Assignment 2
Simulating colour blindness using virtual environments

Alexander Bennett
CEM242 Autumn 2018
Advanced Visualisation and Interactive Technologies
alexander.f.bennett@reading.ac.uk

ABSTRACT
A colour blindness simulator is developed within a virtual environment using Unity 3D. The simulator demonstrates in real time how building users may be affected by different types of colour blindness, allowing designers to improve building accessibility by using different materials, colours or textures to accommodate visually impaired people.

INTRODUCTION
Designers must be sensitive to a wide range of accessibility issues in order to accommodate different building users. This is particularly important within the current scope of UK legislation, such as the Equality Act 2010 [1, 2]. Significant resources are often dedicated to improving accessibility for physical impairment, such as mobility issues, but visual impairment is more difficult to understand and assess.

One such visual impairment is colour blindness, an X-chromosome linked genetic abnormality which affects a disproportionate number of men and a sizeable portion of the general population. It is estimated anywhere from 5-10% of men have a mild to severe form of protanopia or deuteranopia (commonly known as red-green colour blindness). A few rarer forms (tritanopia, known as blue-yellow colour blindness, and achromatopsia, or 'black and white' vision) affect <1% of the population, equally distributed between men and women [3, 4].

Historically, colour blindness has received attention from sectors such as graphic design or cartography, which may incorporate measures to improve readability (e.g. by avoiding colour-coded information, using corresponding symbols/text or selecting 'colour safe' pairs).

To an extent, these measures are reflected in the built environment [5]. Rail signalling, for example, typically uses safer colour pairs like red, yellow and blue-green (as opposed to the red, amber, green combination common on highways). However, awareness of the different types of colour blindness is not widespread and there is limited understanding of how to improve the built environment in this regard. Tube maps, for example, still largely rely on colour-coding [6], while emergency signage in red and green can be problematic [7, 8].

Beyond well-recognised issues, such as red-green clashes, colour blindness can also present unanticipated issues which may be virtually impossible to pick up during the design stage. While unusual, these issues are nuanced and highly situational – consider, for example, a flush panel cladding system which highlights doors in a different colour. In this case, it is unlikely that the lack of contrast for colour blind people would ever be observed by non-visually impaired designers without some method for universally checking colours in the design. The degree of difference may even vary significantly between different types of colour blindness.

Virtual environments (VE) offer an ideal platform for improved understanding of colour blindness and the impact it can have on building users. Building Information Modelling (BIM) is growing in popularity; increasingly, designers are producing 3D models during the design phase which can be adapted for a range of visualisation purposes. There is also a natural synergy with VE, as the visualisation of simulated colour blindness provides immediate feedback to non-visually impaired people, which does not require special expertise to interpret.

By simulating colour blindness real time in VE, designers can intuitively grasp the issues without reference to tables of colour-safe pairs or spectrum charts of shifted hues [9]. These tools are valuable in theory, but often clunky and impractical in application. Similarly, various web tools exist which allow for processing static images to simulate colour blindness, but these are extremely limited by speed, adaptability and integration with a typical BIM workflow.

PROJECT AIMS
The key project aim is to accurately simulate colour blindness in a real-time VE, providing a practical utility for designers to check colours during the design phase. In this scenario, a designer requires a method to assess potential issues with building accessibility for visually impaired users in order to make better decisions on the use of materials, colours, textures etc.

It is envisaged that interactivity with the VE may be scripted, to allow the user to switch from a normal full colour mode to the different simulated colour blindness

modes. Changes to the camera are preferred to directly manipulating objects in the environment (e.g. by changing their material/texture attributes), as the higher level change is more applicable to a wide range of models and use scenarios. It is important to be able to quickly and easily switch between different modes in order to make comparisons, ideally with no other noticeable differences to the VE.

Ultimately, the project aims to raise awareness of these issues, supporting so-called universal design [10].

PROJECT DESIGN

Unity 3D (version 2018.3.3f1) was selected as a convenient platform for simulating colour blindness in VE. Unity is a widely used game engine, with an active community of developers. It includes the ability to import models from commonly used design software, such as Revit or Sketchup. There is also the ability to use scripts written in C#, which allows for a range of unconventional modifications. Finally, Unity can package projects in a standalone executable (.exe) format, which means the demo project can be circulated and tested without any specialist software.

While it is impossible to see through another's eyes, colour blindness ultimately arises from a physiological anomaly in the retina. The Meyer-Greenberg-Wolfmaier-Wickline (MGWW) [11, 12] algorithm was used to simulate colour values for different types of colour blindness with a reasonable degree of accuracy [13]. This benchmark algorithm attempts to render colours based on the differing arrays of photosensitive cells in visually impaired people.

DEVELOPMENT STEPS (DEMONSTRATION PROJECT)

A thorough study of existing research and tools for visualising colour blindness was conducted. Several websites with reprocessing tools, limited to static images, were found [14, 15, 16]. Some of the Ishihara test plates [17] found at this stage were incorporated into the final build, along with a labelled colour wheel. The plate number '12' is used as a control, as it should be equally visible to colour blind people.

Most of the tools found focus on web/game design. Exploring these options led to discovery of the colour blindness filter script by Alan Zucconi [18], which was ultimately used in every version of the project (see Appendix B). Critically, this C# script was fully compatible with Unity, using a custom shader to alter RGB values in line with the MGWW algorithm [12].

Version 1.0 (**see Figure 1** in Appendix A) used VREX [19], an open-source Unity toolbox specifically designed for researchers with limited coding abilities to quickly set up experiments. There were significant compatibility issues with the latest version of Unity, requiring Unity 5.4.1f1 to run stably. Unfortunately, while this autogenerated a VE with furnished rooms as a suitable testing model, it was not possible to use custom scripts, such as the colour blindness filter, and was abandoned as a result.

Version 1.1 (**see Figure 2**) introduced a Revit model, imported to Unity as a .fbx file from a live design project. It was intended that the colours and textures would carry over (**see Figure 3**), again creating a useful prefabricated testing environment. The colour blindness script was also attached to the main camera, and worked well (**see Figure 4**). Unfortunately, it proved difficult to import the textures correctly, even when using 3ds Max as an intermediary, and there were too many individual components to apply them manually in Unity. Worse, the simulation had to be stopped in order to switch colour blindness modes using the menu (**see Figure 5**). This made it difficult to make real-time comparisons, one of the fundamental project aims.

Version 1.2 introduced a simple maze model, created in Sketchup and exported to Unity as a .fbx file (**see Figure 6**). This was much simpler to change materials and was intended to provide a good test bed for showing difficulties in navigating environments using coloured arrows. The virtual reality (VR) component was also introduced following testing in the university's visual laboratory. The SteamVR toolkit was used with the HTC Vive, and the filter was now attached to the VR camera (**see Figure 7**).

