ICVGoggles: Wearable Personalised Simulations of Impaired Colour Vision

Babak Momen

AC40001 Honours Project  
BSc (Hons) Applied Computing   
University of Dundee, 2016  
Supervisor: Dr. David Flatla

**ABSTRACT**

Colour is used to tie in closely with a specific meaning, signal or message; however it is mostly used for aesthetics. People with Impaired Colour Vision (ICV) come across challenges every day to distinguish between colours, designers must consider users with ICV since information can be misinterpreted or even missed. Current software and hardware solutions provide real time simulations of various spectrums of impaired colour vision. This project aims to go above and beyond current technologies to provide designers with adjustable simulations viewed with a hands-free Oculus Rift headset.

# INTRODUCTION

## Background

Most cases of Impaired Colour Vision (ICV) are hereditary meaning it is passed on genetically from parent to offspring. They can also occasionally be acquired as a result of certain eye diseases. Failing to discriminate between red and green is the most common form of ICV (Protanopia / Deuteranopia) and the gene is X- linked recessive which explains the prevalence difference between genders (8% in males and 0.5% in females). Blue-yellow ICV or Tritanopia is rare and tritanomalous symptoms are more commonly acquired from environmental factors such as age, where the eye lens becomes increasingly yellow over time, cataracts or trauma to the front or the back of the head. Monochromacy is even rarer and affects two or more types of cone in the eye. Colour vision can be said to be an illusion created by the interactions of billions of neurons in our brain[[1]](#footnote-1), we do not all perceive colours the same way. There is an information gap for designers about ICV, many do not know the problems some users will suffer because of their colour choices. Impaired colour vision varies in severity and because of this, it affects people differently; sometimes it is unnoticeable. Figure 1 was created using an online tool and demonstrates different types of ICV, however just like most other tools available, these images show the maximum severity only.



Figure 1 - Left-to-right, top-to-bottom: Normal colour vision, Protanopia, Deuteranopia, Tritanopia.

Designers do not normally consider ICV in their design practice; they are a small demographic of users. Without doing so, designers run the risk of creating a colour palette which will be confusing to some users. Small measures such as colour checking and ICV simulations can vastly improve the users’ experience.

## Available Products

When designing for users, it can be useful to view content through the eyes of ICV users. This can reveal poor colour choices which can lead to confusion and even missed information. There are a plethora of applications available on many different platforms which can detect and manipulate pixels to simulate ICV. One flaw most software applications present when simulating ICV is the exclusion of environmental factors such as room brightness.

### Spectrum for Chrome

It is possible to obtain a browser add-on for Google Chrome called “Spectrum” which simulates ICV for the current web page[[2]](#footnote-2). This extension enables the user to select a type of ICV and see what the webpage looks like; it is especially helpful when looking at data visualisations where colours could be misinterpreted. Spectrum does not allow the user to adjust the severity of the condition specified and so it cannot simulate weak/mild ICV. This add-on is also limited to web pages only.

### Vischeck

Vischeck[[3]](#footnote-3) provides different toolsets for simulating ICV, some are available to use online whilst others require downloading and installing. The web version of Vischeck lets users upload images and see them through the eyes of an ICV user. There is also a Photoshop plugin which enables ICV simulations within the application. Vischeck does not provide a real-time ICV simulation tool however.

### Colourblind Vision

Colourblind Vision[[4]](#footnote-4) is an application developed for Android smartphones. It was created by Bradley C. Grimm and provides real-time ICV simulations using the smartphones back camera. The application also enables users to simulate ICV on images found on the camera reel. There are similar applications available for iPhone users, showing there is an interest in simulating ICV on handheld devices.

## Social Context

It can be difficult to understand how ICV affects people; emulating the effects on images is useful, however this confines the condition within just the images. ICVGoggles aims to provide a simulation tool which works in real time giving users a peek at ICV 'in the wild'. Once complete, ICVGoggles can be applied to different social contexts in order to raise awareness and to educate users. For example, the tool could be used on trainee teachers to educate them in how ICV affects people. This sort of training could eventually help teachers to recognise comments made by children who have ICV; the earlier ICV is diagnosed, the better. Another application could be the use of the tool on parents who have children with ICV to give them a rich learning experience.

## Scientific Context

By providing an interactive tool which simulates ICV in real time, this project can be used to inform and educate. Teaching ICV can be a difficult task depending on who is learning; it may be more effective to provide a hands-on tool which can show how it affects people. The severity value can be adjusted by the user; many simulation techniques show the most severe forms of ICV only instead of incrementing or decrementing the severity. Because of this, ICVGoggles is a powerful tool to raise awareness for ICV.

