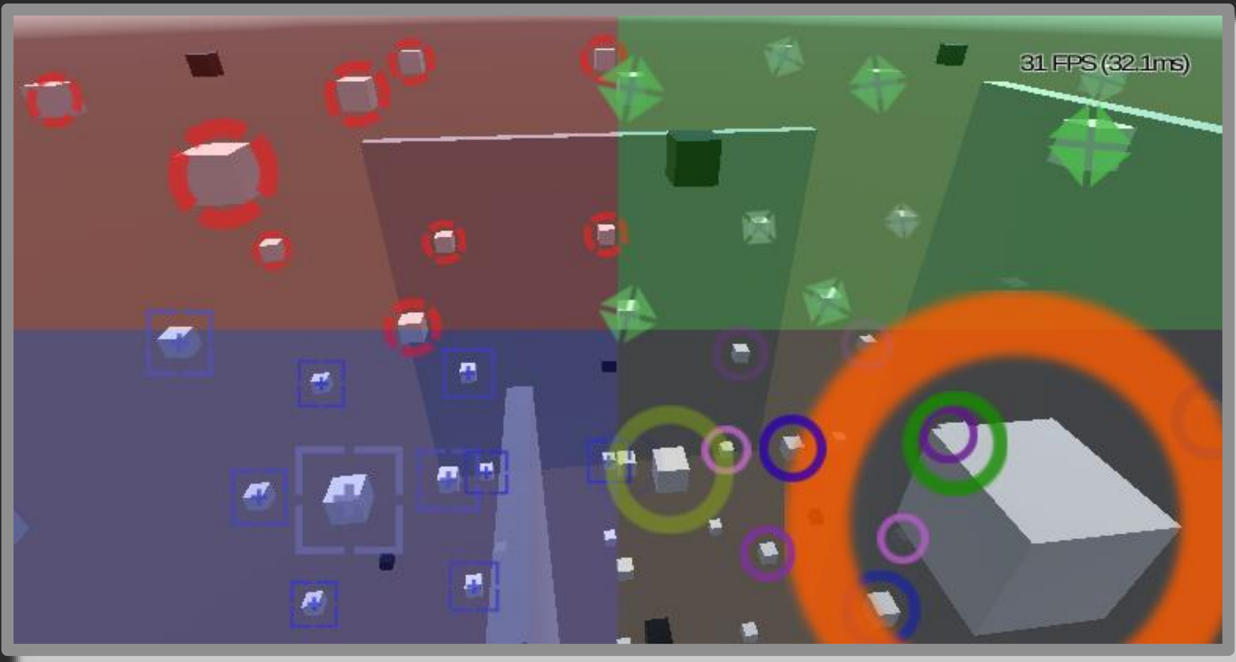
UI Radar

(Unity C# Script)

Documentation

By Roy BODEREAU (Kardux)



Summary

Foreword

First of all please allow me to apology in advance for my English. As non-native English speaker, I try my best to be understandable but I may fail sometimes: if you need any clarification on any point of this document, feel free to contact me and I'll be glad to help you as much as I can.

In this documentation, the following styles will be used:

- **bold** = Unity engine object (**GameObject**, **Canvas**, **Sprite**, etc…)

- ***bold/italic/grey*** = Parts of the script (***MarkerSprite***, ***MaxDistance***, etc…)

- *italic/green* = Paths (*"Assets"*, *"Assets\Editor"*, etc…)

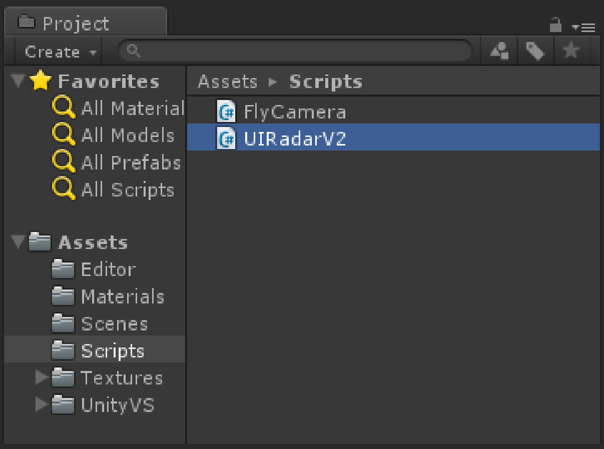
- **bold/red** = Important notes (**MUST**, **"I highly advise […]."**, etc…)

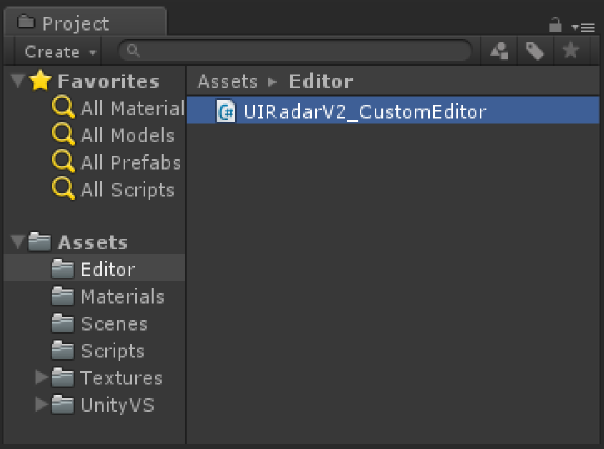
- *italic/red* = Notes (*"Keep in mind […]."*, etc…)

