

■ Dynamic▲cons 1.1 - create your waypoints and indicators easily

by Procedural Imagination

First of all, thanks for your interest in **Dynamic Icons** tool and your support. I hope it helps during the development of **your next amazing game** and it meets all your expectations.

■ TOOL DESCRIPTION:

This **tool generates waypoints, indicators, and compasses** in your game that can help the player to find a distant target, a path to his destination, and all the things that can be pointed with an icon on the screen.

The tool includes four different scripts written in C#:

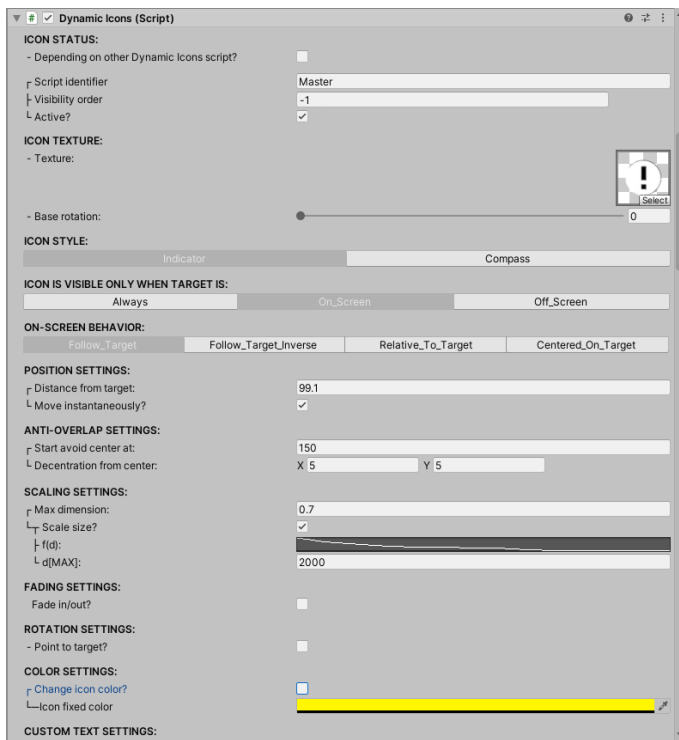
- **DynamicIcons**
- **DIWorldCompass**

- **DIManager**
- **DIDestroyer**
- **DICustomEditor**
- **DIWCCustomEditor**

and 2 custom DLL:

- **DLL_MathExt**
 - **DLL_Uilities**
- ← Click on the DLL name to see the source code.
You can compile it by yourself using NET. Core 2.0.

The first two scripts are the only ones you have to manually add respectively to yours targets and main camera.



■ HOW TO SET UP DYNAMIC ICONS SCRIPT:

DynamicIcons extends Unity's **MonoBehavior** and it's the **core script**.

You only need to add it as a component to a **GameObject** and set it up in the editor to automatically generate the icon with its behavior when the game starts.

There are plenty of options and parameters to set up, so let's start to see all of them.

First of all, please note that **DICustomEditor** will modify the

editor interface on the fly depending on the actual configuration.

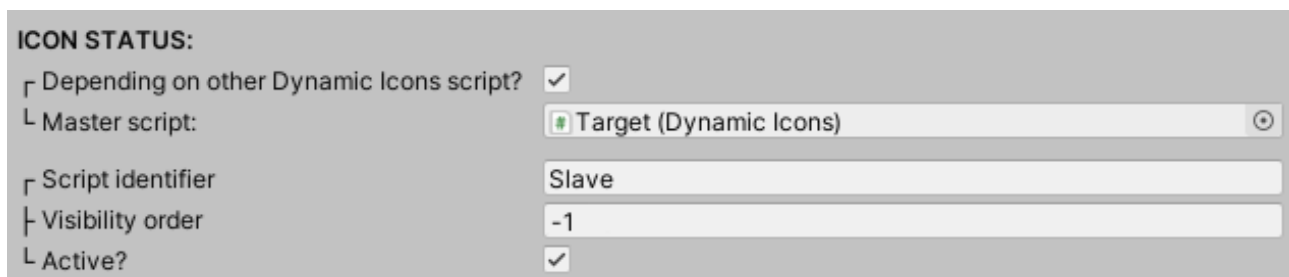
That allows to show you only the fields that have some impact on the icon and texts.

As you can see, you don't need to create a canvas or a **GameObject** with a **RawImage** component. **DynamicIcons** will **auto-generate** what it needs.

The first section is **ICON STATUS**.