## Previous Work

ICVGoggles is a project which has inherited work from previous research. The colour swapping in ICVGoggles is based off of CVDSimulation, a Processing application created by Dr. David Flatla. CVDSimulation provides the ground work for ICVGoggles and is itself based on the paper “A Physiologically-based Model for Simulation of Color Vision Deficiency” [5] and a tutorial[[5]](#footnote-5). The tutorial contains a large 4 dimensional array which contains float values. By applying the correct formula to red, green and blue pixels with these values Protanopia, Deuteranopia and Tritanopia ICV can be simulated.

CVDSimulation's functionality is very similar to what ICVGoggles aims for. The application lets the user select a webcam and then begins streaming. Protanopia, Deuteranopia and Tritanopia are the three types of ICV which can be freely switched between. The severity can also be adjusted, giving the users a sense of how it can affect people differently. CVDSimulation is available to use inside the ICVGoggles GitHub repository.

# SPECIFICATION

## The Problem

The process of simulating ICV has been around for a while and there is no shortage of applications and tools which can do so. What many of these tools cannot do is simulate the surroundings of the user in real-time, this is the problem ICVGoggles aims to solve.

Currently most ICV simulation tools provide a solution to specific problems and have downfalls when applied in other areas. For example, Vischeck provides developers with a way to see their work through the eyes of an ICV user, however it does not take external lighting or screen brightness into consideration. Other solutions such as mobile applications[[6]](#footnote-6) are restricted by the hardware they are on; users must always use one hand to move the mobile device around.

## Motivation

It is becoming more commonplace for applications to have colour blind settings, however there are very few which display ICV simulations in real time. ICVGoggles aims to take simulating ICV to the real world and will use an Oculus Rift headset to do so. This will result in a simulation tool which will give the user both hands to operate the world around them. The tool could be used for designing both digital and physical products, or it could be used to educate and teach. ICVGoggles will be open to many different routes when it is complete.

## Feature List

When planning ICVGoggles, the student and supervisor discussed technical features the software would have. Ideas were bounced back and forth; it was decided to make sure the system was extensible for the future. With that in mind, the student programmed from the ground up and made sure more could be added at any time.

### Display Camera Feed in Real Time

The first feature to initiate the project was to display the OVRVision cameras on the Oculus Rift. This task would set the foundation for ICVGoggles's code. Once the surroundings are within the headset, the next task would be to start doing image manipulation.

### Monochromacy Simulations

After displaying the camera feed, the next task involves simulating Monochromacy. This type of ICV was chosen first as it is relatively simple to program in OpenFrameworks. Since the framework gives access to RGB values in pixels, simulating Monochromacy is done by just invalidating two colours from red, green or blue. Each frame in the camera feed is treated as an image and the pixel adjustments are made to every frame taken in.

### Protan, Deutan and Tritan Simulations

By referring to CVDSimulation, a Processing application developed by Dr. David Flatla, the student would go on to create simulations for different types of ICV. The colour value swap matrices mentioned in *Section 1.5 Previous Work* would be the basis of how to interact with the system. The float array takes in two values from the user, type and severity. By adjusting these values in real-time, the user can actively increment and decrease severity and switch types.

### Control Scheme

With the Oculus Rift on the users face, it can be quite difficult to navigate and press keys on a keyboard. The student took this into consideration when creating the controls for ICVGoggles (See Figure 2). Switching between types uses the numbers 1 - 4 on the keyboard; when deployed in the lab, sticky tack was applied to the keys to give them extra tactility. Pressing Spacebar toggles the heads up display as it is an easy to locate key with a large surface area. Finally, left and right click on the mouse controls the severity value. The severity can range from 0 - 10 and each increment affects the type of ICV displayed.

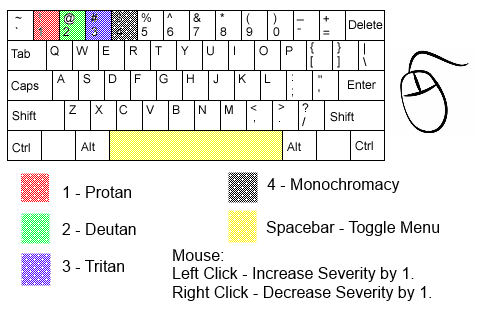


Figure 2 - Proposed Control Scheme for ICVGoggles

### Heads up Display (HUD)

After the simulations and control scheme are completed, the next stage would be to create a HUD to display information to the user. The basic HUD for the project deadline contains information about the current settings and keyboard controls; future versions will become more complex when doing personalised simulations.