I - Usage

1 - Project setup

Simply copy the both script into your new/existing Unity project:



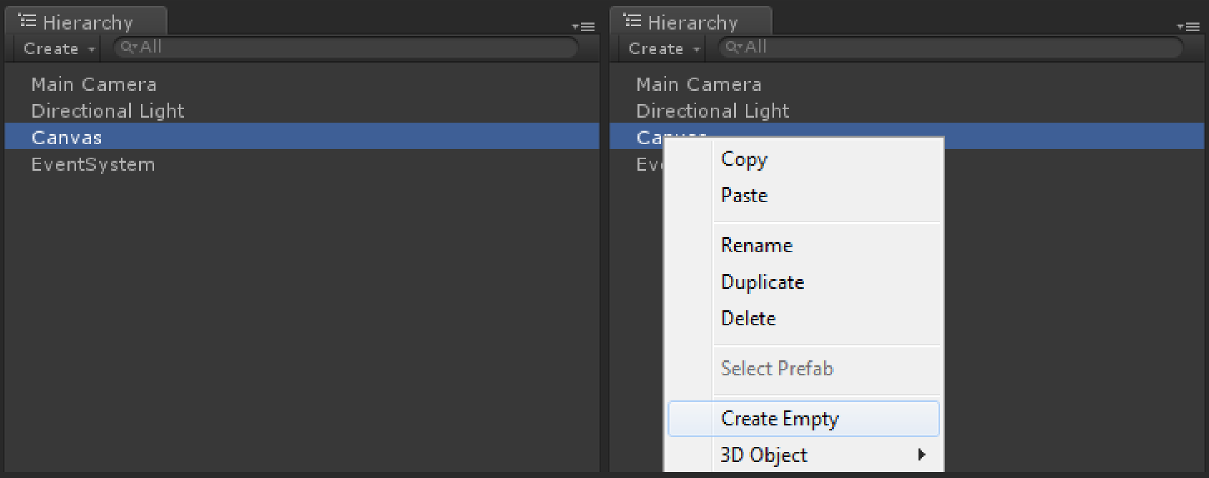


The **UIRadarV2.cs** file can go anywhere inside *"Assets"* folder or subfolders.

The **UIRadarV2_CutomEditor.cs** **MUST** be placed under *"Assets\Editor"*.

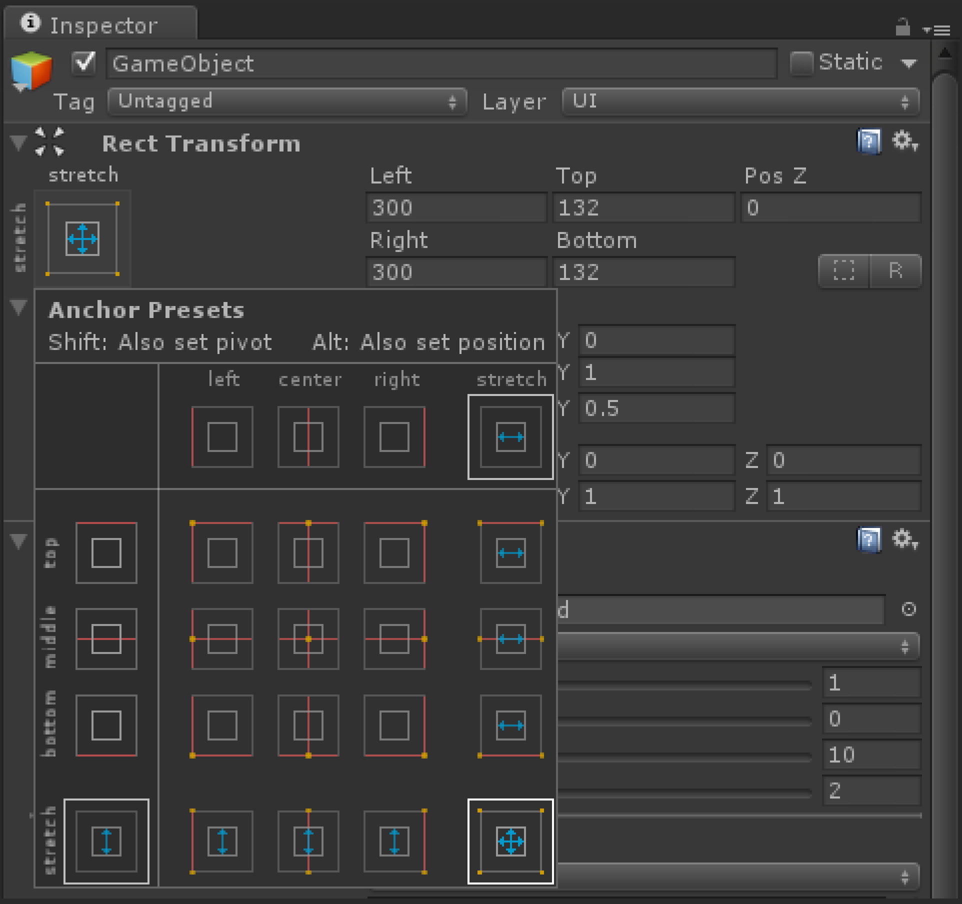
2 - Scene setup

1. On your new/existing scene, create a new **Canvas** (or use one you already have) and create an **Empty Object** inside this canvas:

