-- Data Parameter -->
  <experiment_settings/>        <!-- Experiment Settings Parameter -->
  <pre_task_pages/>             <!-- Pre-Task Pages Parameter -->
</configurations>

```

Listing B.1: A demonstrative example of the XML configuration file parameter structure. Each element represents a configurable parameter (labeled) that can be manipulated to set different experiment configurations. The content of the elements has been omitted for demonstration purposes.

Experimental Paradigm

The experimental paradigm parameter consists solely of the "experimental_paradigm" element holding the value that stipulates the experimental paradigm to be performed. This value is directly linked with the type of *Experiment Manager* used by the system (see the "Experiment Manager" subsection in section 4.1.3 for a review). The software system supports three different "experimental_paradigm" values, shown in table B.1.

Experimental Paradigm Value	Description
bCFS_F_P	Specifies the fovea and periphery bCFS experimental paradigm proposed for this work's experiments.
bCFS_Simple	Specifies a simplified bCFS experiment only registering the participant's reaction time.
bCFS_Sides	Specifies a simplified bCFS experiment registering both the participant's reaction time and stimulus frame side guessed.

Table B.1: Experimental paradigm parameter values implemented in the software platform. Each value instructs the application to set up and perform experiments specifically.

This work's experimental paradigm value is *bCFS_F_P* (see listing B.2 for reference). The application system identifies this value and prepares the experiments for the bCFS paradigm presenting the stimuli to the fovea and periphery positions, as accorded in the protocol designed for the experiments (see section 3.2 for reference). The other values were implemented so the application could set up other experiments based on visual stimuli presentation that use similar methodologies to obtain results, unconstrained from the fovea and periphery experiments' specific operations.

The *bCFS_Simple* value tells the software system to set up a simplified version of a bCFS experiment, only evaluating the participant's reaction time, disregarding the position where the stimulus was detected. The *bCFS_Sides* value provides the same simplified version for bCFS experiments as the *bCFS_Simple* value; however, it registers both the participant's reaction time and stimulus frame side guessed. Adding other experimental paradigm values that instruct the software platform to perform different actions requires a specific Experiment Manager version to be implemented, which is the system component that manages those operations.

```
<experimental_paradigm>bCFS_F_P</experimental_paradigm>
```

Listing B.2: Example of an experimental paradigm parameter with the value for this work's experimental paradigm.

This parameter must contain a value as it is required to set up the experiments. In the case of value omission, an error is shown, stopping application execution.

Display System

The display system parameter includes all the settings about the display platform used for the experiments (see listing B.3 for a reference). The platform is specified in the "platform" attribute, which holds a value to instruct the software application on which platform to set up the experiments. Table B.2 presents the possible values to specify the experiment's platform. The application defaults to the *Desktop* platform in case the platform attribute is not included.

Platform Value	Description
Desktop (DV)	Instructs the system to set up the experiment for the Desktop display platform, using the Red-Blue anaglyph 3D glasses presentation method.
DesktopMirror	Instructs the system to set up the experiment for the Desktop display platform, using the mirror apparatus presentation method.
Vr	Instructs the system to set up the experiment for the Virtual Reality display platform.

Table B.2: Display platform values implemented in the software system. Each value instructs the application to set up experiments to the specified platform value. The default platform value (DV) is "Desktop" if it is not set.

The "display" element comprises the technical information about the display used for the experiment, containing attributes that hold the values for the display's width and height resolution and dimensions. The "distance_to_screen" element has the value for the participant's eyes' distance to the display screen. These values are required so the application system can translate the specified stimuli visual angle sizes into its pixel representation, as explained in section 4.2.4. The measurement unit used to define these values should be the same for the size translation to work correctly, i.e., if the "distance_to_screen" value was set in millimetres, the display's size dimension should be specified in millimetres as well.

```
<display_system platform="Desktop">
  <display pixel_width="1920" pixel_height="1080" width="476" height="267"/>
  <distance_to_screen>500</distance_to_screen>
</display_system>
```

Listing B.3: Example of a system parameter configuration for the Desktop platform experiment. The metric measurements ("width", "height", and "distance_to_screen") are in millimeters.

Data

The data parameter contains all the settings required to store and organize the experiment results file (see listing B.4 for a reference). Section 4.2.5 discussed that the experiment results are stored in the CSV format, saving the results in one CSV line to hold all experiment results in a single file. The "data" element contains a "participant_id_length" attribute that receives an integer value to identify the results with a randomized alphanumeric identification with the specified length. The length of this alphanumeric identification defaults to 8 in case of its omission. Since the results are stored in a CSV file, it is possible to configure the

"csv_delimiter" to better customize the data organization, with a default value ";" in case of exclusion of this attribute.

```
<data participant_id_length="8">
  <csv csv_delimiter=";">
    <save_path base_folder="Desktop">Experiment Results</save_path>
    <filename>Experiment_Results.csv</filename>
  </csv>
</data>
```

Listing B.4: Example of data parameter settings, using the relative path with the base folder method to generate the save path.

The filename and save path for the experiment results can be specified using the "filename" and "save_path" elements, respectively. The "filename" holds a representative name given to the results file. The save path for the experiment results can be specified in two distinct methods (see listing B.5 for an example of both methods). It can be specified as an absolute path or a relative path and a "base_folder" attribute. The latter method relies on a set of special folders¹ recognized by the operating system. The "base folder" should contain a value that represents the intended base folder, e.g., *Desktop*, *MyDocuments*. This attribute will instruct the application to get the absolute path to the specified special folder and join it with the given relative path. Using the "base folder" method circumvents the necessity to replace the absolute path if using a different computer or folder, since the software system gets the path to the Desktop folder, for example, by itself.

```
<!-- Absolute path method -->
<save_path>C:\Users\[USERNAME]\Desktop\Experiment Results</save_path>

<!-- Relative path with base folder method -->
<save_path base_folder="Desktop">Experiment Results</save_path>

<!-- Both method examples generate the same path:
      C:\Users\[USERNAME]\Desktop\Experiment Results -->
```

Listing B.5: Example of both save path methods. The first example shows the absolute path method, while the second shows the relative path with the base folder method. Both demonstrated methods produce the same save path, as shown in the last commented lines.