## Project Plan

This projects lifecycle spanned over seven months and so it was necessary for the student to correctly plan out their course of action. The weekly meetings set up with the supervisor ensured progress could be monitored at regular intervals.

### Work Schedule

The project began on the 5th October 2015 with a meeting with Dr. David Flatla. It was decided that the student should have weekly meetings with their supervisor to ensure all progress was being monitored. The weekly meetings also meant the student could assign smaller deadlines on a weekly basis. Notes were taken at every session and have been compiled intoAppendix A. These minute meetings contain thoughts gathered with the supervisor and were used to plan work and to be reflected upon for the following week. A GitHub repository was set up at the start to store all work done whilst the commits and punch card feature proved useful to track progress. For the mid-term progress hand in, the student produced a Gantt chart which would serve as a rough schedule.

### Deliverables

The student created deliverables in the Gantt chart seen in Appendix B. These were spread out over the course of the project lifecycle and extra time was given for all tasks to give the student some leeway if any complications occurred. Initially the deliverables were centred on the ethics submission which required multiple consent documents to be created and checked by the Ethics Committee. These documents included consent forms for the prestudy interviews and second stage evaluations, both for ICV and Non-ICV participants.

After these were completed, the student then moved onto producing the code for the project. As seen in the Gantt chart, this production was done over the course of several months whilst both user studies were taking place. The plan for the programming was created before any coding had begun, therefore the student made sure the tasks were split into manageable, realistic deadlines. The progression of coding is also seen on the chart and each deliverable is a logical increment towards the final goal.

### Resources

The student took advantage of resources available. Labs within the Queen Mother Building at Dundee University were used for both the prestudy interviews and the second stage evaluations. Within the labs is a PC which contains the Oculus Runtime and this enables the use of ICVGoggles.

Whilst studying in their final semester, the Student chose the ECVD Research Frontiers module to further aid in their understanding of colour vision deficiencies. This module covered many research papers to do with ICV and discussions were held on a weekly basis with the class to cover them in detail.

# DESIGN

## Decisions and Trade-offs

The student required a platform to develop ICVGoggles on. The Oculus SDK is available for Visual Studio and for Unity; both platforms could be used for the project. It was decided by both the student and their supervisor to write the software in C++ on Visual Studio as opposed to within Unity. This is because Unity was designed to create games; even though a simulation tool is entirely possible, it would have still been created in a platform centred on a different technical area. This means there would have been a large amount of unused overhead which can cause system slowdown if the application was not optimised correctly. On the other hand, projects created with C++ in Visual Studio contain just the essentials in order to get up and running. Since the system will be doing image manipulation at around twenty times per second, it is necessary to ensure there is as little overhead as possible to maintain a respectable frame rate.

Due to the timescale of the project, the student and supervisor felt it would be best to postpone the production of personalised simulations until the adjustable ones were complete. By doing so, it gave the student enough time to finish the adjustable simulations for user testing. The personalised simulations have been added to future work.

## Software & Hardware

### C++ in Visual Studio

The coding language of choice for ICVGoggles is C++ since it is what the Oculus SDK is coded in. C++ is a middle level language and is extremely versatile; it allows for low-level memory manipulation [1] which is ideal for changing RGB values efficiently. The student had not touched upon this programming language for a few years and so it was important to solidify the fundamentals early through revision. Thankfully OpenFrameworks contains a project generator which aided in starting up greatly. Visual Studio was chosen as it is the development platform the student is most confident with.

### OpenFrameworks

Coding ICVGoggles with just C++ and the Oculus SDK is a difficult task without the help of a framework. OpenFrameworks is an intuitive open-source C++ framework designed to assist creative processes [2]. The toolkits design philosophy is DIWO (Do it with Others), where creating content is done as a community. This proved helpful during the design process as members of the online forum actively aided in progressing with some small issues for ICVGoggles. OpenFrameworks is very similar to the programming language Processing [3]; a language the student had touched upon in previous years of study. Because of this, it was easier for the student to learn the framework structure and documentation.