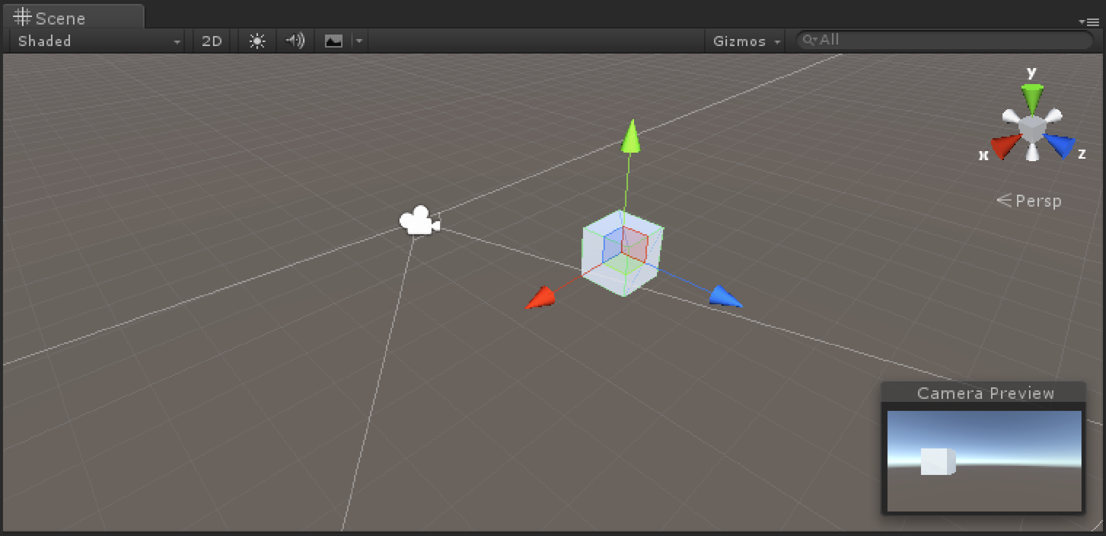


Keep in mind that using **Canvas***, the last component in hierarchy will be drawn on top of the others.*

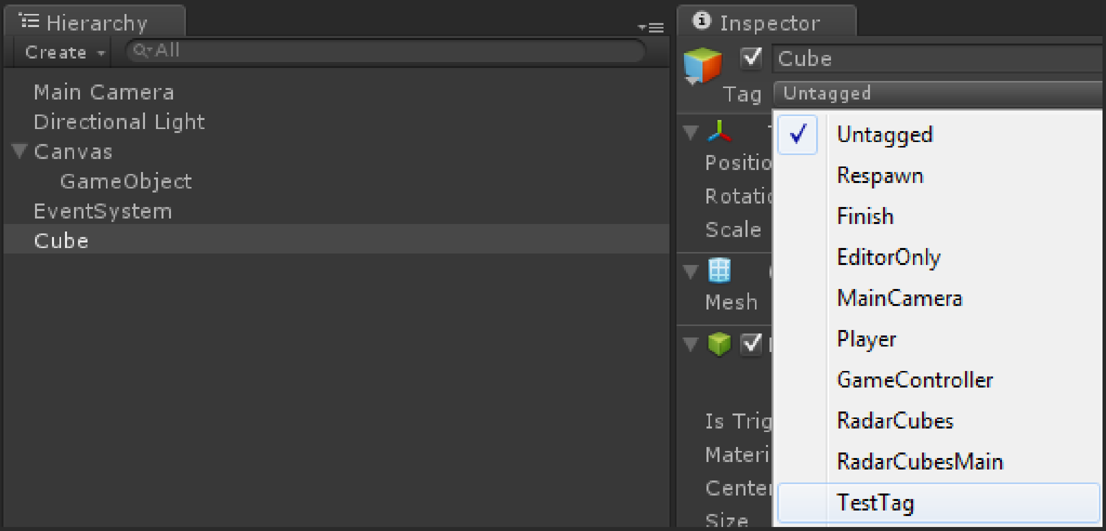
2. Change this object **Rect Transform** properties to "stretch" on both directions:



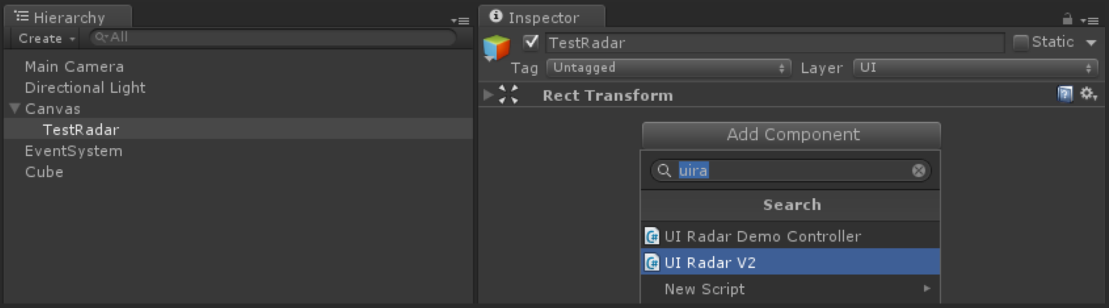
3. You can now add a test **Cube** to the scene and place it near your main **Camera**:



4. Now tag this cube with any **Tag** you want:

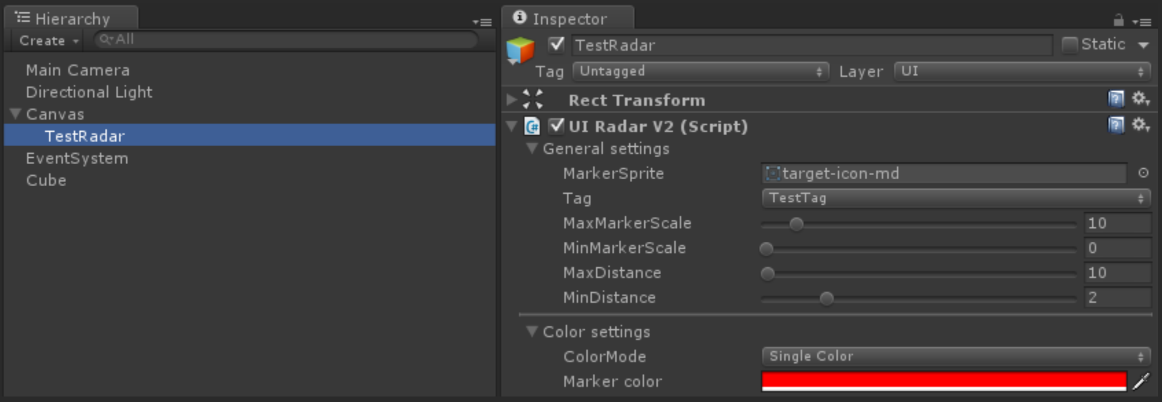


5. You can now add the *UIRadarV2.cs* **Script** to the empty object you created at step 1:



Also let's rename it "TestRadar".

6. Add a **Sprite** to the script and you choose the **Tag** you gave to the cube in step 4:



Hierarchy
Create ▾ All

Main Camera
Directional Light
▼ Canvas
TestRadar
EventSystem
Cube

Inspector

TestRadar — Static
Tag: Untagged — Layer: UI

Rect Transform

UI Radar V2 (Script)

General settings

MarkerSprite	target-icon-md
Tag	TestTag
MaxMarkerScale	10
MinMarkerScale	0
MaxDistance	10
MinDistance	2