Numerous issues required the project to be partially redesigned at this stage. Initially, it was very difficult to implement VR movement, even using the prefabricated components, as there were clashes between the model floor and the invisible VR plane. It was also impossible to achieve a true-to-life appearance inside the maze, as the lighting penetrated the model seams. This could be mitigated by altering the shadow settings, but still resulted in a strange appearance which reduced the simulation fidelity. Finally, it was difficult to create coloured arrows in the model, whether as additional objects or by modifying the materials/textures, and the simulation still had to be stopped to change the camera mode.

Version 1.3 was the last major revision leading to the final build, and several minor changes were made at this stage. Firstly, the VR component was removed and the camera type/user controls were reverted to the first person prefab included in the Unity Standard Assets pack. This was largely due to the difficulty in testing the VE without access to the university's VR equipment, as the simulation would not show changes to the camera without the headset. The second key script was also introduced, allowing the camera to be changed with a key press. This used an array which referenced several duplicate cameras, each running its own colour blind script and each set to a different colour blindness mode (**see Figure 8**). This was an elegant solution, allowing the camera mode to be changed seamlessly without stopping the simulation.

Unfortunately, the camera array also created several issues. Firstly, altering any of the duplicate cameras' position or rotation without selecting the parent object effectively broke the mouse and keyboard controls. It was not clear

what was causing the issue, and this took many attempts to resolve. Secondly, the normal vision camera was able to rotate freely on the X and Y axes, mapped 1:1 to the mouse movement as expected. However, the duplicate cameras (with identical settings) were restricted to the X axis, causing a significant mapping failure when changing modes in real time. It is still unclear why this would occur, but the normal camera was restricted to the X axis as a workaround (**see Figure 9**).

To complete the demonstration, the maze model was replaced with a professionally textured model of a gallery [20], complete with custom images to provide instructions and to highlight some examples of colour blindness affecting user perception (**see Figure 10**). A custom skybox was also added to improve the simulation fidelity and to provide a control for the colour blindness modes.

Finally, there was a significant issue which prevented the colour blindness filter from working at all in the final build .exe. By interrogating the hidden Unity player debug log, a Null Reference Exception error was found and traced back to the custom shader which manipulates the RGB values. In the end, it was discovered that this shader was omitted from the build package due to a bug in Unity, which means that custom shaders are not included in the packaged file unless explicitly named in the project settings (**see Figure 11**). It is notable that this bug occurred even though there were no compiler errors with the scripts or any other issues evident within the Unity project files, thus demonstrating that inexplicable issues can arise which require a level of coding/IT familiarity beyond most design practices.

PROJECT LIMITATIONS

Significant limitations were introduced by the selected tools. Initially, the biggest constraint to developing the tool in VR and exploring different solutions was the need for specialist equipment, such as the HTC Vive setup and the powerful graphics processing computer only available in the university's visual laboratory. That said, although the cost of this equipment is currently significant on a consumer/end user scale, there is potentially much more value for money to a professional design practice. In the end, the decision was made to present the final project demonstration without the VR component, instead using a more conventional first person camera rig commonly found in video games.

The option remains to reinsert the VR toolkit, enabling the full VE experience if required, though it is notably difficult to develop simulations for VR without immediate use of specialist VR equipment. In practice, while the non-VR project is less immersive and convincing at simulating colour blindness, most of the benefits to designers are present in the basic project design, partially overcoming this limitation. Interestingly, there are also potential accessibility issues in restricting the simulation to VR use, as existing equipment can be bulky and difficult for some people to use (particularly visually impaired people!) [21].

The next most significant limitation was the requirement to use highly sophisticated and specialised software (in this case Unity, as well as 3ds Max and Sketchup for tweaking models), along with an element of coding in C#. Numerous issues arose relating to the correspondence between the location of the camera and the user model, the fluid switching between colour blind and full colour modes (which was critical to the project aims), and the user controls. Particularly frustrating was the inability to freely rotate the camera on the Y-axis on the duplicate camera objects used for colour blind modes. Various attempts to free up the movement of the camera failed, and the reason for this lack of functionality is still unclear. In the end, the decision was made to lock the normal full colour camera to X-axis only, which ensured it always matched the other cameras 1:1. The negative consequence of this limitation is particularly evident when attempting to descend the stairs in the demonstration, as there is a natural (in this case, unnecessary) instinct to look down at the steps; the inability to do so reduces the environment fidelity.

While these issues are not insurmountable, they certainly represent a barrier to entry. Design practices may have BIM technicians, but they are currently unlikely to have the requisite computer skills in-house to resolve complex software/coding issues. In this instance, the final packaged .exe at least allows for widespread distribution and use of the project demonstration without the need for any specialist software or coding ability. However, designers would still need to become familiar with Unity in order to test their own models in the simulated colour blind VE.

There are also some more practical limitations relating to use scenarios. Critically, the project relies on existing use of BIM during the design process; the benefits of simulating colour blindness in VE do not by themselves merit the additional resource investment in BIM. Currently, this may prove onerous for small practices, or simply not cost-effective for larger practices on low value design projects. Nevertheless, it is anticipated that the uptake of BIM will continue to increase as the technology and associated skills become more widely available. In turn, this project should offer increased relevance to the typical design scenario.

OPPORTUNITIES FOR FUTURE RESEARCH

Opportunities can be divided into two distinct groups: practical applications/extensions to the research project and improvements to the simulation VE.

In terms of practical applications, it would be useful to test an actual BIM workflow to assess the usefulness of the tool in a typical design process. An early version of the project demonstration included an imported Revit model from a live design project, but it proved difficult and time-intensive to correctly import textures to Unity. This early test was abandoned, as improvements to workflow were needed to test a complex live model (as opposed to the relatively simple demonstration model).

At the outset, when defining project aims, it was also envisaged that the simulation could be extended to other forms of visual impairment, such as partial blindness, cataracts etc. [22] This could be a fruitful area for further research, particularly as certain types of colour blindness such as achromatopsia are often associated with partial blindness. In these cases, the luminosity and contrast between different colours would have an equal bearing on accessibility [23, 24]. For example, red and green can be indistinguishable to a protanope, but also completely lacking in contrast to a partially sighted person suffering from achromatopsia.

In terms of improvements to the simulation VE, a few key areas would enhance functionality, practicality and simulation fidelity.

