Off Screen Target Indicator setup guide 2.0

Introduction:

Off screen target indicator package provides a simple solution to add a target indicator system to your game.

This package includes:

1. A demo scene in 'Pixel Play/_Scenes/_main'
2. Documentation in 'Pixel Play/Documentation/Setup Guide 2.0'
3. 'Arrow Indicator' and 'Box Indicator' Prefabs in 'Pixel Play/Prefabs/'
4. 'PixelPlay.OffScreenIndicator.dll' in 'Pixel Play/Scripts/Core/'
5. Off screen indicator scripts in 'Pixel Play/Scripts/OffScreenIndicator/'
6. A utility script, 'ExtendedFlyCam.cs' in 'Pixel Play/Scripts/'
7. 'WhiteArrow' and 'WhiteBox' sprites in 'Pixel Play/Sprites/'

Steps to setup up the targeting system:

1. Open a unity project and import the 'Off Screen Target Indicator' package.
2. Create a new tag named 'Target' in the 'Tags and Layers' inspector by selecting Edit -> Project Settings -> Tags and Layers.
3. Open the scene in which you want to add the 'Off screen target indicator'.
4. Add a new Canvas to the 'Hierarchy'. Make sure to set the canvas 'Render Mode' to 'Screen Space – Overlay'.
5. Right click on the above canvas and add a panel as a child to it. (You can rename the panel if you want to 'Off screen indicator panel').
6. Set the alpha of the above panel to zero. (You can do so by changing the 'alpha' value of the 'Color' property in the 'Image' script of the panel).
7. Add the 'OffScreenIndicator.cs' script form the 'Pixel Play/Scripts/OffScreenIndicator/' folder to the above panel. (You can do so by dragging and dropping the script from the scripts folder to the panel or clicking the 'Add Component' button and typing 'OffScreenIndicator' and selecting it).
8. In the same way add 'ArrowObjectPool.cs' and 'BoxObjectPool.cs' scripts to the panel. And assign the values of the 'Pooled Object' properties for both the scripts by dragging

and dropping the 'Arrow Indicator' and 'Box Indicator' prefabs respectively from the 'Pixel Play/Prefabs/' folder in those empty values.

9. Now drag and drop the 'Target.cs' script form the scripts folder to all the target game objects in the scene and adjust the various values of the script properties as you see fit.
10. For testing you can add the provided 'ExtendedFlyCam.cs' script to your 'Main camera' and hit 'Play'.

Modifying or creating your own arrow and box prefabs:

1. To modify the 'Arrow' or the 'Box Indicator' prefabs, select the prefabs in the 'Assets' folder.
2. As you will see these are simple 'UI image' components with a 'UI text' component added as a child to it.
3. You can change the size of the prefabs by adjusting the 'Width' and 'Height' properties of the 'Rect transform'
4. You can change the sprites for the arrow and box prefabs by dragging and dropping your own sprite in the 'source image' field for the 'Image' script component.
5. If you are creating your own prefabs, make sure to attach the 'Indicator.cs' script from the scripts folder to both the prefabs.

Video Tutorial:

The current video tutorial is for v1.0 and is deprecated, will upload a new video tutorial soon for v2.0

Change logs from v1.0 to v2.0:

- Added object pooling for the indicators.
- Can now change the indicator colors as per the target.
- Can now enable/disable the indicators for the target.
- Added 'Distance text' with the indicators to show the target distance from camera.
- Can enable/ disable the distance text for the target.
- Can adjust the value for screen bound offset in the off-screen indicator script, by default its 0.9.
- Most of the scripts are now available for you to toy with and modify according to your needs.

FAQs:

Q. Does the package work for HoloLens or any other AR, VR device?

A. As this package is designed to work with only 'Screen Space – Overlay' canvas render mode, it won't work with any of the devices AR/VR devices as in AR or VR 'Screen Space – Overlay' render mode does not work.

Q. How to change the unit for the distance in the distance text?

A. As of now most scripts are available so you can adjust the same in the 'Indicator.cs' -> 'SetDistanceText' function.