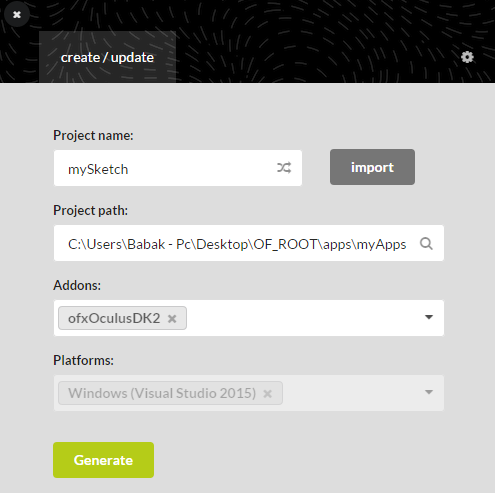


Figure 3 - OpenFrameworks Project Generation

OpenFrameworks comes with a project generation tool. After searching and finding which IDE's are installed on the machine, the tool allows the user to select additional add-ons to include in the project. For ICVGoggles, the only add-on used was ofxOculusDK2.

### Oculus SDK v0.6

In order to utilise the Oculus Rift, the software development kit (SDK) must be installed. This large package contains all code required to use the headset. The Oculus Rift is mostly used for Virtual Reality (VR) and so many code examples and tutorials cater for this. ICVGoggles only requires feeding a camera stream to the headset and so the student decided to use a community add-on which facilitated this functionality. It was important for the student to use the SDK version 0.6 as the community addon required this specific version.

### ofxOculusDK2 Add-on

Since OpenFrameworks is open-source, there is a vibrant and active community which revolves around it. It is commonplace for developers to create helpful packages and share them amongst the community. ofxOculusDK2 (OpenFrameworks Oculus Developer Kit 2) was created by developer Andreas Muller and refactored by James George [4]. This add-on accesses certain parts of the Oculus SDK to enable basic rift rendering functionality, giving the student a lightweight gateway to simulate ICV. The fish eye distortion is applied automatically to anything drawn on the left or right eye, meaning the student had one less task to worry about.

### Oculus Rift

The Oculus Rift was chosen as the headset for ICVGoggles to use; specifically the Dev Kit 2 which was released on March 25th 2014. The headset has an OLED display and uses a positional tracking system which enables the movement within a 3D space. For ICVGoggles, both eyes will be drawing and manipulating a camera feed to simulate the real world.

### OVRVision Cameras

In order to simulate the world in real-time, cameras are required for the Oculus Rift. OVRVision was created by a small tech start-up in Japan and provides two cameras to be mounted on the Oculus Rift. OpenFrameworks was able to detect both mounted cameras and so it was then possible to continue coding.



Figure 4 - The Oculus DK2 with OVRVision Cameras attached.

### Oculus Runtime v0.7

The Oculus Runtime is required to detect the Oculus HMD. This piece of software is essential to run ICVGoggles as it allows the Oculus Rift HMD to become a secondary monitor. It is important for users to have the Oculus Runtime <0.7 as the secondary monitor feature was removed from future versions.

## Design Process

### C++ Setup, Update & Draw

OpenFrameworks projects have a default template which contains the basic structure of an application. Before the student could start designing the software for ICVGoggles, they had to learn the structure OpenFrameworks applications follow. Figure 5 shows the basic templates for a new project. When an application is built and ran, the main method is the first to be initiated. From here, an instance of the application begins and the code flow moves to ofApp.cpp.

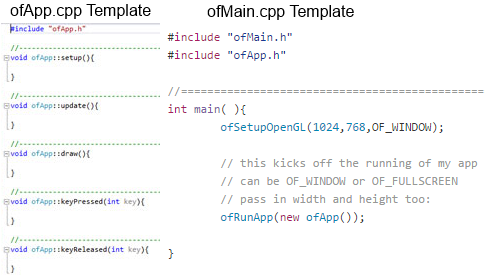


Figure 5 - OpenFrameworks ofApp.cpp & ofMain.cpp Templates

The setup method initialises properties and is ran only once. From here, the update and draw methods loop endlessly, creating a running application. After understanding how applications operate in OpenFrameworks, the student then applied their past knowledge of Processing and C++ to design each feature from the specification.

### Feature Driven Development Methodology

The feature driven development (FDD) methodology [6] was the most suitable for ICVGoggles as the template for OpenFrameworks projects develops the overall model automatically. The feature list (See 2.3 Feature List) was designed by the supervisor and student. Since this stage of the development process was already completed, the project could progress to planning each features design. The features were ordered sequentially since they depend on previous features to operate correctly. For example, the camera feed must be functional in order for the student to implement ICV simulations. From here on, the student developed features and frequently committed work to the GitHub repository (See Appendix G - GitHub Commit Statistics).