Firstly, it would be useful to build a Graphical User Interface (GUI). This could include a 'pause menu', offering a more elegant way to close the application or tweak graphical settings. Equally, the GUI may include a readout indicating which colour mode the simulation is running in (normal, deuteranopia, protanopia, tritanopia or achromatopsia); this functionality is available in the Unity editor via the live debug console (**see Figure 12**), but is significantly more complicated to build into the Unity player (text must be rendered to match the camera position).

Secondly, it would be more useful to designers if external models could be selected within the Unity player to test, without the need to modify the scene within the Unity editor. This would almost certainly require a vastly more complex solution to implement, perhaps beyond the capabilities of the Unity platform. At the same time, this would provide a useful standalone tool, particularly where design practices have limited IT resources or for non-expert users, such as the end client/project stakeholders.

Thirdly, some improvements to the VE interactivity would improve its fidelity. For example, the vertical camera movement bug could be fixed to allow more natural exploration of the VE. Similarly, it may be useful to cycle camera modes forwards *or* backwards, or to link each mode to a specific key input (e.g. 1, 2, 3, 4) to compare between specific types of colour blindness. This was not possible within the constraints of the camera changing script, which used an array to switch cameras based on an incremental key press; alterations to this logic require other 'main' cameras to be disabled when a live camera is selected.

Unfortunately, these items proved difficult to implement, and will likely require sophisticated coding skills to improve the basic demonstration project.

CONCLUSION

Ultimately, the key project aims were achieved, noting the limitations and future opportunities covered previously. Critically, an accurate simulation for different types of colour blindness using VE was possible within the demonstration project. Moreover, there is clear practical application for designers, as the real-time graphical representation of colour blindness can be used to quickly and easily test any 3D model for appropriate colour design (given some basic familiarity with the Unity editor).

The ability to preview models in this way represents a significant improvement on traditional methods, such as the information tables used by graphic designers or the online image reprocessing tools. There is an overwhelming natural synergy between visualisation technologies, such as the Unity VE, and the simulation of visual impairment. In contrast, more traditional methods may lack a visual component altogether or may provide a visual aid – but within an impractically slow workflow.

Even without full use of specialist VR equipment to simulate colour blindness, the basic first person model presented in the final project demonstration shows potential to benefit designers in considering accessibility for a wide range of building users. This is a good example of how the construction industry can adopt technologies from other industries, such as IT/video game design, in order to innovate and improve existing methods.

APPENDIX A (FIGURES)

Figure 1 (Project v1.0, VREX procedurally generated rooms)

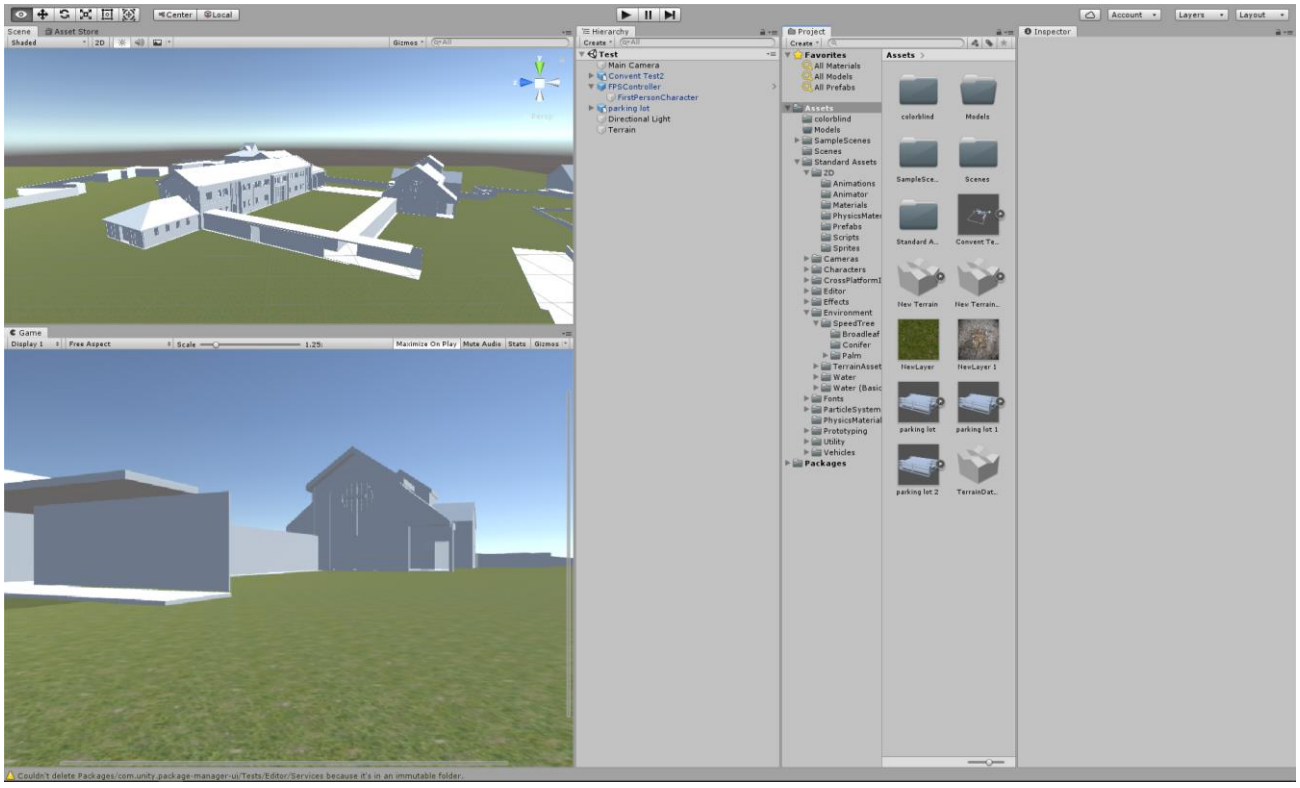

Figure 2 (Project v1.1, imported Revit model)

Figure 3 (Original Revit Model)