Experiment Settings

The experiment settings parameter contains the values responsible for the experiment scenery set up and other procedure and execution settings. For a reference, see listing B.6, which shows an example of the settings used to set up an experiment for the designed protocol, discussed in section 3.2.

¹The special folders are obtained using a specific enumerator as described here (<https://docs.microsoft.com/en-us/dotnet/api/system.environment.specialfolder?view=netcore-3.1>) [last accessed: 02/03/2021]

```

<experiment_settings color="Red_Blue" training_set_size="20" trial_repetitions="3">
  <frame size="16" size_unit="Degree" reference_dimension="Height">
    <stimuli size="2" size_unit="Degree" reference_dimension="Height" sort_type="Random">
      <positions distance_unit="Degree">
        <position horizontal_distance="1" vertical_distance="0" label="fovea"/>
        <position horizontal_distance="6" vertical_distance="0" label="periphery"/>
      </positions>
    </stimuli>
    <masks cfs_step="100" size="15" size_unit="Degree" reference_dimension="Height"/>
  </frame>
  <duration raise_stimulus="1100" lower_mask="4000" stimulus_exposed="3000"/>
  <experiment_data_folders>
    <experiment_folder>Experiment Stimuli</experiment_folder>
    <training_folder>Training Stimuli</training_folder>
    <mask_folder>Masks</mask_folder>
  </experiment_data_folders>
</experiment_settings>

```

Listing B.6: Example of the experiment settings parameter for this work’s experiment.

The "experiment_settings" element nests all the other elements that compose this parameter. This element contains attributes that define some of the experiment procedure, such as the color setting to display the stimuli, training set size and the number of trial repetitions for the experiment. The "color" attribute contains a value that dictates the color filtering used in the experiments. The values implemented can be seen in table B.3. The *Natural* color setting is the default value when the "color" attribute is not specified. This value instructs the application to present the frame stimuli’s true colors (stimuli and mask) to the participant without applying color filtering. When the *Red_Blue* value is specified, the application filters the stimulus and mask colors to the Blue and Red RGB colors, respectively. This setting is required to separate the images to each eye when using the Red-Blue anaglyph glasses for the desktop experiment.

Color Value	Description
Natural (DV)	Presents the experiment’s frame stimuli with the image’s natural color.
Red_Blue	Presents the stimuli and masks filtered to the Red and Blue RGB channels, respectively.

Table B.3: Experiment’s color filtering values implemented in the software system. Each value instructs the application to present the experiment’s frame stimuli with distinct color filtering. The default color value (DV) is "Natural" if it is not defined.

The number of trials for the training phase is specified in the "training_set_size" attribute. This value is required to be defined to perform the training phase with any trial. If, by mistake, the training set size is defined to be larger than the possible amount of trials (not counting trial repetitions), the value defaults to that amount. The number of trial repetitions is defined in the "trial_repetitions" attribute, which defaults to 1 in case of no definition.

The "frame" element nests the elements that form the objects of the experiment frame. Each frame object can be resized through the "size" attribute, i.e., the noise border, stimuli,

and masks. This attribute should be followed by the "size_unit" and "reference_dimension" attributes, which will instruct how to interpret the "size" value to translate to its display pixel representation. Table B.4 shows the possible "size_unit" values the application system recognizes to interpret "size". The default value for the size unit is *Degree*, as it is the most common size measurement used for psychological experiments.

Size/Distance Unit Value	Description
Degree (DV)	The system interprets the "size"/"distance" value as degrees of visual angle and translates it to its display pixel representation.
Radian	The system interprets the "size"/"distance" value as radians of visual angle and translates it to its display pixel representation.
Pixel	The system directly uses the "size"/"distance" value as the amount for its display pixel representation.

Table B.4: Size/distance unit values implemented in the software system. Each value instructs the system to interpret the "size"/"distance" attribute value accordingly. The default size/distance unit value (DV) is *Degree* if it is not specified.

The "reference_dimension" instructs the system to resize an image on the indicated dimension. As stated in section 4.2.4, the images can become deformed if resized on both dimensions due to not equaling the image's aspect ratio. The implemented values can be seen in table B.5. This value defaults to *Height* if it is not specified. As seen in listing B.6, stimuli are resized to have the display pixel representation of 2° of visual field height.

Reference Dimension Value	Description
Height (DV)	The system scales the object according to its height.
Width	The system scales the object according to its width.

Table B.5: Reference dimension values implemented in the software system. Each value instructs the software system to scale the object according to its height or width to avoid deformation. The default reference dimension value (DV) is *Height* if it is not defined.

The "stimuli" element contains a "sort_type" attribute that instructs the software system to organize the stimuli order accordingly for trial presentation. The sort type values implemented can be seen in table B.6. The sort type value defaults to *Random* if it is not set. The "stimuli" element contains a "positions" element that nests multiple "position" elements. These elements contain the information of each position the stimuli should be presented during the experiment. The "positions" element includes a "distance_unit" attribute, similar to "size_unit", which holds the unit type (see table B.4) to interpret the included positions. Each "position" has "horizontal_distance" and "vertical_distance" attributes to define its distance to the center of the frame. Additionally, these elements contain a "label" attribute to identify each position (these labels are not required in the *bCFS_F_P* experimental paradigm, as the system can identify the positions).

The "masks" element contains an additional "cfs_step" element to set the time between interchanging the Mondrian mask image to produce the CFS effect. This value is interpreted

as milliseconds by the system. As shown in the listing B.6 example, it is set to 100 milliseconds, which is the time stipulated to create the CFS effect, discussed in section 3.2.

Sort Type Value	Description
Random (DV)	Produces a random order for stimuli trial presentation.
Ascending	Sorts stimuli alphanumerically ascending using their identification name.
Descending	Sorts stimuli alphanumerically descending using their identification name.
None	Stimuli are presented in the order they are loaded into the system.