### ICVGoggles Workflow

The workflow for ICVGoggles is seen in Appendix C and shows how the code works in real time. Once initialised, the application runs the setup code and some variables are set. Type and severity are both integers which can be used in to locate variables in the 4D float array. The first dimension has a size of three and this represents the three simulations available: protanopia, deuteranopia and tritanopia. The second dimension has a size of ten and this represents the severity which ranges from one to ten. By having access to changing the severity and type while the application is running, the user can change simulations in real-time.

## Ethics & User Testing

The student and supervisor agreed that user testing should be done with this project. Initially the student was going to do quantitative tests with and without the headset to gauge its accuracy and effectiveness. However the focus changed to qualitative data from participants, as both the student and supervisor felt there would be more rich information given this way.

Eventually the student finalised their user testing plans. Whilst the application is in development, the student would begin doing Pre study Interviews with designers and programmers. In these short interviews, data would be gathered about the participants’ current understanding of ICV and to invite them to the second stage of testing.

When the application is able to simulate adjustable ICV, the participants will be asked to return to try out ICVGoggles. The second stage evaluations are more in depth than the prestudy interviews and participants gain the opportunity to play about with the settings and view their surroundings. For both parts of user testing, the student will be recording the session with a video camera. The participants must read the ethics form and consent must be gained before recordings can take place.

The ethics documentation can be found in Appendix G and all of these forms have been reviewed and accepted by the Ethics committee at the University of Dundee. Explicit consent is required and a copy of all forms signed must be given to participants. When referring to their data, the student guaranteed anonymity.

# IMPLEMENTATION & TESTING

## Production

Soon after the ethics documentation had been completed and sent off, production of the code began. Code was only committed to the repository if the project could be built successfully and if it worked as intended. The student spent the first few weeks setting up the development environment. This was an unnecessarily long process as it took a while for the student to discover the ofxOculusDK2 addon only worked on Windows 8.1 or below. Since the student was developing on Windows 10, they had to partition their hard drive for a previous version. Eventually the project was set up and all of the header files from addons were linking correctly.

### Global Variables & Setup Function

The global variables and objects for ICVGoggles are declared in both Main.cpp and within ofApp.cpp. These include severity, type, ofVideoGrabber (a webcam device), the Oculus HMD, overlay bool and size dimensions. Some of these need to be actively changed whilst the application is running, hence why they have been made globally accessible. The most important object set is the 4D float array, mult. This is how the application can determine which colour values to display depending on the type and severity set.

The setup function is the start of the application and is where the environment is initialised. Firstly, the code forces itself into fullscreen since the Oculus Rift has difficulty rendering windowed applications. Then the system sets the default variables such as zero for severity and type and overlay as false. Lastly, the application creates a char \* array to store each individual pixel on a texture with dimensions set earlier.

### Camera Feed on Oculus Rift

The oculusRift object is the end result of the ofxOculusDK2 addon; it provides several basic functions to interact with the Oculus Rift in OpenFrameworks. This object has its base camera set to an ofCam (OpenFrameworks Cam object), which initiates the canvas for the Oculus to render. Once the setup method has finished, the update and draw methods loop multiple times a second.

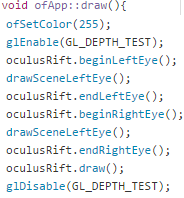


Figure 6 - ofxOculusDK2 methods used

Functions from the addon ofxOculusDK2 are used in the draw method of the application as seen in Figure 6. Each eye is begun individually, with a drawSceneLeftEye method called before they end. This method simply draws whatever is loaded into the mirror texture, in this case, the camera feed.

The camera feed is updated multiple times per second in the update method. Before simulations were included, the update method began with just the allocation of camera feed frames to a texture.

### Pixel Colour Splitting

The camera feed was now displaying on the Oculus Rift, however the output was horizontally flipped. After browsing the framework forums, the student found a solution to this and a way to progress with simulations. A user pointed out the frames can be split into their pixels and these pixels can be split into their red, green and blue values. The application checks if there is a new frame, if so the entire image is stripped down and the pixels are reordered to horizontally flip itself.

