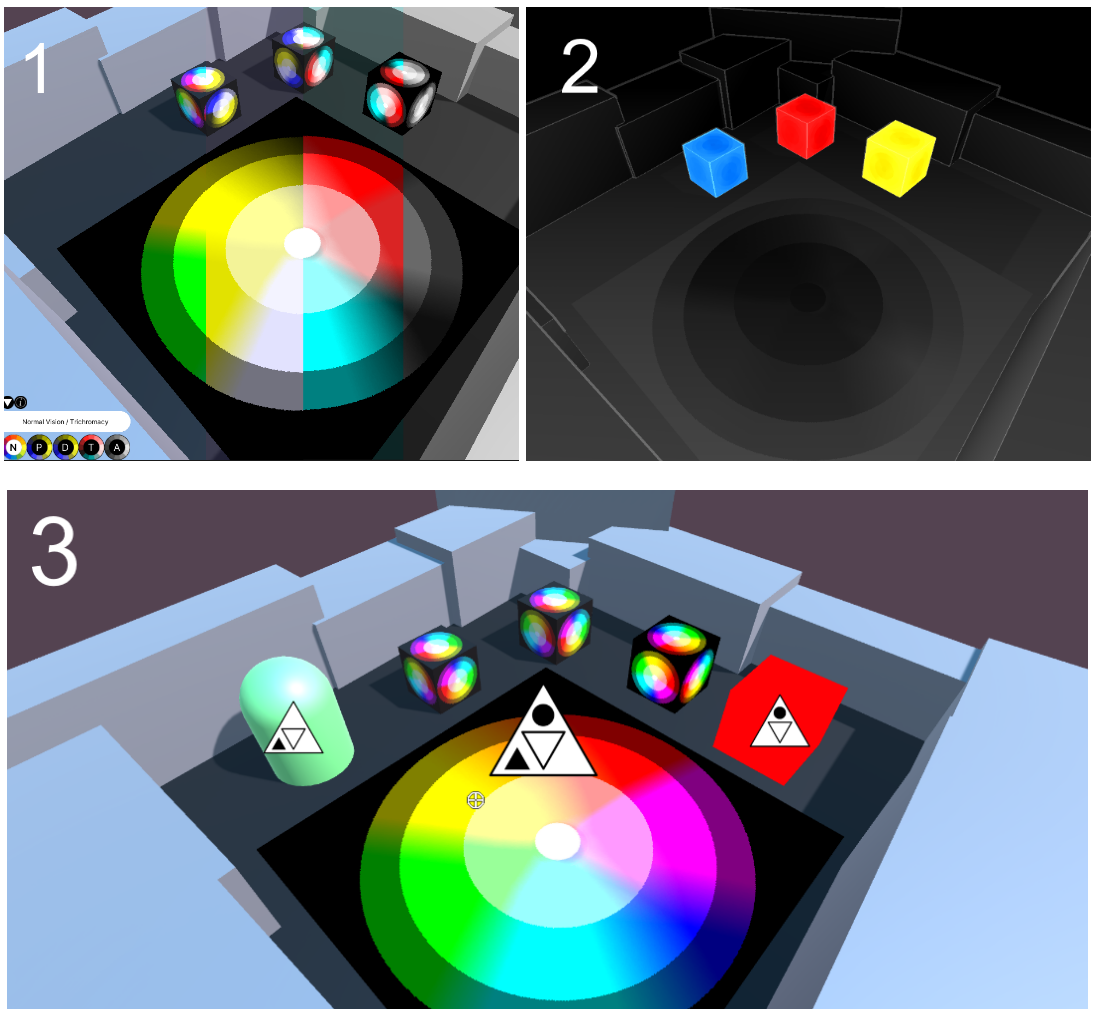
Easy Color Blind Tool Documentation

Created on Unity 2021.1

Created by Joel Torrent @_jowent

Overview

This Tool offers 3 different solutions to solve color blindness issues that could show in the build version of the game. These solutions are divided into two categories: design and final game functionalities.

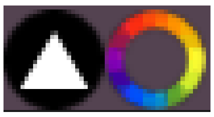


Solution 1 - Editor Filters

This is not a filter to apply to the final game, because IT'S NOT A SOLUTION, IT'S A SIMULATION.