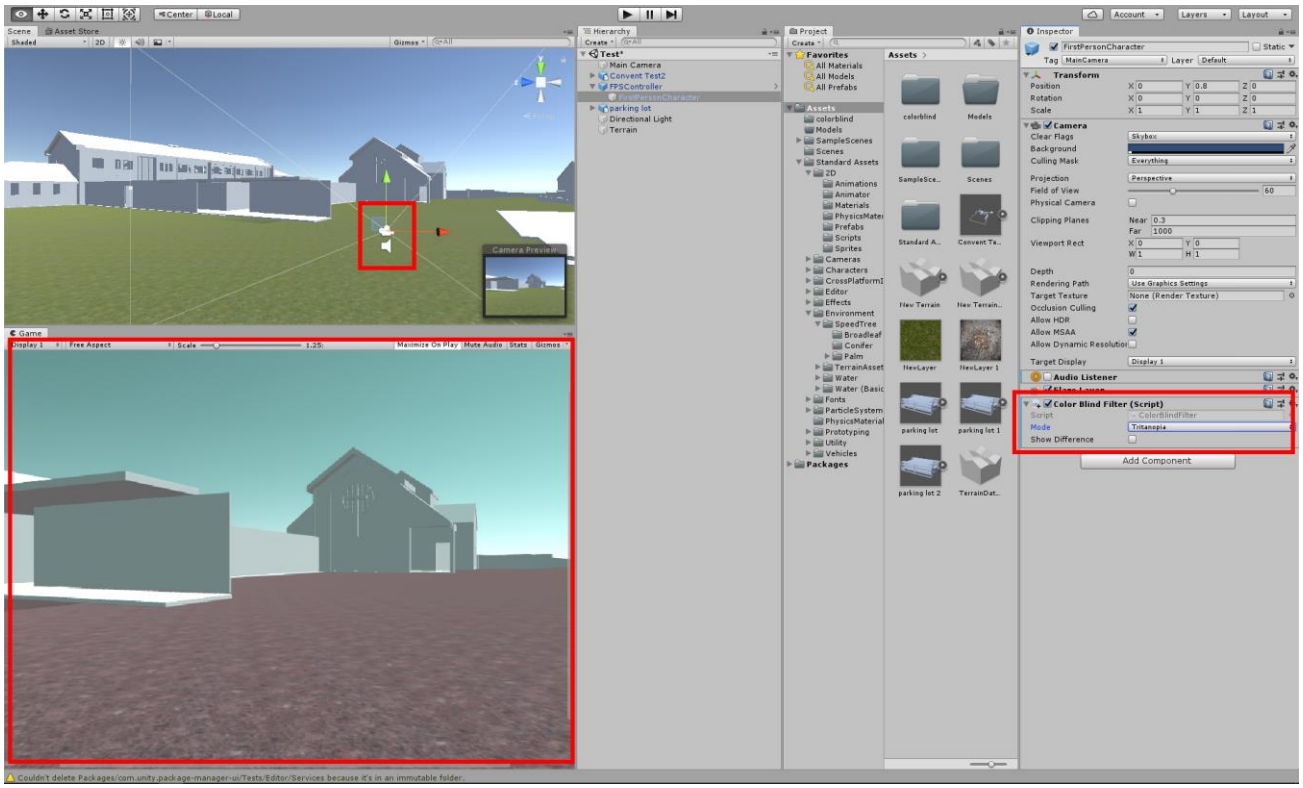

Figure 4 (Project v1.1, working prototype of colour blindness filter)

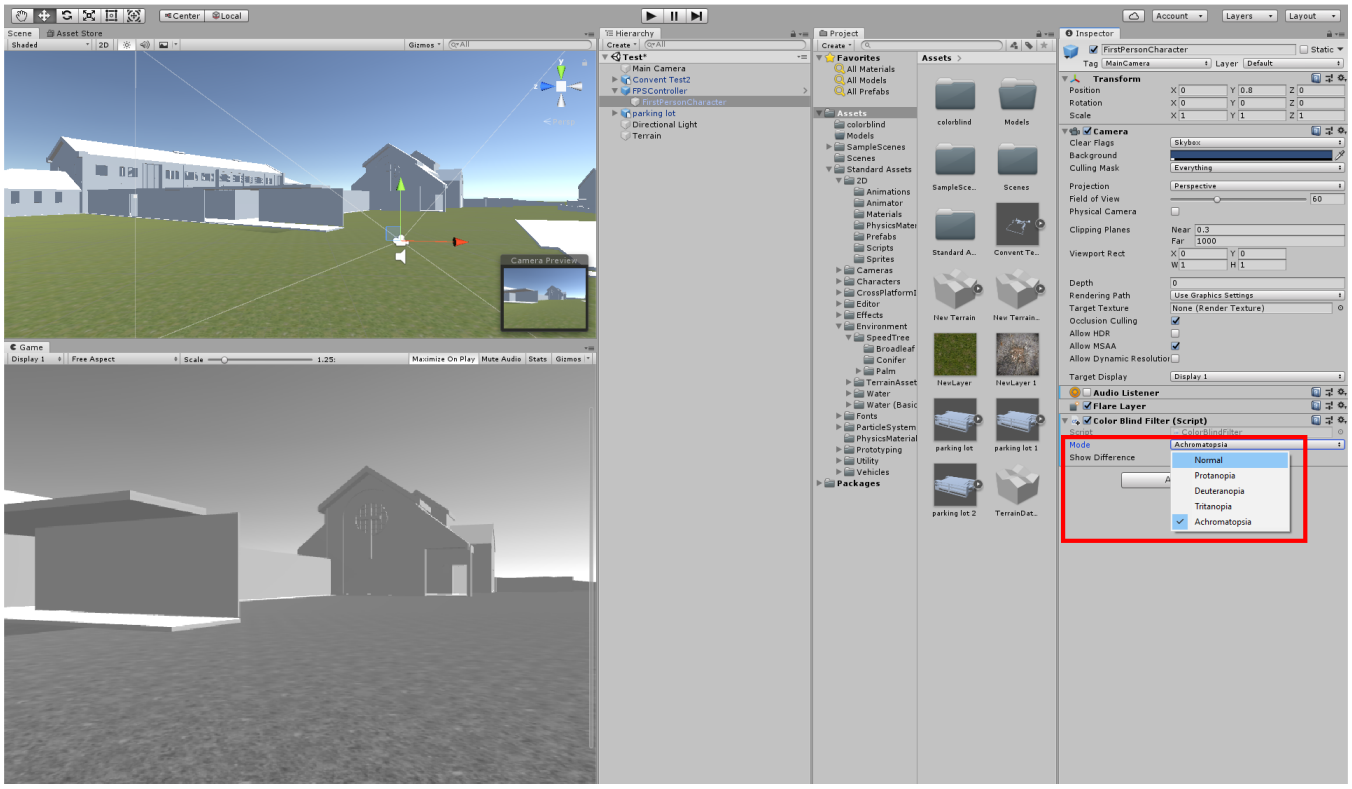

Figure 5 (Project v1.1, colour blindness filter menu)