### Monochromacy Simulation

With the applications now accessing each pixels RGB values, the student added a keyPressed option to switch between the types. If the type value became 3, the update method would specifically change the pixels to become monochrome using a luminosity method found from "Converting to Greyscale" by John D. Cook[[7]](#footnote-7), a blog post on the internet. The value range for RGB is 0 - 255 and all colours on screens are compromised of three values between that range. By doing certain multiplications, accurate monochromacy can be simulated. Initially the student simply removed two of the three values being returned to simulate monochromacy, however the values from John D. Cook produce a more accurate representation.

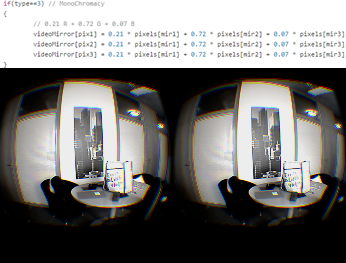


Figure 7 - Luminosity method values applied to RGB values of each pixel & example monochromacy simulation.

### Protan, Deutan and Tritan Simulations

Once the Monochromacy simulations were working correctly, the student moved onto simulating other types of ICV. The calculations for each colour starts at line 178 in ofApp.cpp and these were heavily influenced from those done in the CVDSimulation Processing application. The correctValue function which wraps around each pixel calculation is to ensure the char value does not exceed 255 or go below zero. This prevents integer overflow, which causes certain areas of the image to change into a bright colour, mostly colours which that type of ICV should not be able to see.

The RGB values are calculated by multiplying the pixels’ RGB to values in the mult 4D array. The first two dimensions of mult were explained in 3.3.3 ICVGoggles Workflow and depend on the type and severity values. The third dimension depends on which pixel to change, be it Red, Green or Blue. The fourth dimension depends on the position of where the multiplication is taking place.

With all of these multiplications combined, the application now simulates ICV depending on what type and severity are set. Each pixel is split into its RGB components and the values are swapped depending on the settings; surprisingly the application runs efficiently and can reach thirty frames per second.

### Control Scheme Development

Deciding upon a control scheme was a difficult task for the student, as it was the first time they had considered designing for a user with a VR headset on. The control scheme proposed in 2.3.4 Control Scheme seemed the most logical. Left and right click were made to increase and decrease severity as the student felt users would change this value quite often in comparison to type. Changing the type is mapped to 1-4 on the keyboard; when the application is deployed, the keys will have a tactile object such as blue tack attached to aid the users in finding the keys.

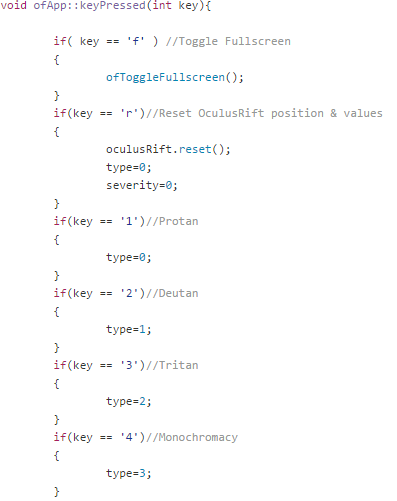


Figure 8 - ICVGoggles code for keyboard controls.

The reset functionality was added after user testing was done for the second stage evaluations. This is because Participant 6 felt it would be helpful if there was a key which would reset the values so they could see the difference instantly.

### HUD Development

Before starting user testing, the student decided to complete the HUD, since it would contain keyboard controls and helpful information. The bool named overlay is initially set to false, however when the spacebar is pressed, the state of overlay toggles. When overlay is true, the oculusRift.beginOverlay method starts the drawing process. Between here and oculusRift.endOverlay, all draw methods apply to the overlay, a special area designated to move with the head positioning. This is why the overlay stays in a fixed position.

## Testing & Debugging

Since Visual Studio was being used as the development environment, ICVGoggles could be rapidly tested thanks to the debugger tool included. Once ofxOculusDK2 was included in the project successfully, the student could make minor adjustments to the draw methods and rebuild the solution. This meant progress could be constantly tested; only working code which could be built was ever committed to the GitHub repository.

During the testing process, the student discovered multiple errors, some of which were corrected. For example, without the correctValue() wrapper function, ICVGoggles returned values greater than 255 for some pixel colours. This resulted in colours being displayed erroneously. Another bug found was to do with the OVRVision cameras. Due to a USB power issue, both cameras could not be instantiated at the same time. This resulted in the student using just one camera and blending them into both eyes.