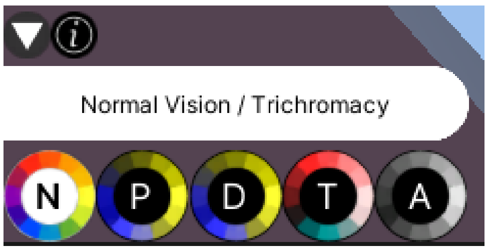
This part of the tool applies filters to the scene view & play mode in the editor. These filters simulate how the different types of colorblindness reduce the color spectrum so the developers can detect differences between normal vision and colorblind vision to adjust the contrast of the elements of the scene.

When the tool is installed an icon appears at the left bottom of the scene view. To enable the menu just press the ▲ Button.

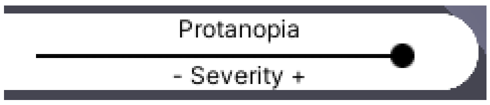


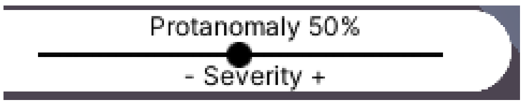
Once the menu is unfolded you can see the different types of Color Vision:

- **N**: Normal Vision, also called Trichromacy
- **P**: Protanopia: Loss of red cones
- **D**: Deuteranopia: Loss of green cones
- **T**: Tritanopia: Loss of blue cones
- **A**: Achromatopsia: Loss of 2 or more cones. Use of rods.

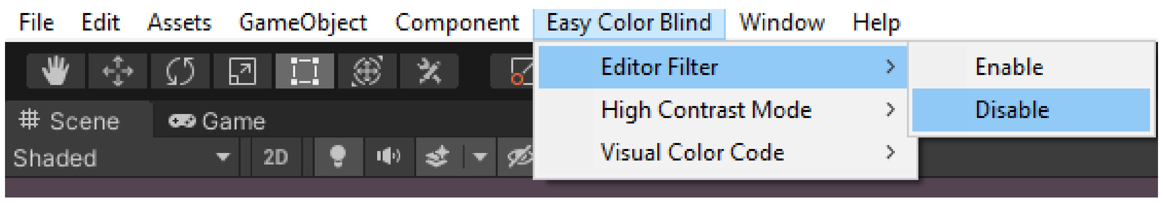


There's a slider for Protanopia, Deuteranopia and Tritanopia, so the loss of the respective cones can be partial. <100% means that is a partial anomaly.