Table B.6: Sort type values implemented in the software system. Each value instructs the system to organize the stimuli accordingly for the experiment trials. The default sort type value (DV) is *Random* if it is not set.

The "duration" element comprises the time duration for each phase of the trial. It contains attributes representing each trial phase, performing a specific action with that duration in coordination with the experimental paradigm. Each attribute value is interpreted in milliseconds by the system and is specific to the bCFS paradigm. The "raise_stimulus" attribute represents the duration to perform the stimulus's fading to make it visible. The "lower_mask" attribute indicates the duration to lower the intensity of CFS, while the "stimulus_exposed" attribute retains the amount of time until the trial ends, where the stimulus is exposed.

The objects presented during the experiment, i.e., stimuli and Mondrian mask images, come from external locations (folders) of the computer, where the system will load every image comprised in them. Using this approach allows full control to the examiner to add or change stimuli without needing further assistance. These locations are specified in a set of elements nested inside "experiment_data_folders". The "experiment_folder" sets the path where the stimuli images intended to be used for the experiment are located. Similarly, the "training_folder" sets the folder path to the stimuli images to be used for the training phase. The "mask_folder" contains the folder path for the pre-rendered Mondrian pattern images to be used to create the CFS effect. The folder path can be defined as an absolute path to the location or relative path. When using a relative path, the software system will search for it under *StreamingAssets/ExperimentData*, a generated folder under the application data directory (see listing B.7 for an example).

```

<!-- Absolute path method -->
<experiment_folder>C:\Users\[USERNAME]\Pictures\Experiment Stimuli</experiment_folder>
<training_folder>C:\Users\[USERNAME]\Pictures\Training Stimuli</training_folder>

<!-- Relative path method -->
<mask_folder>Masks</mask_folder>
<!-- Generated Path:
[APPLICATION_PATH]\[APP_NAME]_Data\StreamingAssets\ExperimentData\Masks -->

```

Listing B.7: Example of the absolute path and relative path usage to indicate the experiment data folder paths. The system searches for the indicated relative path under *StreamingAssets/ExperimentData* in the application data directory.

Pre-Task Pages

The pre-task pages parameter allows the system to set a customized text for the descriptive pages that appear before the experiment. The "pre_task_pages" element nests a set of "page" elements, each containing "name" attribute and nesting a "title" and "description" element. The "name" attribute instructs the system on which specific page the "title" and "description" texts should appear. Table B.7 shows a description of all the values that identify the pages. Listing B.8 shows the example used for section 4.4 to display the descriptive pages.

Page Name Value	Description
Introduction	Text is displayed in the first page that introduces to the task.
Participant_Information	Text is displayed in the participant information area.
Pre_Calibration	Text is displayed in the page before the calibration phase.
Pre_Training	Text is displayed in the page before the training phase.
Pre_Experiment	Text is displayed in the page before the experiment.
Experiment_End	Text is displayed in the page at the end of the experiment.

Table B.7: Page name values implemented in the software system to instruct in which page to insert the customized title and description text.

```
<pre_task_pages>
  <page name="Introduction">[...]</page>
  <page name="Participant_Information">
    <title>Information data area</title>
    <description>
Please enter the information requested.
This information is relevant for the experiment and analysis.
    </description>
  </page>
  <page name="Pre_Calibration">
    <title>Image Calibration</title>
    <description>
The next task intends to calibrate the position of the scene images.
The system will present an image to each eye consisting of a fixation cross and a noise border.
Each image should be aligned with each eye.

Use the arrow keys or the controller's trackpad to move the images.
To move a single image, press the left/right control (CTRL) key, or the left/right trigger buttons,
accordingly, while moving the images.
    </description>
  </page>
  <page name="Pre_Training">[...]</page>
  <page name="Pre_Experiment">[...]</page>
  <page name="Experiment_End">[...]</page>
</pre_task_pages>
```

Listing B.8: Example of the pre-task pages parameter used for section 4.4 to display the descriptive pages. Some "page" elements are shown to have their content omitted for the sake of brevity, as it was established a pattern of their usage.

B.2 CONFIGURATION FILE COMPATIBILITY DISCUSSION

The configuration parameter structure was designed to be compatible with similar nature experiments, using images as stimuli and presenting them in a frame that performs CFS. However, the performance and result of experiments are dependent on the implementations of *Experiment Managers* (see section 4.1.3 for a reference), which are directly tied to the defined "experimental_paradigm" value.

For example, for this work's experiment, a specific "Fovea Periphery" Experiment Manager was created to control the experiment according to the established experimental paradigm. This Experiment Manager will be used when specifying the *bCFS_F_P* experimental paradigm value. There are two generic Experiment Managers variants to add support for other paradigms: a "Simple" (*bCFS_Simple*) and a "Sides" (*bCFS_Sides*) Experiment Manager. The "Simple" variant controls the experiment, similar to the "Fovea Periphery" implementation. However, it does not have any specificity, registering only reaction times for the results. The "Sides" variant has the same behaviour; however, it registers the reaction times and the side the user guessed the stimulus to be, similar to the "Fovea Periphery" experiments.

The "Fovea Periphery" Experiment Manager includes specific processes to generate the experiment set. It only requires two positions, the fovea and periphery, generating the left and right positions for each. It also contains other processes for the evaluation and registration of results. Conversely, the generic variants process the positions that are configured without any specific processing. To this end, if the experiment requires handling special operations that the generic Experiment Managers do not cover, it would be necessary to implement a specific variant of it.

In case an experiment requires a different structure of the configuration file, that support can be achieved by implementing the configuration data structure that allows the configuration file's deserialization to an object for the system to access. The configuration data object can be instantiated by implementing a method that identifies the "experimental_paradigm" value and returns the correct configuration data object. However, in the current software application version, that support is not implemented, as it was not required.