Debugging was slightly difficult as the source code could not be loaded onto any machine other than the development one in use at the student’s home. This is because C++ uses linking to other libraries, and these paths were all set for Babak – PC. When downloaded and unpackaged on another machine, the project does not load correctly since the IDE cannot find the linked libraries. This meant the student had to describe his errors with his supervisor and peers when asking for help, as opposed to showing them. Thankfully this did not hinder the student too much as the OpenFrameworks forum user base is extremely active.

## Finished Features & Release Package

With adjustable simulations working and the controls set up, the student felt the application was ready to be used in the second stage evaluations. Figure 9 compares an image captured with normal colour vision with Protanopia set on severity 10. Both eyes are shown in the images, however when the headset is on, these two images are blended together to form one image. This functionality is provided in the ofxOculusDK2 add-on. The most noticeable difference is the lack of red in the bottom image.



Figure 9 - Normal colour vision compared to Protanopia on severity 10.

The control scheme can be seen if the user presses spacebar and toggles the HUD. This can be seen in Figure 10, where the severity and type are also displayed. The images used here can be found in the GitHub repository under the folder "Images". Figure 10 also displays the Tritan simulation on severity ten; the colours turn pinkish.

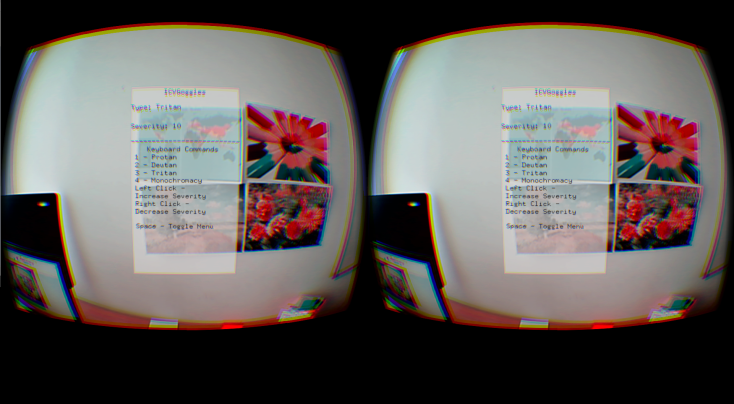


Figure - HUD in ICVGoggles.

The release package was built in Visual Studio and contains all of the necessary files to run ICVGoggles apart from the Oculus Runtime. There is also an instructions manual seen in Appendix F which contains an install guide and controls.

## Project Plan Deviations

Due to certain limitations and time constraints, the student had to deviate slightly from the project plan in order to get a working copy of ICVGoggles for the user testing. Firstly, personalised simulations are not in the release package, however it will most likely be the first feature to complete in the future. The student and supervisor felt the adjustable simulations would be an ideal feature to complete for user testing, as changing the severity would give users a sense of scale. The OVRVision cameras held the project back as it took a while for the student to find out the two cameras would not stream at the same time in OpenFrameworks. One possible solution is for the student to invest in a new set of front facing cameras, the functionality already exists and so it would be quick to code ICVGoggles to use two cameras.

# DESCRIPTION OF FINAL PRODUCT

ICVGoggles provides adjustable simulations of ICV in real-time through the use of the Oculus Rift. Users can select from four different types of ICV to simulate where their severities can be changed; watch as the world around you changes colour. The controls are simple to learn and the application provides an engaging and informative look at how some people with ICV see the world. With the headset on, users have their hands free to interact with objects, something not possible with other real-time ICV simulation apps.

Full description of final product, well worded and should NOT be neglected

Reference instruction manual

# EVALUATION

Usability should be evaluated with a description of the user-centred design methods employed to produce a usable product, including rapid prototyping, usability methods, results and re-designs as appropriate. Other relevant criteria such as accuracy and computational efficiency should also be employed for evaluation as appropriate.