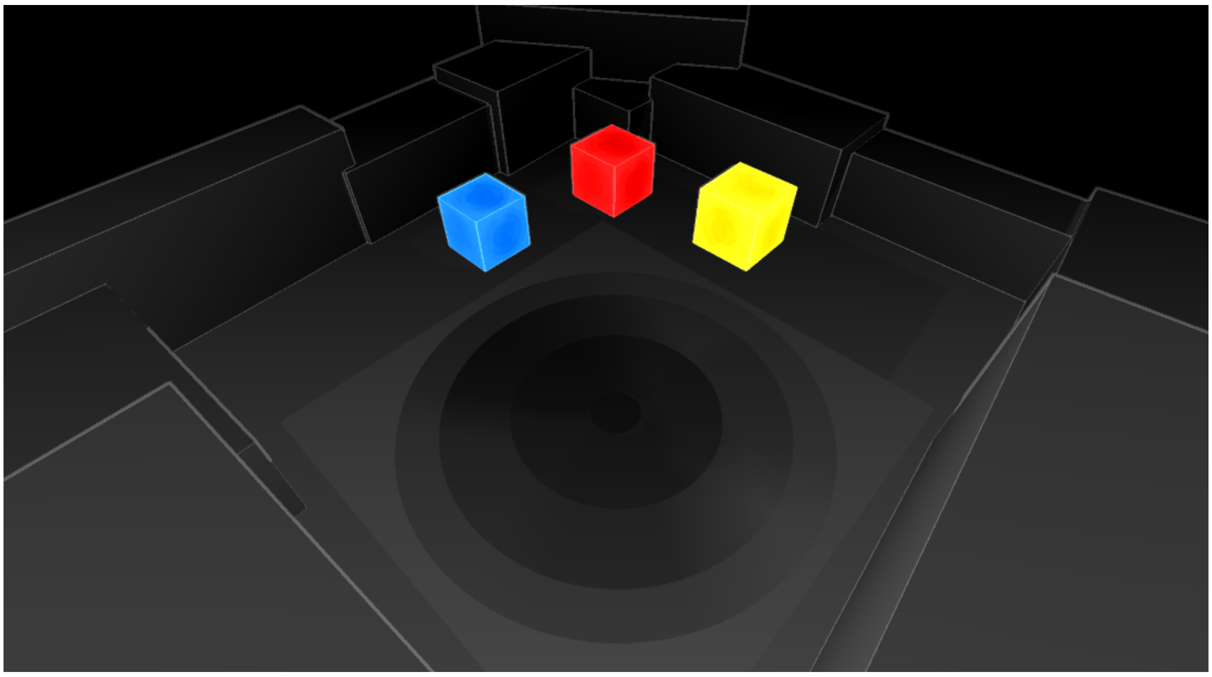


If you want to hide this functionality from the Scene View, click on Easy Color Blind > Editor Filter > Disable.



Solution 2 - High Contrast Mode

This is a final look to be implemented in the final Build of the game. Inspired by The Last Of Us Part II, the final look consists of a grayscale background to show the depth of the scene, and outline and a recolor of the important gameplay elements to achieve the maximum contrast of a scene. The implementation will be done with shaders and a component that controls the behavior.



2.1 Setting Up the shaders

This look is achieved to special shaders that interpolate from the usual look to the high contrast version.

The tool provides 3 shaders that use these functionalities. When you create a material, select one of these shaders:

- **Surface**: Easy Color Blind/High Contrast/Surface High Contrast
- **Unlit**: Easy Color Blind/High Contrast/Unlit High Contrast
- **Skybox**: Easy Color Blind/High Contrast/Skybox High Contrast

If you want to create your own shaders that use these functionalities there's a few steps you need to follow

A - If you are creating a normal Vertex-Fragment pipeline Shader you need to:

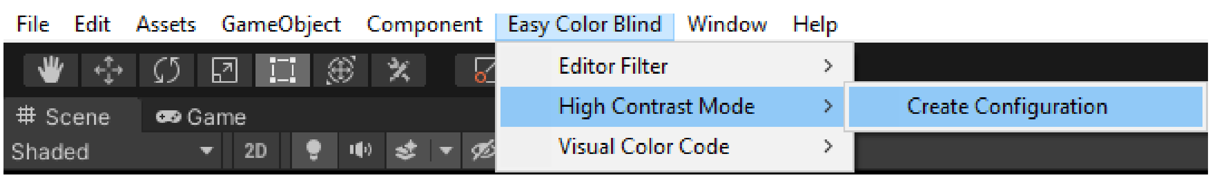
1. Add **_HighContrastColor("High Contrast Color", Color) = (0,0,0,0)** inside the properties.
2. Write **#include "CGIncludes/HighContrastFunctions.cginc"** under the **#pragma vertex** and fragment lines
3. Declare a **float4 screenPosition** inside the **v2f struct**;
4. In the **v2f vert** function add **o.screenPosition = ComputeScreenPos(o.vertex);** just before the **return o**;
5. In the **fixed4 frag** function, add **ApplyHighContrastFrag(color, i.screenPosition);** before the final return of the color;

B - If you create a Unity surface shader:

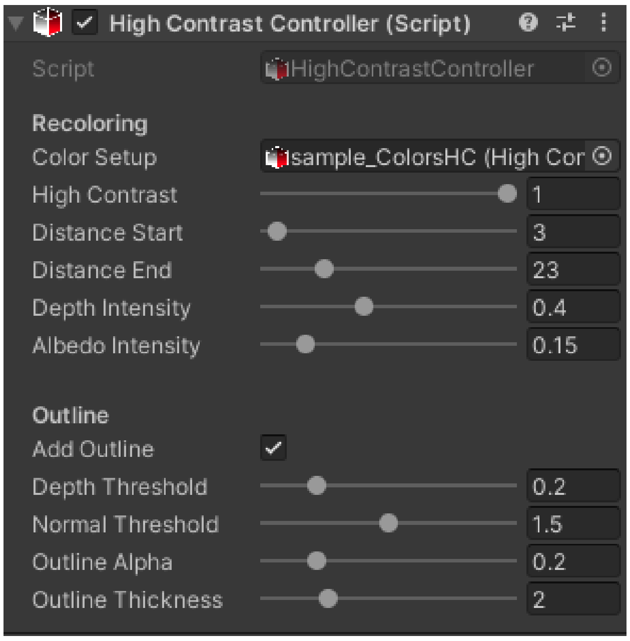
1. Add **_HighContrastColor("High Contrast Color", Color) = (0,0,0,0)** inside the properties.
2. Change the **#pragma surface surf** line into **#pragma surface surf Standard finalcolor:SurfaceFinalColor**
3. Inside the **Input struct**, declare a **float4 screenPos**;
4. After declaring the Input struct, add 2 lines: **#define shader_surf** and **#include "CGIncludes/HighContrastFunctions.cginc"**