B.3 FUTURE IMPROVEMENTS

The configuration file could have some improvements made. The "duration" element could have a similar structure to the "positions" element, where "duration" elements would be nested to represent actions and their period. Implementing this methodology would add more complexity, as it requires linking each "duration" to a specific action. Limiting the number of durations each experimental paradigm supports and organizing them with an "order" attribute could prove to be a solution for this issue. For instance, in this work's experimental paradigm, the trials have three phases, each with a given duration amount. If more or less than three durations were instantiated in the configuration, the system would prompt an error, referring to this issue. Linking a duration to each trial phase's action could be accomplished by the

organized "duration" set, as the first "duration" of the set would be linked to the first action, and so on.

Other "quality of life" improvements could be provided. The "base folder" method could be implemented for the paths nested inside "experiment_data_folders". This approach would bring the same advantages as the "base folder" method, as stated previously. The data parameter could include an option that instructs the system to automatically join a subfolder to the results save path corresponding to the experimental paradigm. For instance, for this work's "bCFS Fovea Periphery" experimental paradigm, a subfolder named "BreakingCfsFoveaPeriphery" would be joined to the save path. This option would automatize the process of explicitly instantiating the save path for each experimental paradigm if that folder organization was required. Another improvement could come from creating an in-app editor to manage the configuration file. This way, the examiner could edit the file without searching for it in the application data directory and facilitating its manipulation through an intuitive interface design.

Application Usage Notes

This appendix presents the defined keyboard keys and HTC Vive controller buttons to interact with the software platform. Its purpose is to complement section 4.4, where it was discussed the application usage, with the action that each input performs to control the experiment or for the participant inputs. It will also mention the system requirements to use the application and discuss some improvements that can be made.

C.1 APPLICATION INPUT MAPPING

The software platform supports inputs from both the keyboard and the Vive controllers to perform specific actions according to the application phase. The inputs will be necessary for the calibration, training, and experiment phase. A Vive controller button diagram is shown in figure C.1. This section will indicate each input supported and explain the action it performs.

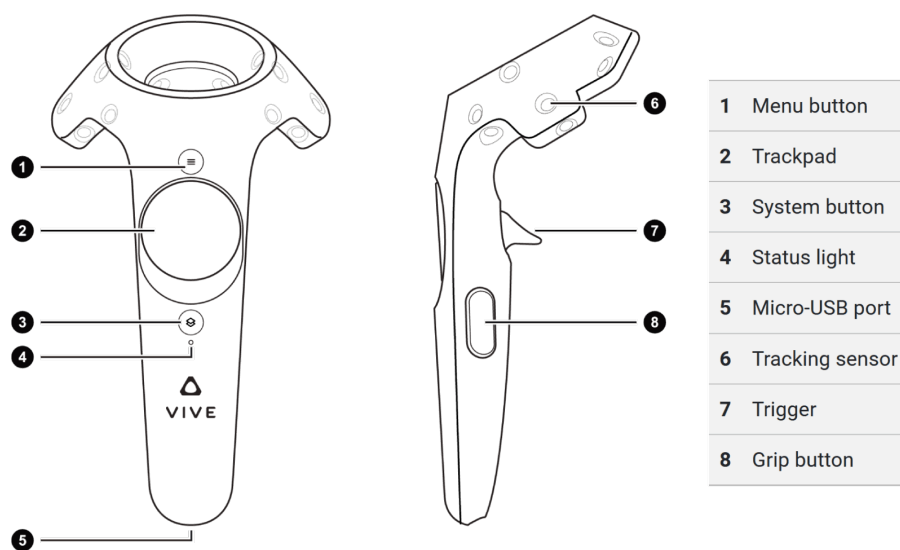


Figure C.1: Vive controller input diagram. Image retrieved from [here](#)¹.

Calibration Inputs

The calibration phase aims to adjust the experiment frame’s position to align with the participant’s eyes. For the binocular experiments, i.e., desktop with mirrors and VR, there are two frames, one for each eye, that need to be controlled. The desktop experiment using anaglyph glasses contains only a single frame.

The single experiment frame’s calibration phase supports four actions: frame repositioning, frame repositioning speed control, reset position, and confirm positioning. Repositioning the frame is done by pressing the left/right/up/down inputs to move the frame accordingly. The keys supported to reposition the frame are the *WASD* or *Arrow* keys. The Vive controller buttons for this action are the *Up*, *Down*, *Left*, and *Right* buttons of the *Trackpad*. The reposition control input can be held to restricts the movement speed, providing more precision, achieved by holding the *Left/Right CTRL* keys or the grip button using the Vive controllers. Pressing the *R* key or the Menu button resets the frame’s position to the initial one, while pressing the *Enter* key confirms the frame’s calibrated position, finishing the calibration phase. Table C.1 shows the list of inputs and their action for the single frame calibration phase.

Input		Action Description
Keyboard	Vive controller	
W or Up Arrow	Trackpad Up	Hold to move the frame towards the top.
A or Left Arrow	Trackpad Left	Hold to move the frame to the left.
S or Down Arrow	Trackpad Down	Hold to move the frame towards the bottom.
D or Right Arrow	Trackpad Right	Hold to move the frame to the right.
Left or Right CTRL	Grip button	Hold to restrict the frame’s movement speed when moving.
R	Menu button	Click to reset the frame to the initial position.
Enter	None	Click to confirm the frame’s calibrated position and finalize the calibration phase.

Table C.1: Input action mapping for the single frame calibration phase (desktop platform using anaglyph glasses).

The calibration phase for the binocular (dual) experiment frames, i.e., VR and desktop with mirror platforms, has a more advanced input mapping to accommodate a better calibration control to both experiment frames. This input mapping shares the same reset, confirm, and frame repositioning speed control actions previously discussed for the single frame calibration. However, the repositioning input controls follow a different logic. Since there are two experiment frames to be repositioned, the frame movement inputs move the frames symmetrically. If a frame moves up or down, the other follows. If moving left or right, the

¹https://www.vive.com/us/support/vive/category_howto/about-the-controllers.html
[last accessed: 02/03/2021]

other frame moves the opposite way. The *WASD* keys, or the left Vive controller’s *Trackpad* buttons (*Up*, *Down*, *Left*, and *Right*), move both frames using the left one as the reference to manage which frame commands the movement. Similarly, the *Arrow* keys, or the right controller’s *Trackpad* buttons, move both frames using the right one as the reference.