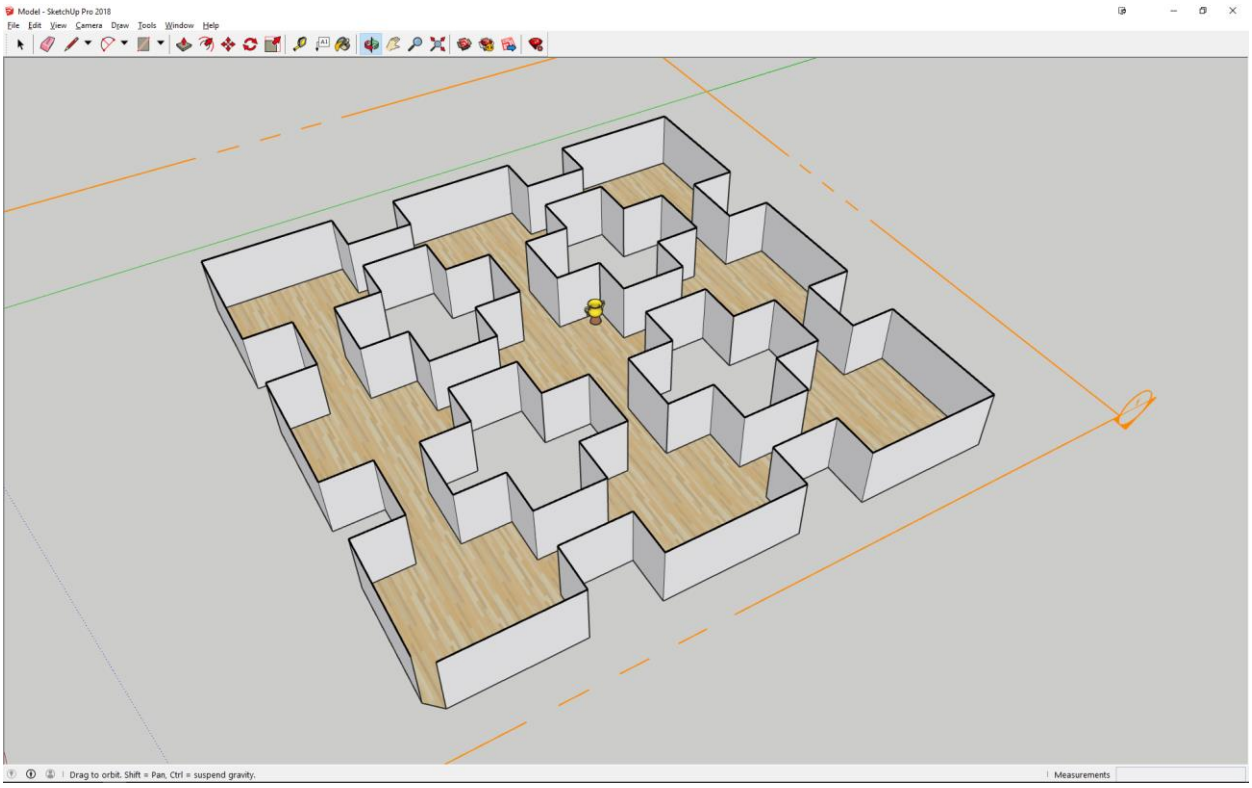

Figure 6 (Sketchup Model)

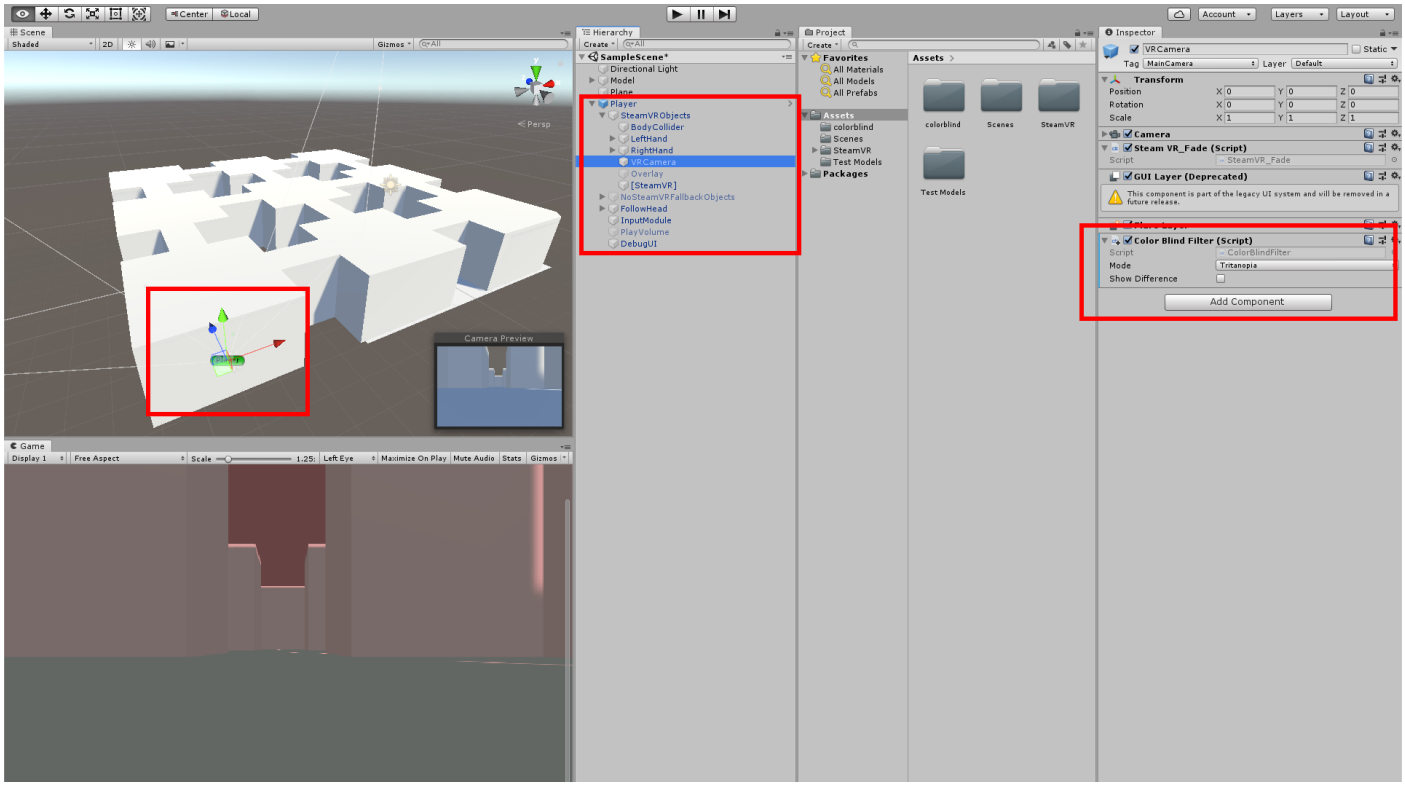

Figure 7 (Project v1.2, SteamVR camera with filter)