2.2 Configuration

Once you have your materials setup with the special shaders you can add the controller of this effect by a click on the menu: Easy Color Blind > High Contrast Mode > Create Configuration



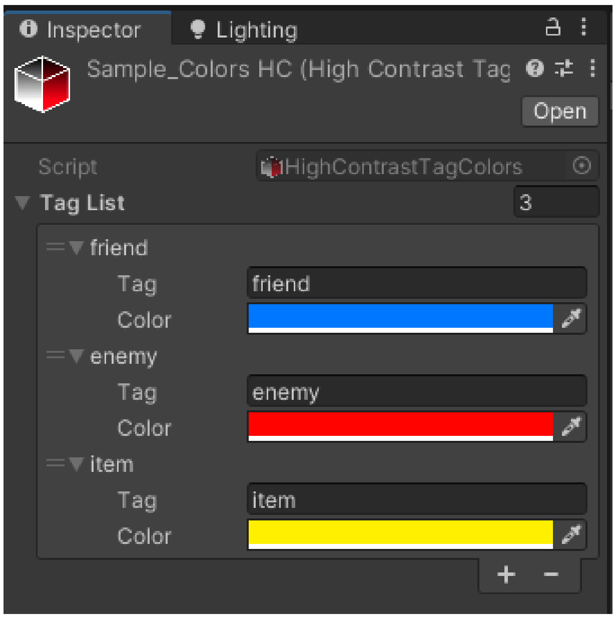
With this function a new gameobject will be added to the scene with the needed component.



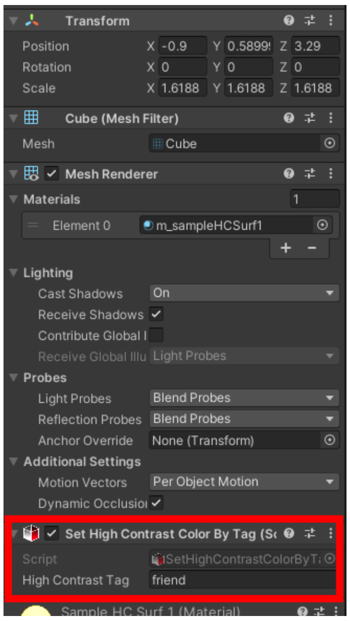
```
High Contrast Controller (Script)
Script              HighContrastController
Recoloring
Color Setup         sample_ColorsHC (High Cor
High Contrast       1
Distance Start      3
Distance End        23
Depth Intensity     0.4
Albedo Intensity    0.15
Outline
Add Outline         ✓
Depth Threshold     0.2
Normal Threshold    1.5
Outline Alpha       0.2
Outline Thickness   2
```

There are few parameters to adjust:

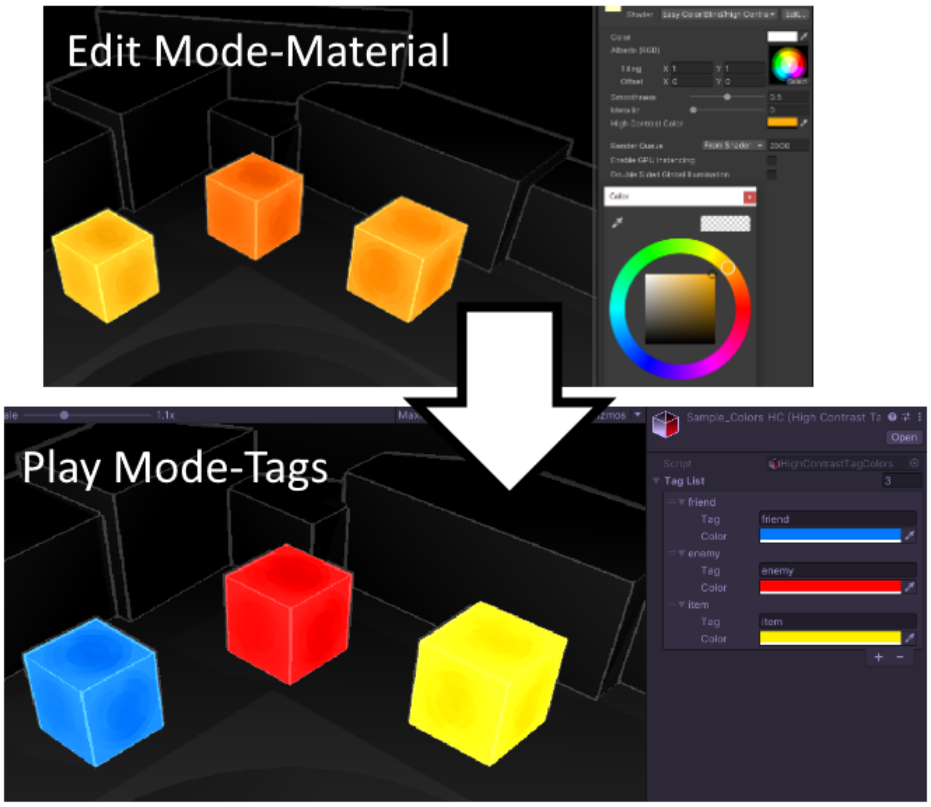
- High Contrast
  - Activate the mode. Can be interpolated From 0 to 1 to make a fade.
- Color Setup
  - If NULL the final color of the object will be determined by the _HighContrastColor field of the material.
  - If not null, using a Scriptable Object found in **Create>Easy Color Blind>HighContrastTagColors** will determine the final color if the object has the component **Set High Contrast Color by tag added**. The final colors will be applied in play mode.



```
Inspector   Lighting
Sample_Colors HC (High Contrast Tag
Open
Script      HighContrastTagColors
Tag List    3
  friend
    Tag     friend
    Color   🟦
  enemy
    Tag     enemy
    Color   🟥
  item
    Tag     item
    Color   🟨
```



```
Transform
Position    X -0.9     Y 0.5899!  Z 3.29
Rotation    X 0        Y 0        Z 0
Scale       X 1.6188   Y 1.6188   Z 1.6188
Cube (Mesh Filter)
Mesh        Cube
Mesh Renderer
Materials   1
  Element 0   m_sampleHCSurf1
Lighting
  Cast Shadows        On
  Receive Shadows     ✓
  Contribute Global I
  Receive Global Illu Light Probes
Probes
  Light Probes        Blend Probes
  Reflection Probes   Blend Probes
  Anchor Override     None (Transform)
Additional Settings
  Motion Vectors      Per Object Motion
  Dynamic Occlusioı   ✓
Set High Contrast Color By Tag (Sc
Script              SetHighContrastColorByTa
High Contrast Tag   friend
Sample HC Surf 1 (Material)
```



Script HighContrastTagColors
Tag List 3
friend
Tag friend
Color 🟦
enemy
Tag enemy
Color 🟥
item
Tag item
Color 🟨

- **Distance Start-End**
    - Linear distances, in unity units to remap the depth gradient.
- **Depth intensity**:
    - Opacity of the depth gradient.
- **Albedo intensity**
    - How much the Albedo will contrast.
- **Outline**
    - Enable/Disable.
- **Depth and Normal Threshold**:
    - Parameters to adjust the edge detection through depth and faces normals of the elements of the scene.
- **Outline Alpha and Thickness**:
    - Final edge look.

Solution 3 - Visual Color Code

This solution doesn't change the appearance of the game. Instead the tool draws a figure over the meshes or the HUD to represent a color.