Additionally, it was added an input that selects a single frame to move solely. Holding the *Left Shift* key or the left controller’s *Trigger* will select the left frame to move exclusively. Similarly, holding the *Right Shift* key or the right controller’s *Trigger* will select the right frame to move. While selecting a single frame to move, pressing any of the movement inputs will move that frame. Table C.2 shows the list of inputs and their action for the dual-frame calibration phase.

Input		Action Description
Keyboard	Vive controller	
W or Up Arrow	(Any) Trackpad Up	Hold to move both frames, or the selected frame, towards the top.
S or Down Arrow	(Any) Trackpad Down	Hold to move both frames, or the selected frame, towards the bottom.
A	(Left) Trackpad Left	Hold to move both frames symmetrically to the left frame’s left or the selected frame to the left.
D	(Left) Trackpad Right	Hold to move both frames symmetrically to the left frame’s right or the selected frame to the right.
Left Arrow	(Right) Trackpad Left	Hold to move both frames symmetrically to the right frame’s left or the selected frame to the left.
Right Arrow	(Right) Trackpad Right	Hold to move both frames symmetrically to the right frame’s right or the selected frame to the right.
Left or Right CTRL	(Any) Grip button	Hold to restrict the frames’ movement speed when moving.
Left Shift	(Left) Trigger	Hold to select the left frame to move exclusively.
Right Shift	(Right) Trigger	Hold to select the right frame to move exclusively.
R	(Any) Menu button	Click to reset the frames to the initial position.
Enter	None	Click to confirm the frame’s calibrated position and finalize the calibration phase.

Table C.2: Input action mapping for the dual-frame calibration phase (VR and desktop using mirror apparatus platforms). The Vive controller that performs a specific action is specified in parenthesis.

Experiment Inputs

The Experiment (and training) phase performs the experiment trials, where the participant will react to stimuli according to the experimental paradigm. There are two types of input action sets defined for this phase: inputs to control the experiment and inputs to react to the trials. All of the display platforms share the same inputs. However, the reaction inputs

vary according to the experimental paradigm. The experiment control input actions are not influenced by this factor and remain the same.

The defined reaction inputs support the experiment trials for the implemented experimental paradigms, i.e., "Fovea Periphery", "Simple", and "Sides" (see "Experimental Paradigm" in section B.1 for a reference). The "Fovea Periphery" and "Sides" experimental paradigms register which side of the frame the participant detected the stimulus. The detection of the stimulus to the left side of the frame is registered by pressing the *A* key or the left Vive controller *Trackpad* buttons or *Trigger*. Pressing the *L* key or the right controller *Trackpad* buttons or *Trigger* registers a participant detection of the stimulus to the frame's right side. The "Simple" experimental paradigm only registers a reaction to the stimulus, regardless of its position. Pressing the *Spacebar* key or the *Trackpad* buttons or *Trigger* of any Vive controller registers the participant's detection of the stimulus.

The experiment control inputs are organized into three distinct actions: start/pause the experiment trial execution, reset experiment trial execution, and retry the save process. Pressing the *Enter* key or the *Menu* button of any Vive controller starts or pauses the experiment's execution. At the beginning of the experiment trials, the participant presses this button to start the experiment at will. If needed, pressing the same input during the trials halts the experiment execution, while pressing it again resumes the experiment. If some event disturbs the experiment execution, a reset to the beginning of the experiment execution can be performed by pressing the *R* key. At the end of the experiment, if the experiment results' save process was disrupted because, for instance, the save file's access is locked by another program, the save process can be retried by pressing the *B* key. Table C.3 shows the list of inputs and their action for the experiment (and training) phase.

Input		Action Description
Keyboard	Vive controller	
A	(Left) Trackpad or Trigger	Click to register the detection of the stimulus on the frame's left side.
L	(Right) Trackpad or Trigger	Click to register the detection of the stimulus on the frame's right side.
Spacebar	(Any) Trackpad or Trigger	Click to register the detection of the stimulus ("Simple" experimental paradigm only).
Enter	(Any) Menu button	Click to start/pause the experiment execution.
R	None	Click to reset the experiment execution to the beginning.
B	None	Click to retry the save process of the experiment results.

Table C.3: Input action mapping for the Experiment (and training) phase. The Vive controller that performs a specific action is specified in parenthesis.

As a side note, it is possible to exit the application at any phase. Pressing the *Escape* key will display a window that prompts a confirmation to exit the application. This option was implemented if the application needed to be restarted for any reason, e.g., the configuration

file specified the wrong platform for that experiment session.

C.2 SYSTEM REQUIREMENTS

The software application is not very resource-intensive, and any modern system should be able to run it. However, to use the HTC Vive VR system, the computer system should equip the recommended specs² for better compatibility. Additionally, the software platform requires *SteamVR* to be installed in the system.

The computer should include a *QWERTY* keyboard that includes the *Arrow* keys to use the implemented calibration phase keyboard inputs to the full extent.

C.3 FUTURE IMPROVEMENTS

The software platform could have some improvements made, which were not required for this version, to improve its usage quality. The application would benefit from an input mapping system that allows changing keys or buttons instead of using a fixed input mapping. This system could be implemented through an in-app menu or via a button mapping configuration file. This solution could prove useful for different scenarios, such as if a different button mapping was required for a particular experiment or a physically impaired participant.

The Vive controllers lack buttons to implement all input actions. While some were purposefully implemented only for the keyboard (reset experiment execution and retry the save process), the controllers lack an input method to confirm the frame position in the calibration phase. This input could be implemented, for example, by creating a method that detects the same inputs, e.g., *Grip* buttons, in both controllers being held at the same time for a few seconds. This way, the whole calibration process could be controlled using the Vive controllers.

Other "quality of life" usage improvements could be implemented, such as an input method from the keyboard or Vive controllers to advance from the descriptive pages to the next, essentially "clicking" the displayed "Next" button without the computer mouse. Another input method could be added, which displays a "Help" page visually explaining all the inputs available for that phase of the application.