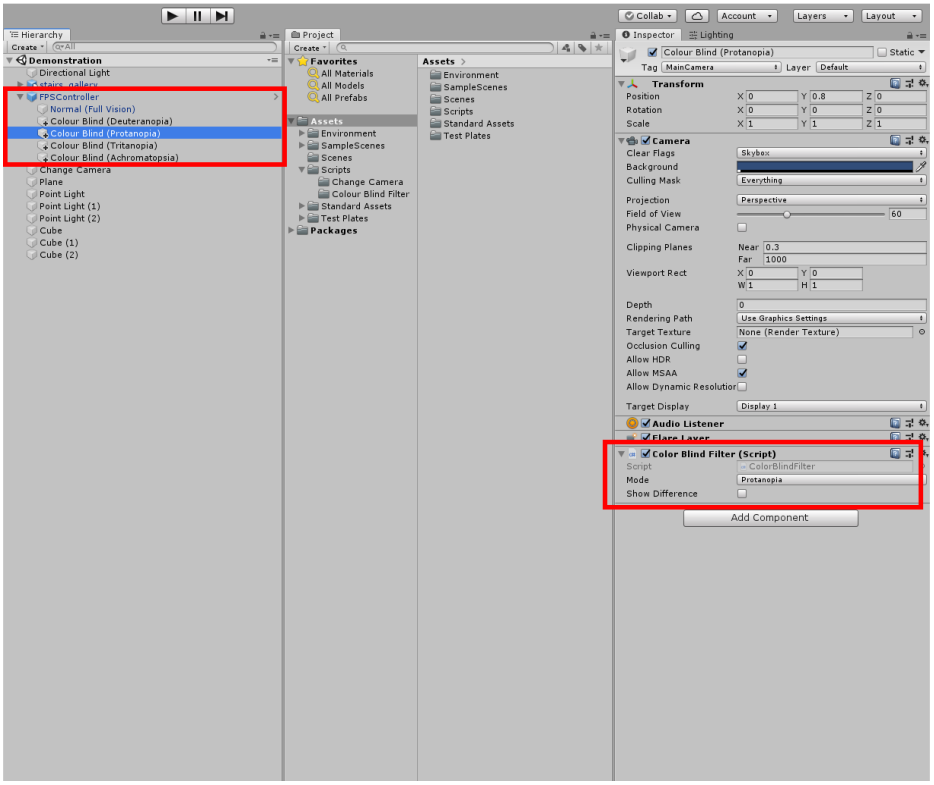

Figure 8 (Project v1.3, camera array with filters)

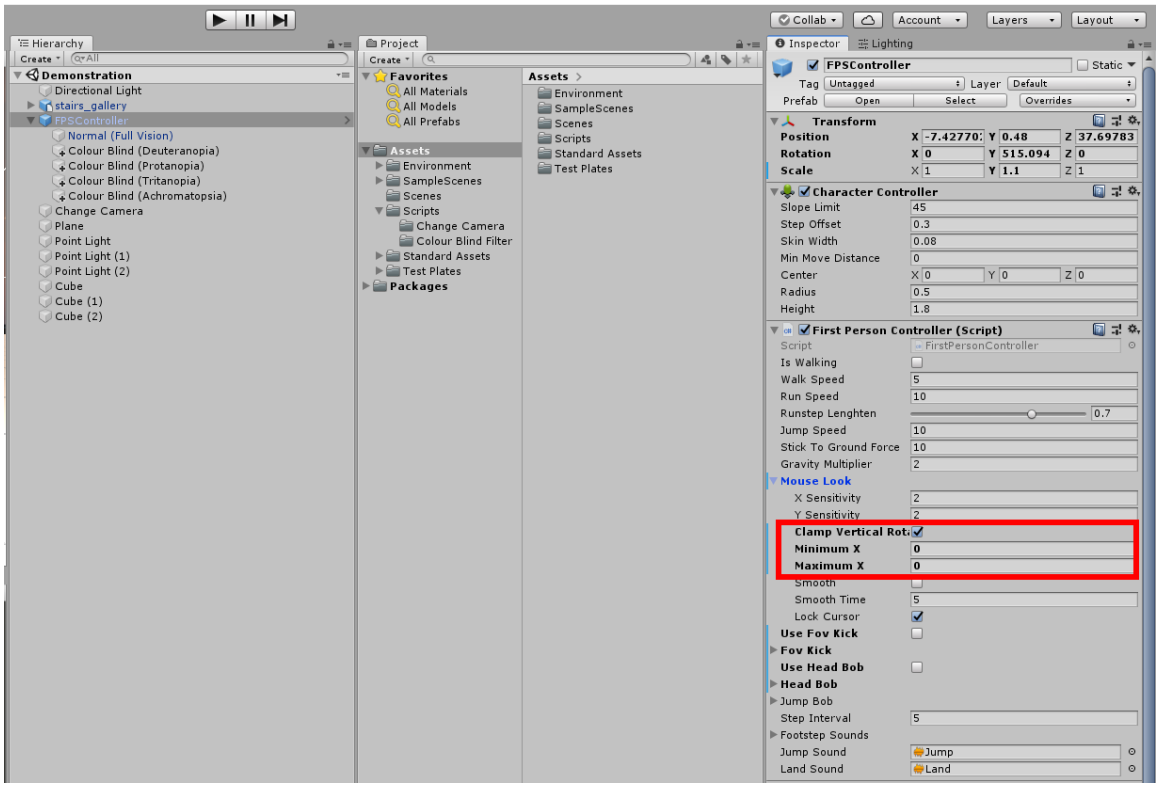

Figure 9 (Project v1.3, restricting camera movement)