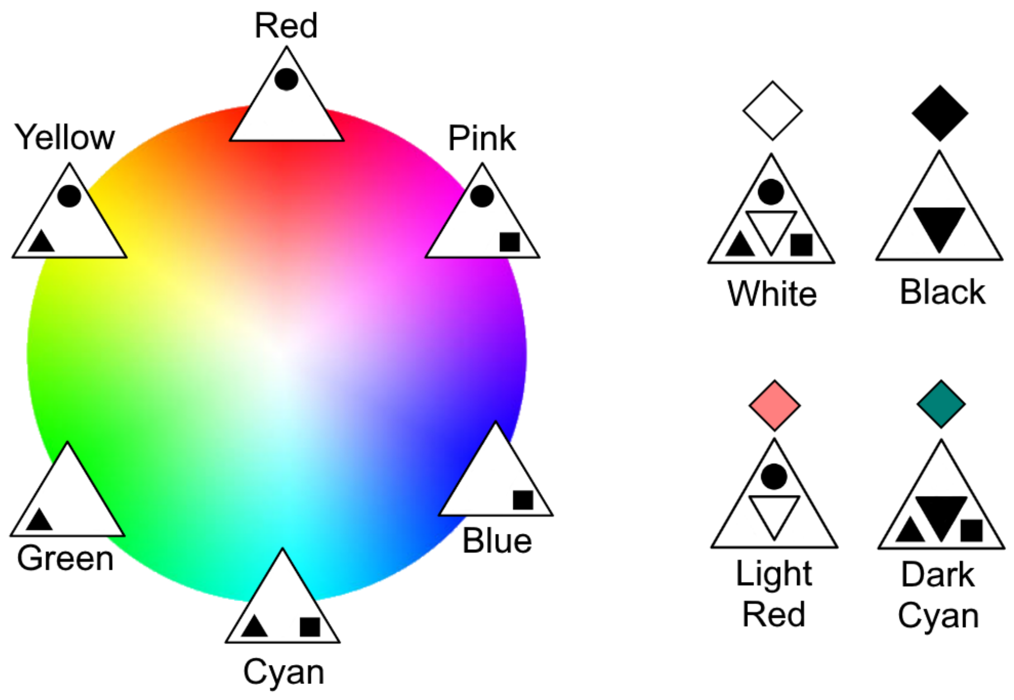
The Color code is inspired by an existing one, named ColorAdd. This is a new visual color code to achieve more consistency in terms of color space (using RGB instead of using Red, Yellow and Blue as Color Add).

3.1 Visual Color Code Explanation

The Visual color consists of a triangular base to act as a background and 4 elements:

- A Circle (●) on the top of the triangle to represent the **RED** Component.
- A Triangle (▲) on the bottom left to represent the **GREEN** Component.
- A Square (■) on the bottom right to represent the **BLUE** Component.
- An inverted triangle (▼) to represent if the color is **Dark** or **Light**.

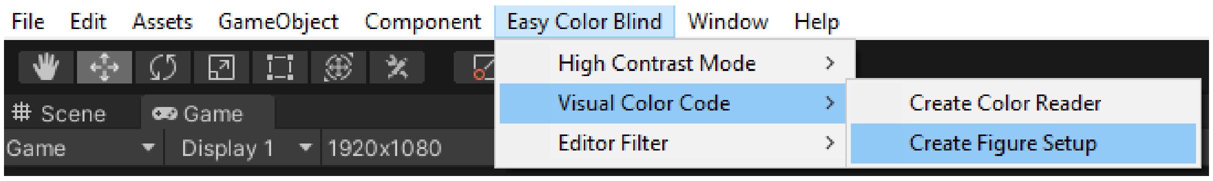
By creating combinations of symbols there is a spectrum of 24 colors.



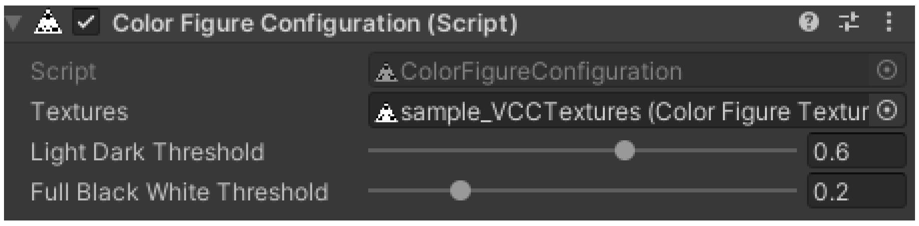
3.2 Visual Color Code Setup

To apply the visual color code textures and configuration it is necessary to create an object with a Color Figure Configuration component and assign a set of textures by a ScriptableObject called Color Figure Textures.