²HTC Vive's recommended computer specifications (<https://www.vive.com/eu/ready/>) [last accessed: 02/03/2021].

References

- [1] L. A. Isbell, “Snakes as agents of evolutionary change in primate brains”, *Journal of Human Evolution*, vol. 51, no. 1, pp. 1–35, 2006. DOI: 10.1016/j.jhevol.2005.12.012.
- [2] C. Darwin, *The expression of the emotions in man and animals*. John Murray, 1872. DOI: 10.1037/10001-000.
- [3] I. Almeida, S. C. Soares, and M. Castelo-Branco, “The distinct role of the amygdala, superior colliculus and pulvinar in processing of central and peripheral snakes”, *PLoS ONE*, vol. 10, no. 6, e0129949, 2015. DOI: 10.1371/journal.pone.0129949.
- [4] Q. Van Le, L. A. Isbell, J. Matsumoto, V. Q. Le, E. Hori, A. H. Tran, R. S. Maior, C. Tomaz, T. Ono, and H. Nishijo, “Monkey pulvinar neurons fire differentially to snake postures”, *PLoS ONE*, vol. 9, no. 12, N. P. Holmes, Ed., e114258, 2014. DOI: 10.1371/journal.pone.0114258.
- [5] R. S. Maior, E. Hori, M. Barros, D. S. Teixeira, M. C. H. Tavares, T. Ono, H. Nishijo, and C. Tomaz, “Superior colliculus lesions impair threat responsiveness in infant capuchin monkeys”, *Neuroscience Letters*, vol. 504, no. 3, pp. 257–260, 2011. DOI: 10.1016/j.neulet.2011.09.042.
- [6] N. Tsuchiya and C. Koch, “Continuous flash suppression reduces negative afterimages”, *Nature Neuroscience*, vol. 8, no. 8, pp. 1096–1101, 2005. DOI: 10.1038/nn1500.
- [7] Y. Jiang, P. Costello, and S. He, “Processing of invisible stimuli: Advantage of upright faces and recognizable words in overcoming interocular suppression”, *Psychological Science*, vol. 18, no. 4, pp. 349–355, 2007. DOI: 10.1111/j.1467-9280.2007.01902.x.
- [8] N. Gomes, S. Silva, C. F. Silva, and S. C. Soares, “Beware the serpent: the advantage of ecologically-relevant stimuli in accessing visual awareness”, *Evolution and Human Behavior*, vol. 38, no. 2, pp. 227–234, 2017. DOI: 10.1016/j.evolhumbehav.2016.10.004.
- [9] N. Gomes, S. C. Soares, S. Silva, and C. F. Silva, “Mind the snake: Fear detection relies on low spatial frequencies.”, *Emotion*, vol. 18, no. 6, pp. 886–895, 2018. DOI: 10.1037/emo0000391.
- [10] T. Stein, K. Seymour, M. N. Hebart, and P. Sterzer, “Rapid Fear Detection Relies on High Spatial Frequencies”, *Psychological Science*, vol. 25, no. 2, pp. 566–574, 2014. DOI: 10.1177/0956797613512509.
- [11] R. M. Foerster, C. H. Poth, C. Behler, M. Botsch, and W. X. Schneider, “Using the virtual reality device Oculus Rift for neuropsychological assessment of visual processing capabilities”, *Scientific Reports*, vol. 6, no. 1, p. 37016, 2016. DOI: 10.1038/srep37016.
- [12] —, “Neuropsychological assessment of visual selective attention and processing capacity with head-mounted displays”, *Neuropsychology*, vol. 33, no. 3, pp. 309–318, 2019. DOI: 10.1037/neu0000517.
- [13] U. Korisky, R. Hirschhorn, and L. Mudrik, ““Real-life” continuous flash suppression (CFS)-CFS with real-world objects using augmented reality goggles”, *Behavior Research Methods*, pp. 1–13, 2018. DOI: 10.3758/s13428-018-1162-0.
- [14] I. E. Sutherland, “A head-mounted three dimensional display”, in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS ’68 (Fall, part I)*, vol. 33, ACM Press, 1968, pp. 757–764. DOI: 10.1145/1476589.1476686.
- [15] T. Nakamoto, T. Hirasawa, and Y. Hanyu, “Virtual environment with smell using wearable olfactory display and computational fluid dynamics simulation”, *Institute of Electrical and Electronics Engineers (IEEE)*, 2020, pp. 713–720. DOI: 10.1109/vr46266.2020.00094.

- [16] I. P. Tussyadiah, D. Wang, T. H. Jung, and M. C. tom Dieck, "Virtual reality, presence, and attitude change: Empirical evidence from tourism", *Tourism Management*, vol. 66, pp. 140–154, 2018. DOI: 10.1016/j.tourman.2017.12.003.
- [17] R. M. Clifford, S. Jung, S. Hoerrmann, M. Billingham, and R. W. Lindeman, "Creating a stressful decision making environment for aerial firefighter training in virtual reality", in *26th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2019 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 181–189. DOI: 10.1109/VR.2019.8797889.
- [18] S. Pedram, S. Palmisano, P. Perez, R. Mursic, and M. Farrelly, "Examining the potential of virtual reality to deliver remote rehabilitation", *Computers in Human Behavior*, vol. 105, p. 106223, 2020. DOI: 10.1016/j.chb.2019.106223.
- [19] J. Nguyen, C. Smith, Z. Magoz, and J. Sears, "Screen door effect reduction using mechanical shifting for virtual reality displays", in *Optical Architectures for Displays and Sensing in Augmented, Virtual, and Mixed Reality (AR, VR, MR)*, B. C. Kress and C. Peroz, Eds., vol. 11310, SPIE, 2020, p. 22. DOI: 10.1117/12.2544479.
- [20] C. Hoffman, "What Is the "Screen Door Effect" in VR?", *How-To Geek*, Feb. 2019, [Online; accessed 27. Oct. 2020]. [Online]. Available: <https://www.howtogeek.com/404491/what-is-the-screen-door-effect-in-vr>.