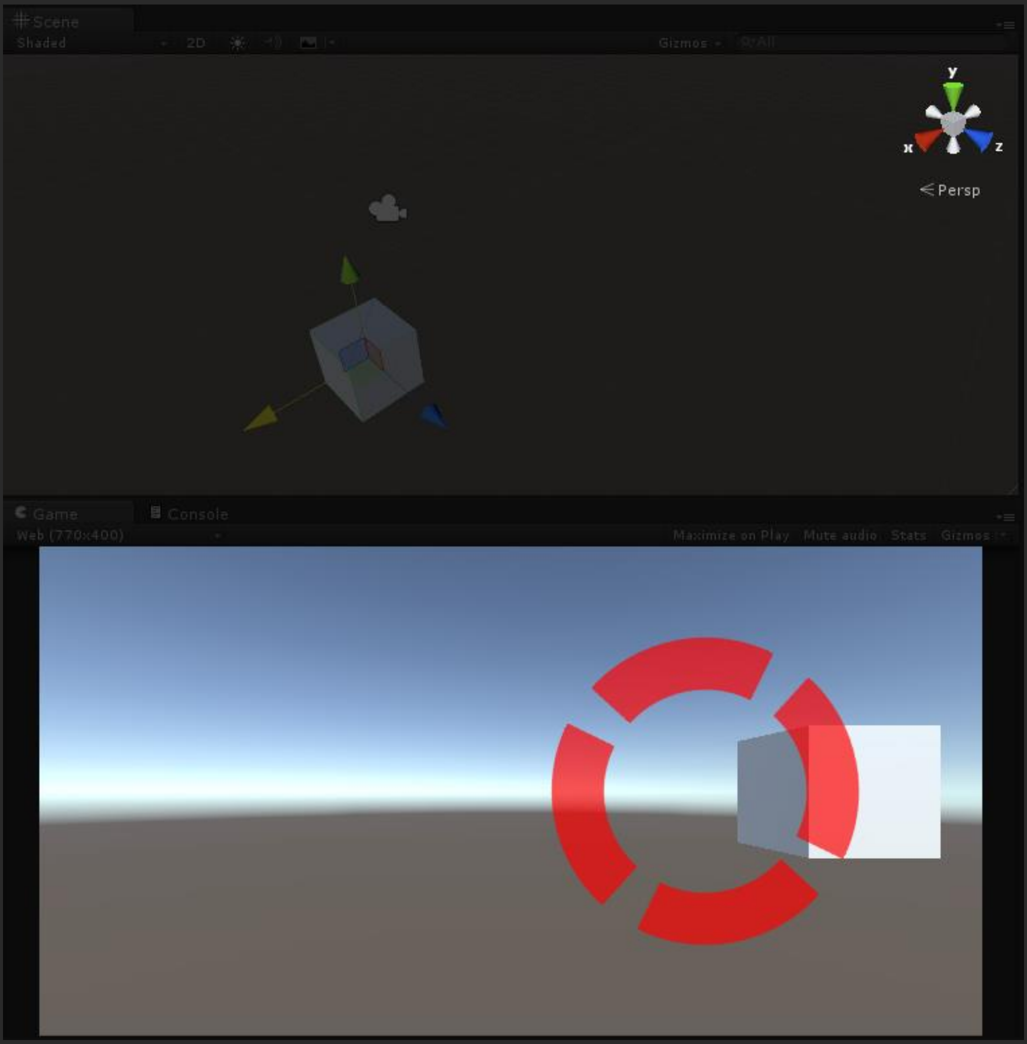
Color settings

ColorMode	Single Color
Marker color	🟥

You can also increase the ***MaxMarkerScale*** *a bit and give the* **Sprite** *a nice* ***MarkerColor****.*

To create a **Sprite***, simply import a texture (better with transparency such as .png) into your project and edit its* **Texture Type** *import settings.*

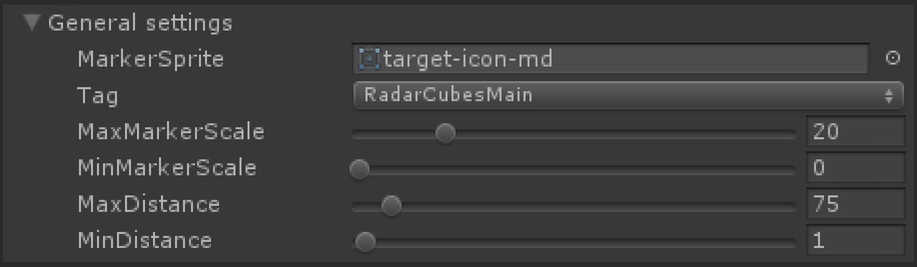
7. You scene setup is done! You can play it and try moving the cube (through the scene explorer) to watch your marker smoothly follow the cube:



II - Parameters

1 - General settings

This section contains the most important elements of the radar:



General settings	
MarkerSprite	target-icon-md
Tag	RadarCubesMain
MaxMarkerScale	20
MinMarkerScale	0
MaxDistance	75
MinDistance	1

- ***MarkerSprite*** = the Sprite that will be displayed

- ***Tag*** = the **GameObject** tagged with this **Tag** will be tracked by the radar

- ***MaxMarkerScale***/***MinMarkerScale*** = the maximum and minimum **Scale** of the markers (growing up when approaching)

- ***MaxDistance***/***MinDistance*** = at maximum distance, the markers **Scale** will be ***MinMarkerScale*** and grow to ***MaxMarkerScale*** when coming closer (if ***MinDistance*** is crossed, the markers **Scale** will be ***MinMarkerScale*** again)

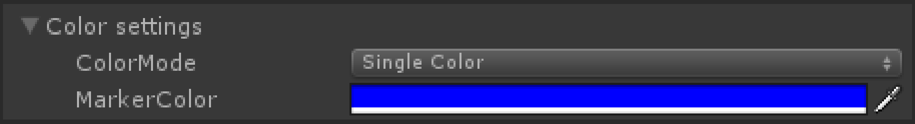
I highly advise to keep *MinMarkerScale* at 0 since with this value you'll keeping the appearing/disappearing effect on the marker.

2 - Color settings

This setting has 3 choices for the ***ColorMode***: Single Color - Simple Gradient - Multiple Colors

- Single Color

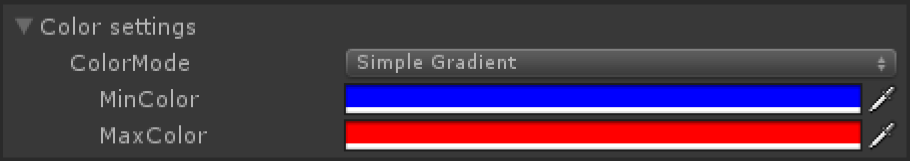
The markers will always keep the given color. You can adjust it with ***MarkerColor***:



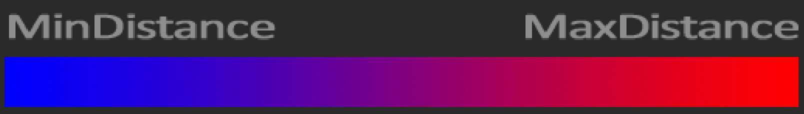
Color settings	
ColorMode	Single Color
MarkerColor	🟦

- Simple Gradient

The markers color will change from ***MinColor*** when being at ***MinDistance*** to ***MaxColor*** when going to ***MaxDistance***:

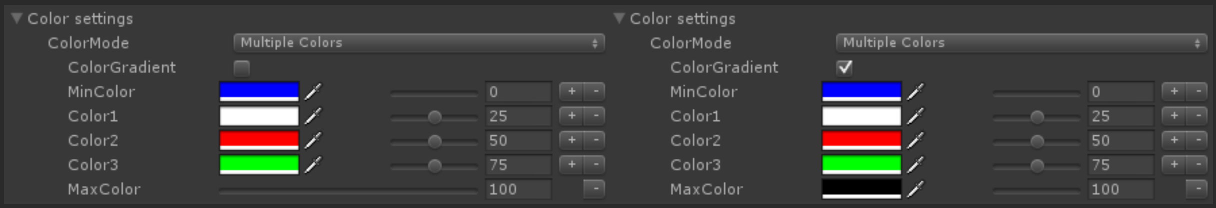


Color settings	
ColorMode	Simple Gradient
MinColor	🟦
MaxColor	🟥



- *Multiple Colors*

When using this ***ColorMode***, the color of the marker will change over distance passing through as many colors you want:



▼ Color settings				
ColorMode	Multiple Colors			
ColorGradient	☐			
MinColor	🟦		0	+ -
Color1	⬜		25	+ -
Color2	🟥		50	+ -
Color3	🟩		75	+ -
MaxColor			100	-

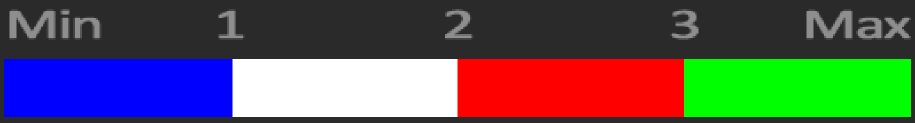
▼ Color settings				
ColorMode	Multiple Colors			
ColorGradient	✓			
MinColor	🟦		0	+ -
Color1	⬜		25	+ -
Color2	🟥		50	+ -
Color3	🟩		75	+ -
MaxColor	⬛		100	-

- When being at ***MinDistance***, the markers color will match ***MinColor*** and will progressively change to ***MaxColor*** when going away from the target.

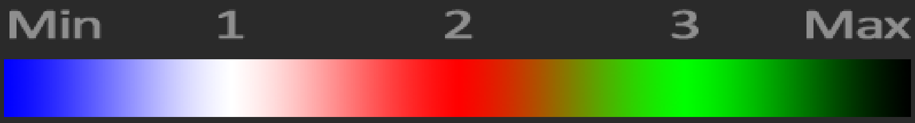
- You can change the percentages where you want the transition to take place (in this example I took 0% (which correspond to ***MinDistance***), 25% (which correspond to ***MinDistance + (MaxDistance - MinDistance) * 0.25***), 50% and 75%.

- You can also add or delete any color using the "+" and "-" button on the right of this section.

- Finally, the ***ColorGradient*** option allows you to choose between direct color changes or gradient-like transitions:



Did you see what I did there? (Hint: I'm French!)



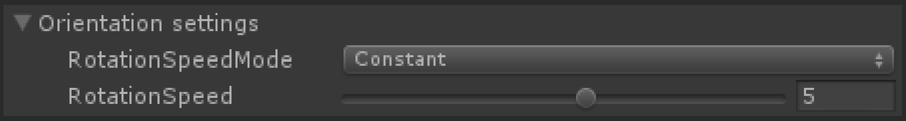
*Please note that when target's distance is smaller than **MinDistance** or greater than **MaxDistance**, the markers color will be set to **MinColor**.*

3 - Orientation settings

In this section you'll be able to choose the way the marker will rotate. Here again you have 3 choices for ***RotationSpeedMode***: Constant - Over Distance - Random

- *Constant*

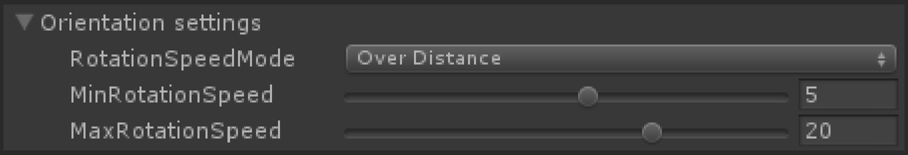
You set the value of the rotation speed (can be negative) and the marker will constantly rotate at this speed:



▼ Orientation settings
RotationSpeedMode: Constant
RotationSpeed: 5

- Over Distance

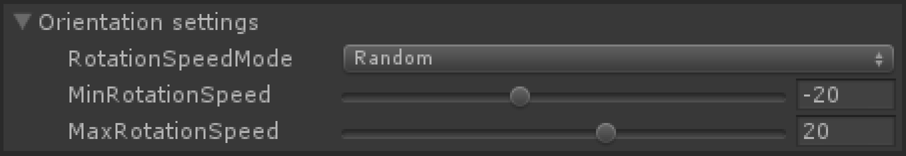
Here you'll have quite the same effect as scaling: at ***MinDistance*** the rotation speed will match ***MinRotationSpeed*** and will progressively change to ***MaxRotationSpeed*** when going away from the target:



Orientation settings
RotationSpeedMode: Over Distance
MinRotationSpeed: 5
MaxRotationSpeed: 20

- Random

When using this ***RotationSpeedMode***, the rotation speed will be set initially for each of the radar's markers. The value will be randomly picked between ***MinRotationSpeed*** and ***MaxRotationSpeed*** and stay constant during the scene:

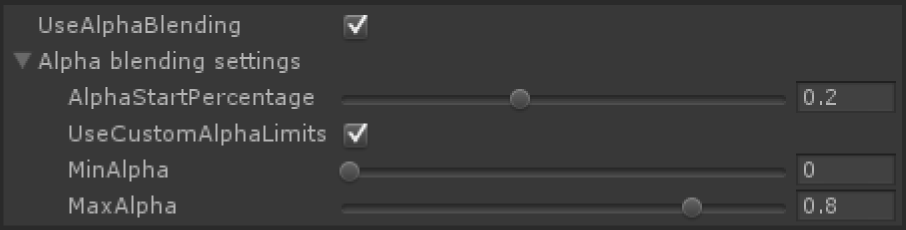


Orientation settings
RotationSpeedMode: Random
MinRotationSpeed: -20
MaxRotationSpeed: 20

4 - Alpha blending settings

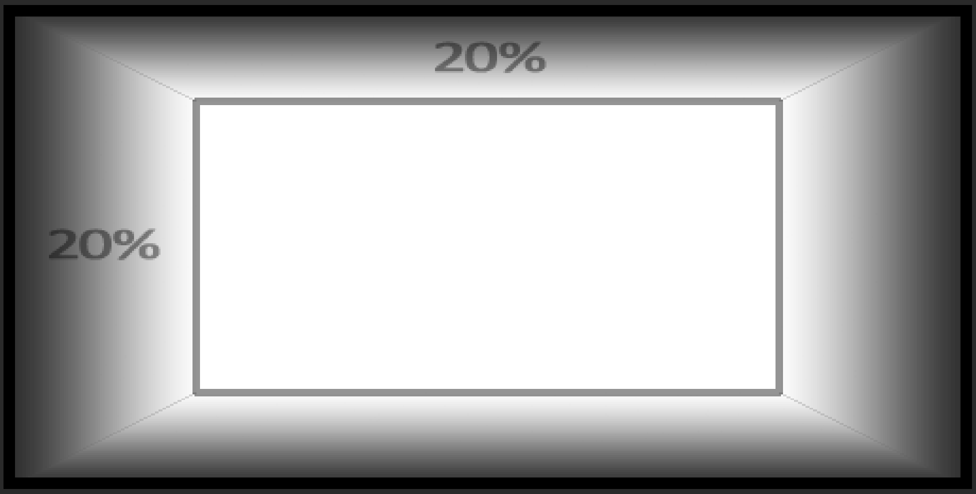
To enable this section, you'll have to toggle ***UseAlphaBlending*** to true.

Alpha blending allows your marker to smoothly fade out on the edges of the radar:



UseAlphaBlending: ✓
Alpha blending settings
AlphaStartPercentage: 0.2
UseCustomAlphaLimits: ✓
MinAlpha: 0
MaxAlpha: 0.8

- When setting AlphaStartPercentage to 0.2, this means the marker will start fading at 20% of the size of the radar:

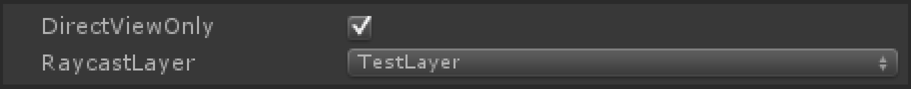


- The UseCustomAlphaLimits allows you to set your own alpha limits (by default 0.0 and 1.0). This way you can for example limit the MaxAlpha of your marker in order to keep it always a bit transparent.

I also advise keeping *MinAlpha* to 0 even if some cool effect can be achieved by having a positive value to *MinMarkerScale* and positive value to *MinAlpha*!

5 - Direct view

This simple option enable the "Direct view mode" and avoid the radar tracking obstacles hidden by other **GameObjects**. For more convenience, you can choose the Layer in which the **Raycasting** will be done:



DirectViewOnly	✓
RaycastLayer	TestLayer

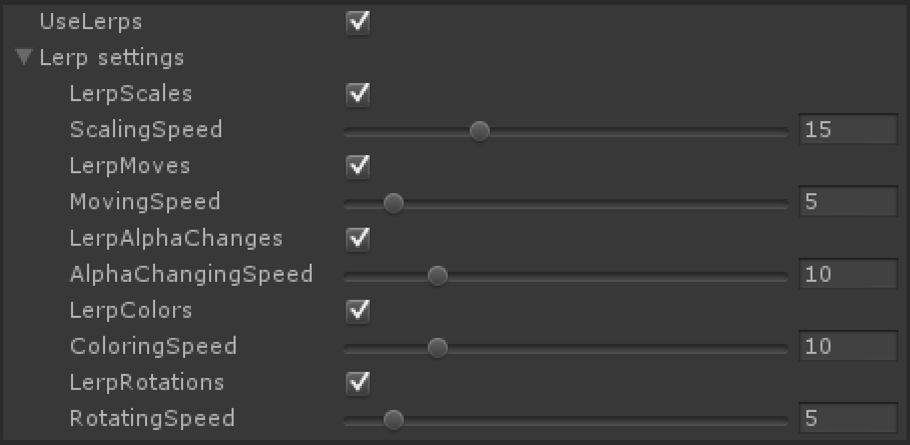
WARNING: this mode can sometimes have weird behaviour! (I'm still investigating on it)

Keep in mind that **Raycasting** *is a very heavy operation (in terms of computer resources) and can badly affect your FPS if used in bad conditions (such as many targets). Also don't forget that a* **Raycast** *needs a* **Collider** *in order to detect a* **GameObject***.*

6 - Lerp settings

To enable this section you'll have to toggle *UseLerps* to true.