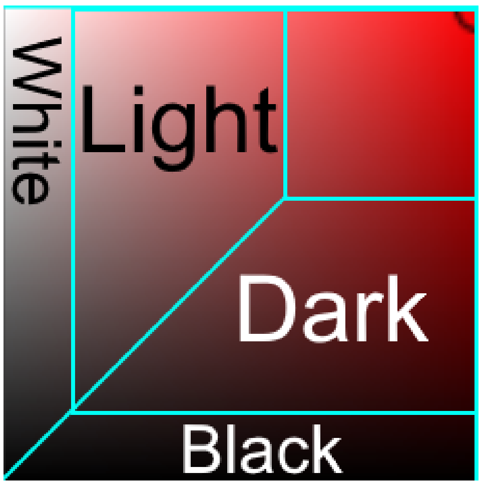
You can use the default set of textures and automatic creation of the object by clicking on the menu EasyColorBlind>Visual Color Code>Create Figure Setup



The component of the new object will have 2 slider parameters to tweak the thresholds that change the color to light/dark and to change the color to full white/black.

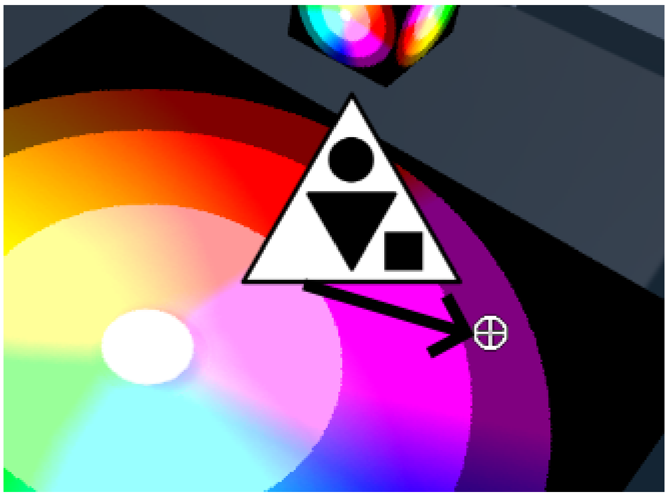

Color Figure Configuration (Script)

Script: ColorFigureConfiguration
Textures: sample_VCCTextures (Color Figure Textur
Light Dark Threshold: 0.6
Full Black White Threshold: 0.2

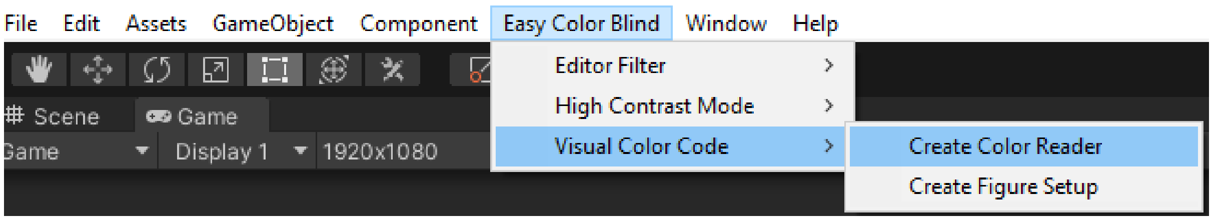


3.3 Implementing Color Read Crosshair

The first implementation of the Visual Color Code is a HUD crosshair that translates into its respective figure at one side.



To add this feature to the scene click on Menu>Easy Color Blind>Visual Color Code>Create Color Reader.



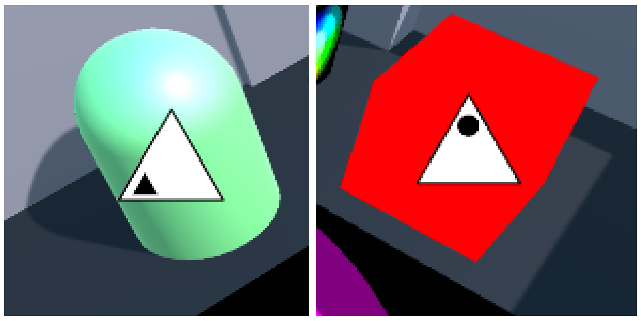
By default the crosshair (**VCC TARGET**) has a Crosshair.cs component that offers two types of movement: **Following the mouse** or **Fixed in the center of the screen**.

If you want to move the crosshair by your own code, remove the Crosshair.cs component of the **VCC TARGET** and move its transform by its own behaviour.

The parameters **Figure Offset** And **Figure Damp** will tweak the separation of the figure from the crosshair and the speed at which the figure follows it.