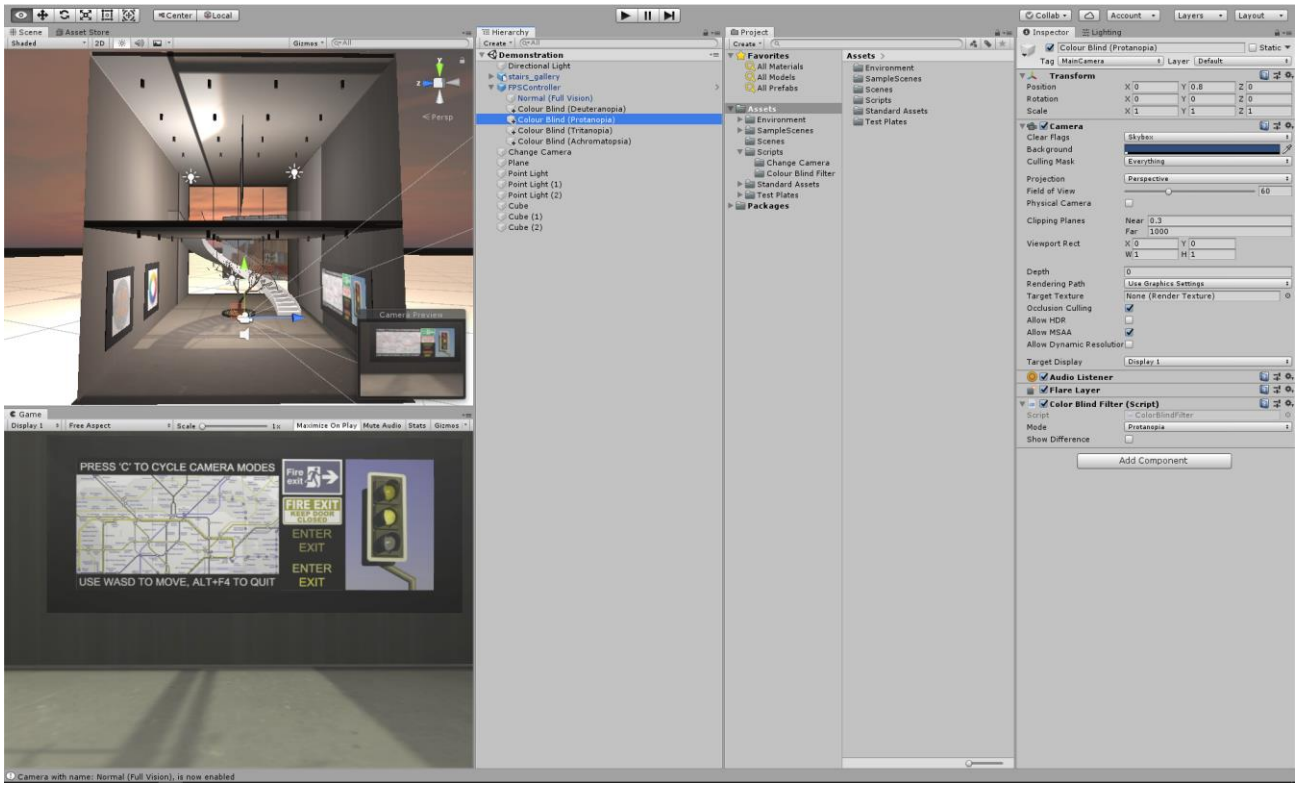

Figure 10 (Project v1.3, finished build setup)

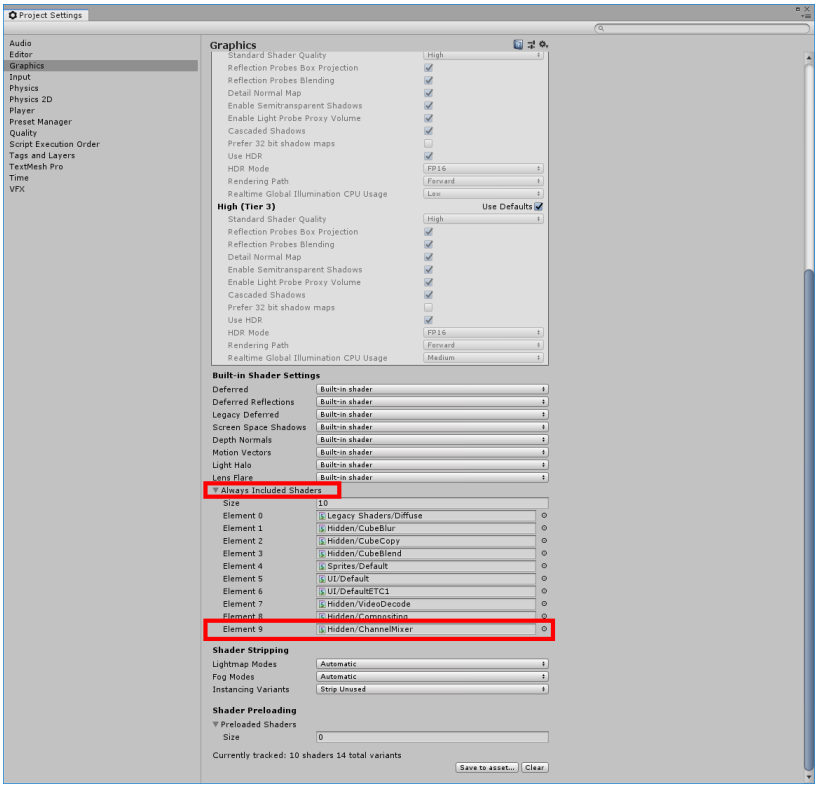

Figure 11 (Project v1.3, Build Settings error)

Figure 12 (Debug Console)

APPENDIX B (SCRIPTS)

Colour Blindness Filter, script written by Alan Zuconni [18].

```
// Alan Zucconi
// www.alanzucconi.com
using UnityEngine;

public enum ColorBlindMode
{
    Normal = 0,
    Protanopia = 1,
    Protanomaly = 2,
    Deuteranopia = 3,
    Deuteranomaly = 4,
    Tritanopia = 5,
    Tritanomaly = 6,
    Achromatopsia = 7,
    Achromatomaly = 8,
}

[ExecuteInEditMode]
public class ColorBlindFilter : MonoBehaviour
{
    public ColorBlindMode mode = ColorBlindMode.Normal;
    private ColorBlindMode previousMode = ColorBlindMode.Normal;

    public bool showDifference = false;

    private Material material;

    private static Color[,] RGB =
    {
        { new Color(1f,0f,0f),   new Color(0f,1f,0f), new Color(0f,0f,1f) },    // Normal
        { new Color(.56667f, .43333f, 0f), new Color(.55833f, .44167f, 0f), new Color(0f,
.24167f, .75833f) },    // Protanopia
        { new Color(.81667f, .18333f, 0f), new Color(.33333f, .66667f, 0f), new Color(0f, .125f,
.875f)    }, // Protanomaly
        { new Color(.625f, .375f, 0f), new Color(.70f, .30f, 0f), new Color(0f, .30f, .70f)    },
// Deuteranopia
        { new Color(.80f, .20f, 0f), new Color(.25833f, .74167f, 0), new Color(0f, .14167f,
.85833f)    },    // Deuteranomaly
        { new Color(.95f, .05f, 0), new Color(0f, .43333f, .56667f), new Color(0f, .475f, .525f)
}, // Tritanopia
        { new Color(.96667f, .03333f, 0), new Color(0f, .73333f, .26667f), new Color(0f, .18333f,
.81667f) }, // Tritanomaly
        { new Color(.299f, .587f, .114f), new Color(.299f, .587f, .114f), new Color(.299f, .587f,
.114f)  },   // Achromatopsia
        { new Color(.618f, .32f, .062f), new Color(.163f, .775f, .062f), new Color(.163f, .320f,
.516f)  }    // Achromatomaly
    };

    void Awake()
    {
        material = new Material(Shader.Find("Hidden/ChannelMixer"));
        material.SetColor("_R", RGB[0, 0]);
        material.SetColor("_G", RGB[0, 1]);
        material.SetColor("_B", RGB[0, 2]);
    }

    void OnRenderImage(RenderTexture source, RenderTexture destination)
```

```
    {
        // No effect
        if (mode == ColorBlindMode.Normal)
        {
            Graphics.Blit(source, destination);
            return;
        }

        // Change effect
        if (mode != previousMode)
        {
            material.SetColor("_R", RGB[(int)mode, 0]);
            material.SetColor("_G", RGB[(int)mode, 1]);
            material.SetColor("_B", RGB[(int)mode, 2]);
            previousMode = mode;
        }

        // Apply effect
        Graphics.Blit(source, destination, material, showDifference ? 1 : 0);
    }
}
```