- [21] R. Beams, A. S. Kim, and A. Badano, "Transverse chromatic aberration in virtual reality head-mounted displays", *Optics Express*, vol. 27, no. 18, p. 24877, 2019. DOI: 10.1364/oe.27.024877.
- [22] K. M. Stanney, R. S. Kennedy, and J. M. Drexler, "Cybersickness is not simulator sickness", in *Proceedings of the Human Factors and Ergonomics Society*, vol. 2, Human Factors and Ergonomics Society, Inc., 1997, pp. 1138–1141. DOI: 10.1177/107118139704100292.
- [23] Y. Farmani and R. J. Teather, "Evaluating discrete viewpoint control to reduce cybersickness in virtual reality", *Virtual Reality*, vol. 24, no. 4, pp. 645–664, 2020. DOI: 10.1007/s10055-020-00425-x.
- [24] S. Thompson, "Motion Sickness in VR: Why it happens and how to minimise it", *VirtualSpeech*, Apr. 2020, [Online; accessed 27. Oct. 2020]. [Online]. Available: <https://virtualspeech.com/blog/motion-sickness-vr>.
- [25] Unity Technologies, *Unity*, [Online; accessed 2. Sep. 2020], 2020. [Online]. Available: <https://unity.com>.
- [26] —, *Scripting in Unity for experienced C# & C++ programmers | Unity*, [Online; accessed 2. Sep. 2020], 2020. [Online]. Available: <https://unity.com/how-to/programming-unity#if-you-have-c-background>.
- [27] —, *Multiplatform | Unity*, [Online; accessed 2. Sep. 2020], 2020. [Online]. Available: <https://unity.com/features/multiplatform>.
- [28] —, *Unity - Getting started with VR development in Unity*, [Online; accessed 2. Sep. 2020], 2020. [Online]. Available: <https://docs.unity3d.com/Manual/VR0verview.html>.
- [29] Valve Corporation, *SteamVR (Steamworks Documentation)*, [Online; accessed 3. Sep. 2020], 2020. [Online]. Available: <https://partner.steamgames.com/doc/features/steamvr/info>.
- [30] *SteamVR - Valve Developer Community*, [Online; accessed 3. Sep. 2020], 2020. [Online]. Available: <https://developer.valvesoftware.com/wiki/SteamVR>.
- [31] D. Takahashi, "Valve launches OpenVR dev kit for virtual reality hardware makers", *VentureBeat*, Dec. 2019, [Online; accessed 3. Sep. 2020]. [Online]. Available: <https://venturebeat.com/2015/04/30/valve-launches-openvr-dev-kit-for-virtaul-reality-hardware-makers>.
- [32] Valve Corporation, *SteamVR Unity Plugin*, [Online; accessed 3. Sep. 2020], 2020. [Online]. Available: https://valvesoftware.github.io/steamvr_unity_plugin.
- [33] E. Carl, A. T. Stein, A. Levihn-Coon, J. R. Pogue, B. Rothbaum, P. Emmelkamp, G. J. Asmundson, P. Carlbring, and M. B. Powers, "Virtual reality exposure therapy for anxiety and related disorders: A meta-analysis of randomized controlled trials", *Journal of Anxiety Disorders*, vol. 61, pp. 27–36, 2019. DOI: 10.1016/j.janxdis.2018.08.003.

- [34] C. Botella, J. Fernández-Álvarez, V. Guillén, A. García-Palacios, and R. Baños, “Recent Progress in Virtual Reality Exposure Therapy for Phobias: A Systematic Review”, *Current Psychiatry Reports*, vol. 19, no. 7, pp. 1–13, 2017. DOI: 10.1007/s11920-017-0788-4.
- [35] A. Garcia-Palacios, C. Botella, H. Hoffman, and S. Fabregat, “Comparing acceptance and refusal rates of virtual reality exposure vs. in vivo exposure by patients with specific phobias”, *Cyberpsychology and Behavior*, vol. 10, no. 5, pp. 722–724, 2007. DOI: 10.1089/cpb.2007.9962.
- [36] S. Bouchard, S. Dumoulin, G. Robillard, T. Guitard, E. Klinger, H. Forget, C. Loranger, and F. X. Roucaut, “Virtual reality compared with in vivo exposure in the treatment of social anxiety disorder: A three-arm randomised controlled trial”, *British Journal of Psychiatry*, vol. 210, no. 4, pp. 276–283, 2017. DOI: 10.1192/bjp.bp.116.184234.
- [37] D. Freeman, P. Haselton, J. Freeman, B. Spanlang, S. Kishore, E. Albery, M. Denne, P. Brown, M. Slater, and A. Nickless, “Automated psychological therapy using immersive virtual reality for treatment of fear of heights: a single-blind, parallel-group, randomised controlled trial”, *The Lancet Psychiatry*, vol. 5, no. 8, pp. 625–632, 2018. DOI: 10.1016/S2215-0366(18)30226-8.
- [38] A. Miloff, P. Lindner, P. Dafgård, S. Deak, M. Garke, W. Hamilton, J. Heinsoo, G. Kristoffersson, J. Rafi, K. Sindemark, J. Sjölund, M. Zenger, L. Reuterskiöld, G. Andersson, and P. Carlbring, “Automated virtual reality exposure therapy for spider phobia vs. in-vivo one-session treatment: A randomized non-inferiority trial”, *Behaviour Research and Therapy*, vol. 118, pp. 130–140, 2019. DOI: 10.1016/j.brat.2019.04.004.
- [39] C. Suso-Ribera, J. Fernández-Álvarez, A. García-Palacios, H. G. Hoffman, J. Bretón-López, R. M. Baños, S. Quero, and C. Botella, “Virtual Reality, Augmented Reality, and in Vivo Exposure Therapy: A Preliminary Comparison of Treatment Efficacy in Small Animal Phobia”, *Cyberpsychology, Behavior, and Social Networking*, vol. 22, no. 1, pp. 31–38, 2019. DOI: 10.1089/cyber.2017.0672.