3.4 Implementing Figure On Shaders

Another option to use the Visual Color Code is to draw the figure inside the bounds of a mesh, above the colors of the pixels (lit/unlit) always projected to the camera. Like the High Contrast Mode, this requires special shaders.



The tool provides 2 shaders to create materials:

- **Unlit**: Easy Color Blind/Visual Color Code/Unlit Texture
- **Surface**: Easy Color Blind/Visual Color Code/Standard Surface

Like in High Contrast Mode shaders, if you want to create your own shaders that use these functionalities there's a few steps you need to follow:

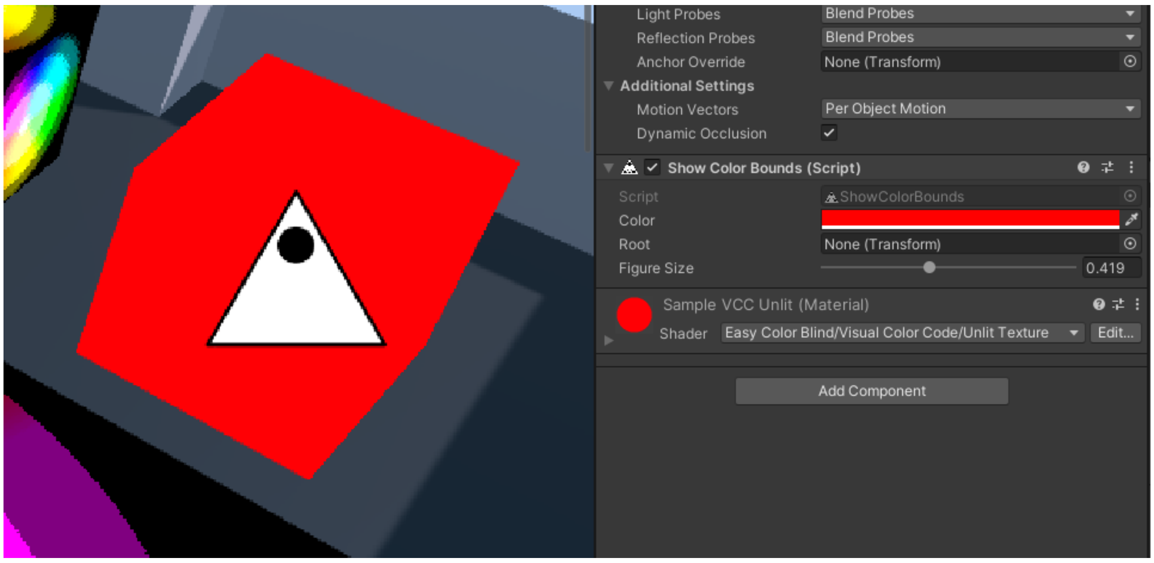
A - If you are creating a normal Vertex-Fragment pipeline Shader you need to:

1. Write **#include "CGIncludes/HighContrastFunctions.cginc"** under the #pragma vertex and fragment lines
2. Declare a **float4 screenPosition : TEXCOORD1;** inside the **v2f struct**;
3. In the **v2f vert** function add **o.screenPosition = ComputeScreenPos(o.vertex);** just before the **return o**;
4. In the **fixed4 frag** function, add **ApplyFigure(color, i.screenPosition);** before the final return of the color;

B - If you create a Unity surface shader:

1. Change the **#pragma surface surf** line into **#pragma surface surf Standard finalcolor:SurfaceFinalColor**
2. Inside the **Input struct**, declare a **float4 screenPos**;
3. After declaring the Input struct, add 2 lines: **#define shader_surf** and **#include "CGIncludes/FigureColorFunctions.cginc"**

With the material set on the renderer of the object, it is necessary to add a **Show Color Bounds** component to set manually the color, **offset the root** to calculate the bounds of the object and tweak the **size of the figure** inside the mesh.



Light Probes	Blend Probes
Reflection Probes	Blend Probes
Anchor Override	None (Transform)
Additional Settings	
Motion Vectors	Per Object Motion
Dynamic Occlusion	✓
Show Color Bounds (Script)	
Script	ShowColorBounds
Color	🟥
Root	None (Transform)
Figure Size	0.419
Sample VCC Unlit (Material)	
Shader	Easy Color Blind/Visual Color Code/Unlit Texture