Camera Controller, script written by 'TheArtist' [25].

```
using UnityEngine;
using System.Collections;

public class CameraController : MonoBehaviour {
     public Camera[] cameras;
     private int currentCameraIndex;

     // Use this for initialization
     void Start () {
         currentCameraIndex = 0;

         //Turn all cameras off, except the first default one
         for (int i=1; i<cameras.Length; i++)
         {
             cameras[i].gameObject.SetActive(false);
         }

         //If any cameras were added to the controller, enable the first one
         if (cameras.Length>0)
         {
             cameras [0].gameObject.SetActive (true);
             Debug.Log ("Camera with name: " + cameras [0].GetComponent<Camera>().name + ", is
now enabled");
         }
     }

     // Update is called once per frame
     void Update () {
         //If the c button is pressed, switch to the next camera
         //Set the camera at the current index to inactive, and set the next one in the array to
active
         //When we reach the end of the camera array, move back to the beginning or the array.
         if (Input.GetKeyDown(KeyCode.C))
         {
             currentCameraIndex ++;
             Debug.Log ("C button has been pressed. Switching to the next camera");
             if (currentCameraIndex < cameras.Length)
             {
                 cameras[currentCameraIndex-1].gameObject.SetActive(false);
                 cameras[currentCameraIndex].gameObject.SetActive(true);
                 Debug.Log ("Camera with name: " + cameras
[currentCameraIndex].GetComponent<Camera>().name + ", is now enabled");
             }
             else
             {
                 cameras[currentCameraIndex-1].gameObject.SetActive(false);
                 currentCameraIndex = 0;
                 cameras[currentCameraIndex].gameObject.SetActive(true);
                 Debug.Log ("Camera with name: " + cameras
[currentCameraIndex].GetComponent<Camera>().name + ", is now enabled");
             }
         }
     }
 }
```

REFERENCES

1. Colour Blind Awareness: Living with Colour Vision Deficiency. Available at http://www.colourblindawareness.org/colour-blindness/living-with-colour-vision-deficiency/
2. CPD 6 2018. Colour-considered Design for the Visually Impaired. Available at https://www.bdonline.co.uk/cpd/cpd-6-2018-colour-considered-design-for-the-visually-impaired/5093226.article
3. National Eye Institute: Facts About Color Blindness. Available at https://nei.nih.gov/health/color_blindness/facts_about
4. Royal National Institute of Blind People: Colour Vision Deficiency. Available at https://www.rnib.org.uk/nb-online/colour-vision-deficiency
5. Koch, E. (2015), Color and Contrast in the Built Environment. Available at https://www.bdcnetwork.com/color-and-contrast-built-environment
6. Fickett, M. (2015), Color Universal Design and Architecture. Available at https://www.payette.com/cool-stuff/color-universal-design-and-architecture/
7. Tang, C-H., Wu, W-T. and Lin, C-Y. (2009), Using virtual reality to determine how emergency signs facilitate way-finding. *Applied Ergonomics 40*, 722-730.
8. Krösl, K., Bauer, D., Schwärzler, M., Fuchs, H., Suter, G. and Wimmer, M. (2018), A VR-based user study on the effects of vision impairments on recognition distances of escape-route signs. *The Visual Computer 34*, 911-923.
9. Coloring for Colorblindness. Available at https://davidmathlogic.com/colorblind/#%23D81B60-%231E88E5-%23FF0000-%23004D40
10. Ahmer, C. (2014), Making Architecture Visible to the Visually Impaired.
11. Meyer, G. W. and Greenberg, D. P. (1988), Color-Defective Vision.
12. Wickline Color Laboratory. Available at http://web.archive.org/web/20081216005758/http://colorlab.wickline.org/colorblind/colorlab/
13. Color Blindness Simulation: Color Matrix. Available at https://web.archive.org/web/20081014161121/http://www.colorjack.com/labs/colormatrix/
14. Coblis: Color Blindness Simulator. Available at https://www.color-blindness.com/coblis-color-blindness-simulator/
15. Color-Blindness Simulators. Available at https://lpetrich.org/Science/ColorBlindnessSim/ColorBlindnessSim.html
16. WebAIM: Color Contrast Checker. Available at https://webaim.org/resources/contrastchecker/
17. Colormax: Color Blind Test. Available at https://colormax.org/color-blind-test/
18. Alan Zucconi: Accessibility Design, Color Blindness. Available at https://www.alanzucconi.com/2015/12/16/color-blindness/
19. Vasser, M., Kängsepp, M., Magomedkerimov, M., Kilvits, K., Stafinjak, V., Kivisik, T., Vicente, R., & Aru, J. (2017), VREX: an open-source toolbox for creating 3D virtual reality experiments. *BMC Psychology 5, 4*.
20. Lubecki, M. (2018), VR Staircase Art Gallery 2018: 3D model. Available at https://skfb.ly/6GUMS
21. Hamilton, I. (2018), A practitioner reflection on accessibility in virtual reality environments. *The Computer Games Journal 7,2*, 63-74.
22. Stewart, G. W. and McCrindle, R. J. (2017), Visual impairment simulator for auditing and design. *Journal of Alternative Medicine Research 9, 4*, 419-430.
23. Bright, K. and Egger, V. (2008), Using visual contrast for effective, inclusive, environments. *Information Design Journal 16,3*, 178-189.
24. Bright, K. (2014), Making the built environment and public transport more user-friendly for visually-impaired people. University *of Reading. Impact Case Study (REF3b)*.
25. Camera Controller script. Available at https://answers.unity.com/questions/16146/changing-between-cameras.html