- [40] K. R. Gujjar, A. van Wijk, R. Kumar, and A. de Jongh, “Efficacy of virtual reality exposure therapy for the treatment of dental phobia in adults: A randomized controlled trial”, *Journal of Anxiety Disorders*, vol. 62, pp. 100–108, 2019. DOI: 10.1016/j.janxdis.2018.12.001.
- [41] J. L. Maples-Keller, C. Yasinski, N. Manjin, and B. O. Rothbaum, “Virtual Reality-Enhanced Extinction of Phobias and Post-Traumatic Stress”, *Neurotherapeutics*, vol. 14, no. 3, pp. 554–563, 2017. DOI: 10.1007/s13311-017-0534-y.
- [42] T. Broady, A. Chan, and P. Caputi, “Comparison of older and younger adults’ attitudes towards and abilities with computers: Implications for training and learning”, *British Journal of Educational Technology*, vol. 41, no. 3, pp. 473–485, 2010. DOI: 10.1111/j.1467-8535.2008.00914.x.
- [43] N. Hauk, J. Hüffmeier, and S. Krumm, “Ready to be a Silver Surfer? A Meta-analysis on the Relationship Between Chronological Age and Technology Acceptance”, *Computers in Human Behavior*, vol. 84, pp. 304–319, 2018. DOI: 10.1016/j.chb.2018.01.020.
- [44] H. Huygelier, B. Schraepen, R. van Ee, V. Vanden Abeele, and C. R. Gillebert, “Acceptance of immersive head-mounted virtual reality in older adults”, *Scientific Reports*, vol. 9, no. 1, p. 4519, 2019. DOI: 10.1038/s41598-019-41200-6.
- [45] L. Appel, E. Appel, O. Bogler, M. Wiseman, L. Cohen, N. Ein, H. B. Abrams, and J. L. Campos, “Older Adults With Cognitive and/or Physical Impairments Can Benefit From Immersive Virtual Reality Experiences: A Feasibility Study”, *Frontiers in Medicine*, vol. 6, p. 329, 2020. DOI: 10.3389/fmed.2019.00329.
- [46] P. Kourtesis, S. Collina, L. A. Dumas, and S. E. MacPherson, “Technological Competence Is a Pre-condition for Effective Implementation of Virtual Reality Head Mounted Displays in Human Neuroscience: A Technological Review and Meta-Analysis”, *Frontiers in Human Neuroscience*, vol. 13, p. 342, 2019. DOI: 10.3389/fnhum.2019.00342.
- [47] J. C. M. Figueroa, R. A. B. Arellano, and J. M. E. Calinisan, “A comparative study of virtual reality and 2D display methods in visual search in real scenes”, in *Advances in Intelligent Systems and Computing*, vol. 591, Springer Verlag, 2018, pp. 366–377. DOI: 10.1007/978-3-319-60591-3_33.
- [48] B. Olk, A. Dinu, D. J. Zielinski, and R. Kopper, “Measuring visual search and distraction in immersive virtual reality”, *Royal Society Open Science*, vol. 5, no. 5, p. 172331, 2018. DOI: 10.1098/rsos.172331.

- [49] J. Brookes, M. Warburton, M. Alghadier, M. Mon-Williams, and F. Mushtaq, “Studying human behavior with virtual reality: The Unity Experiment Framework”, *Behavior Research Methods*, pp. 1–9, 2019. DOI: 10.3758/s13428-019-01242-0.
- [50] S. C. Soares, B. Lindström, F. Esteves, and A. Öhman, “The hidden snake in the grass: Superior detection of snakes in challenging attentional conditions”, *PLoS ONE*, vol. 9, no. 12, H. Nishijo, Ed., e114724, 2014. DOI: 10.1371/journal.pone.0114724.
- [51] S. C. Soares, “The lurking snake in the grass: Interference of snake stimuli in visually taxing conditions”, *Evolutionary Psychology*, vol. 10, no. 2, pp. 187–197, 2012. DOI: 10.1177/147470491201000202.
- [52] S. C. Soares and F. Esteves, “A glimpse of fear: Fast detection of threatening targets in visual search with brief stimulus durations”, *PsyCh Journal*, vol. 2, no. 1, pp. 11–16, 2013. DOI: 10.1002/pchj.18.
- [53] R. Blake and N. K. Logothetis, “Visual competition”, *Nature Reviews Neuroscience*, vol. 3, no. 1, pp. 13–21, 2002. DOI: 10.1038/nrn701.
- [54] P. Sterzer, T. Hilgenfeldt, P. Freudenberg, F. Bermpohl, and M. Adli, “Access of emotional information to visual awareness in patients with major depressive disorder”, *Psychological Medicine*, vol. 41, no. 8, pp. 1615–1624, 2011. DOI: 10.1017/S0033291710002540.
- [55] M. Zhan, R. Hortensius, and B. De Gelder, “The body as a tool for anger awareness-differential effects of angry facial and bodily expressions on suppression from awareness”, *PLoS ONE*, vol. 10, no. 10, M. Ptito, Ed., e0139768, 2015. DOI: 10.1371/journal.pone.0139768.
- [56] K. Schmack, J. Burk, J. D. Haynes, and P. Sterzer, “Predicting Subjective Affective Salience from Cortical Responses to Invisible Object Stimuli”, *Cerebral Cortex*, vol. 26, no. 8, pp. 3453–3460, 2016. DOI: 10.1093/cercor/bhv174.
- [57] T. Stein, M. N. Hebart, and P. Sterzer, “Breaking continuous flash suppression: A new measure of unconscious processing during interocular suppression?”, *Frontiers in Human Neuroscience*, vol. 5, p. 167, 2011. DOI: 10.3389/fnhum.2011.00167.
- [58] S. Gayet, S. Van Der Stigchel, and C. L. Paffen, “Breaking continuous flash suppression: Competing for consciousness on the pre-semantic battlefield”, *Frontiers in Psychology*, vol. 5, p. 460, 2014. DOI: 10.3389/fpsyg.2014.00460.
- [59] W. R. Miles, “Ocular dominance in human adults”, *Journal of General Psychology*, vol. 3, pp. 412–430, 1930. DOI: 10.1080/00221309.1930.9918218.