## Prestudy Interviews

## -prestudy interviews, main testing (plates and exploration), questionairres

## Second Stage Evaluation

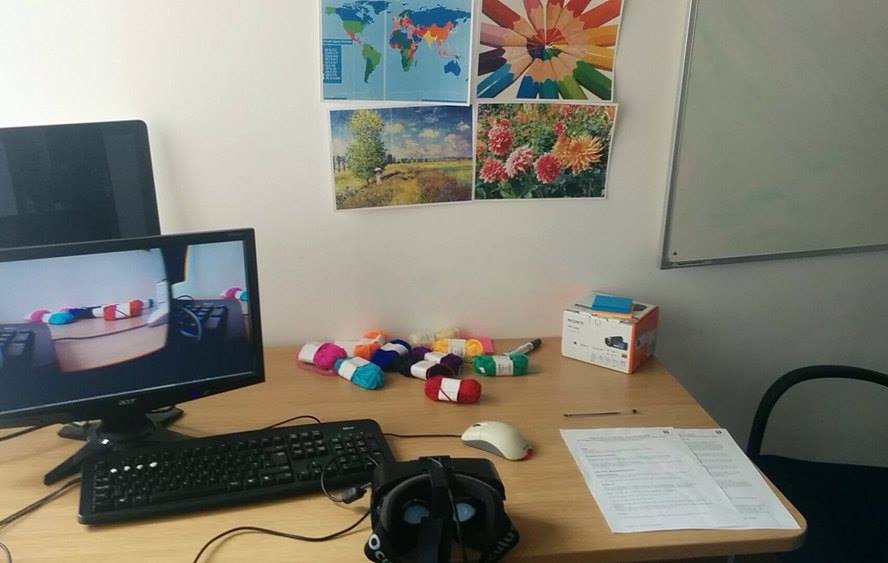


Figure 11 - ICVGoggles set up in the lab with coloured yarn balls and posters.

## Analysis

Of results. Methods used.

## Evaluation of Results

## Usability

## Accuracy (important!)

# DISCUSSION

Area where I discuss reasons for the results found and how these results may benefit ICVGoggles.

# APPRAISAL

A critical appraisal of the project indicating the rationale for design/implementation decisions, lessons learnt during the course of the project and an evaluation (with hindsight) of the final product and the process of its production (including a review of the plan and any deviations from it).

## -Rationale for design

## -Rationale for implementation decisions

## -Lessons learnt

## -Evaluation including hindsight

**A description of any research/hypothesis**

# SUMMARY & CONCLUSIONS

## Summary

## Conclusions

# Qualitative evaluation, qualitative (IF DONE) evaluation, personal feelings on project and how it went

# FUTURE WORK

-Mobile ICVGoggles (garreth said a battery back pack, possible?)

-two cameras

-personalised simulations

A copy of the mid-project progress report should be included.

# REFERENCES

1. Graham M. Seed 1996. C++ and its low level memory management. *An Introduction to Object-Oriented Programming in C++.*
2. OpenFrameworks C++ Toolkit. www.openframeworks.cc/about/
3. Processing, the flexible software sketchbook. https://www.processing.org/
4. ofxOculusDK2 OpenFrameworks Add-on. <https://github.com/obviousjim/ofxOculusDK2>
5. Gustavo M. Machado, Manuel M. Oliveira & Leandro A. F. Fernandes. Previous research used for CVDSimulation. *A Physiologically-based Model for Simulation of Color Vision Deficiency*
6. Steve R. Palmer, Mac Felsing 2001. Feature Driven Development. *A Practical Guide to Feature-Driven Development* .

# APPENDICES

Appendix A - Minute Meetings

Appendix B - Gantt Chart

Appendix C – ICVGoggles Flow Chart

Appendix D – Prestudy Transcripts

Appendix E – Second Stage Evaluation Transcripts

Appendix F - Instructions Manual

Appendix G - GitHub Commit Statistics

Appendix H – Ethics Documentation

1. P.Gouras, 'Colour Vision', in webvision.med.utah.edu, last update 1 July, 2009 [↑](#footnote-ref-1)
2. Spectrum, offered by Yehor Lvivski for Google Chrome [↑](#footnote-ref-2)
3. Vischeck, Simulation tools for web and photoshop. http://www.vischeck.com/vischeck/ [↑](#footnote-ref-3)
4. Colourblind Vision by Bradley C. Grimm. https://play.google.com/store/apps/details?id=com.givewaygames.colorblind\_ads&hl=en [↑](#footnote-ref-4)
5. Tutorial where the CVDSimulation colour value matrices can be found. http://www.inf.ufrgs.br/~oliveira/pubs\_files/CVD\_Simulation/CVD\_Simulation.html [↑](#footnote-ref-5)
6. 20 iPhone Apps for the Colour Blind - http://www.color-blindness.com/2010/12/13/20-iphone-apps-for-the-color-blind/ [↑](#footnote-ref-6)
7. Converting to Greyscale - http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/ [↑](#footnote-ref-7)