This section is all about lerping and concerns all the previous sections. Almost all parameters can be lerped by simply enabling their matching toggles:



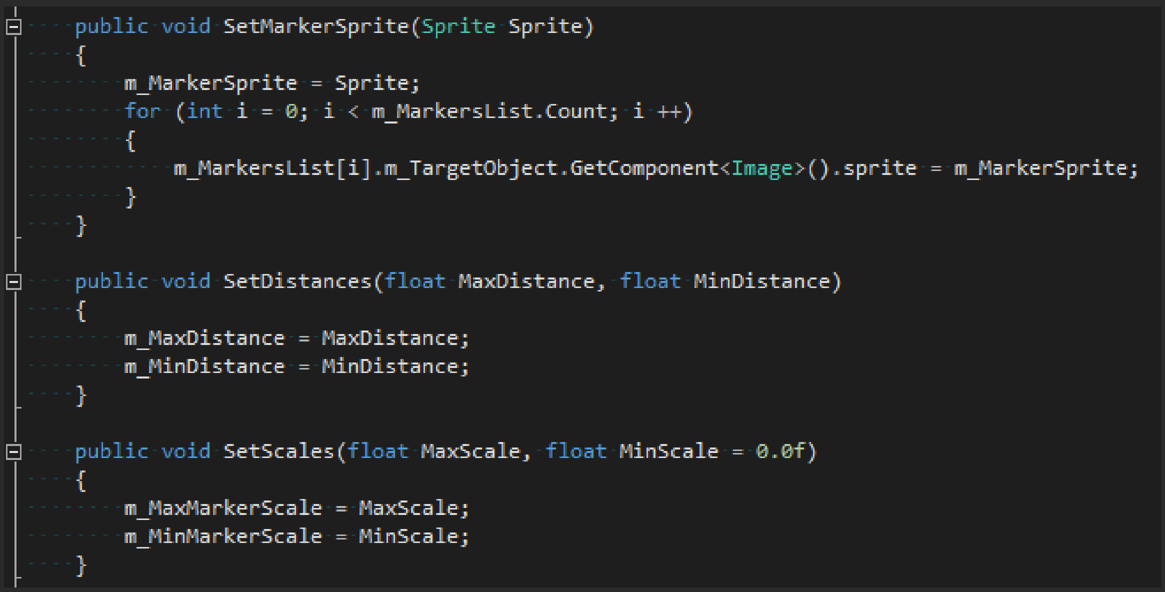
UseLerps	✓
Lerp settings	
LerpScales	✓
ScalingSpeed	15
LerpMoves	✓
MovingSpeed	5
LerpAlphaChanges	✓
AlphaChangingSpeed	10
LerpColors	✓
ColoringSpeed	10
LerpRotations	✓
RotatingSpeed	5

Such as Raycasting, Lerping is really a heavy operation, only use it if your support or your scene allows it: you may have huge FPS drops if you use it wrong.

III - Scripting

1 - Setter

To get this script user friendly as much as possible, I wrote some very simple setters:



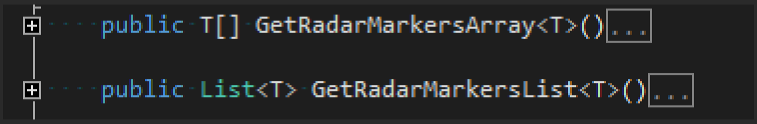
```
public void SetMarkerSprite(Sprite Sprite)
{
    m_MarkerSprite = Sprite;
    for (int i = 0; i < m_MarkersList.Count; i ++)
    {
        m_MarkersList[i].m_TargetObject.GetComponent<Image>().sprite = m_MarkerSprite;
    }
}

public void SetDistances(float MaxDistance, float MinDistance)
{
    m_MaxDistance = MaxDistance;
    m_MinDistance = MinDistance;
}

public void SetScales(float MaxScale, float MinScale = 0.0f)
{
    m_MaxMarkerScale = MaxScale;
    m_MinMarkerScale = MinScale;
}
```

Here we have ***SetMarkerSprite***, ***SetDistances*** and ***SetScales*** methods: they simply allow you to set the radar parameters while playing.

6 other methods are already written but you can add as many as you want/need.

2 - Getter

I also wrote 2 generic methods to access the ***Markers***. This way you can easily tweak them (some cool ideas could be adding them **Shadow** or **Outline**):



```
public T[] GetRadarMarkersArray<T>()...

public List<T> GetRadarMarkersList<T>()...
```

To call these methods, simply specify the kind of return you want:

- ***GetRadarMarkersList<GameObject>()*** => **List<GameObject>**

- ***GetRadarMarkersList<Image>()*** => **List<Image>**

- ***GetRadarMarkersArray<GameObject>()*** => **GameObject[]**

- ***GetRadarMarkersArray<Image>()*** => **Image[]**

Feel free to adapt/simplify/improve this script as much as you want but please don't forget I chose to release it under CC-BY-SA license: which means you'll have to share your own modified version.

IV - Links

- Project on **GitHub**: ***https://github.com/Kardux/UIRadar***

- Project **WebGL** demo: ***http://www.roy-bodereau.fr/hudradar_demo_en.html***

- Project thread on **Unity** forum: ***http://forum.unity3d.com/threads/hud-radar-[...]182186/***

- This documentation **URL**: ***https://github.com/Kardux/[...]/Documentation.pdf***