To move the icon, **DynamicIcons** need to calculate some values like the distance between the target and the camera.

You can always add a new instance of **DynamicIcons** to the same object and often some settings and calculations are the same for all the scripts.



If « Depending on other Dynamic Icons script? » is true, you should drag on the « Master script: » field the **DynamicIcons** component that will send the information to this one.

If the field has a null reference, nothing happens and the component will work independently from the others.

« Script identifier » is just the **name of this instance** of **DynamicIcons**. It will

be used in the auto-generated hierarchy. It's not mandatory to add a name but is strongly recommended to distinguish multiple instances of DynamicIcons.

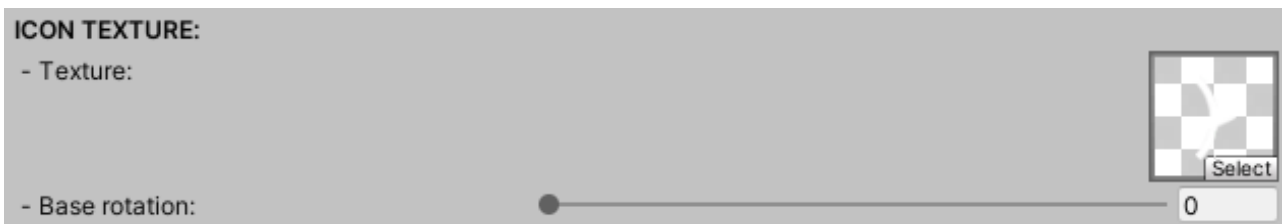
«Visibility order» is an integer starting from -1. **Icons with a higher number will appear above those with a lower number.** There will be a small delay during initialization equal to the highest number of «Visibility order».

For example, if it's 10, the icon generation will complete after 10 frames. By adding a new instance of DynamicIcons during the game, the new icon will be always placed on top of all the others (in this case «Visibility order» hasn't any effect).

« Active? » is self-explanatory. **When true the icon and text are visible and their positions, colors, transparencies, and strings update every frame.**

You can access this public variable from another script and activate and deactivate it when needed (for example, you can use DynamicIcons to show a pointer only when the camera is locked on the target).

In the **ICON TEXTURE** section there are the parameters related to the icon texture.



« Texture » is the **2D texture of the icon.** It should point to the right, but you can correct its rotation using the « Base rotation » slider that goes from 0 to 360 degrees.

The texture should be white to change its color and have a transparent background.

The **ICON STYLE** section contains two buttons: « Indicator » and « Compass ».



If the first one is selected, **the icon follows the target.**

If you select the second one, **the icon will stay in a fixed position in the screen pointing to the target.**

Both of them have specific sections to customize the behavior of the icon.

The next four sections (**ICON IS VISIBLE, ON-SCREEN BEHAVIOR,**

OFF-SCREEN BEHAVIOR, and POSITION SETTINGS) are accessible only when « Indicator » is selected.

ICON IS VISIBLE:

Always On_Screen Off_Screen

ON-SCREEN BEHAVIOR:

Follow_Target Follow_Target_Inverse Relative_To_Target Centered_On_Target

OFF-SCREEN BEHAVIOR:

Along_Screen_Border Circular_Rotation

- Circle eccentricity: 1

POSITION SETTINGS:

Distance from screen border: 25

Distance from target: 99.1

Fixed angle: 206

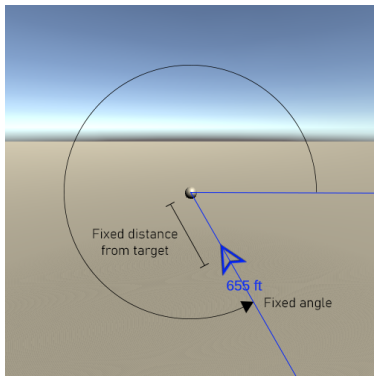
Move instantaneously? ☐

Movement speed: 10

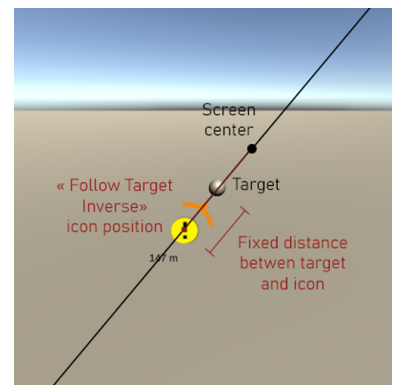
The first three buttons determines when the icon should be visible. If « Always » or « On Screen » is selected, the icon will follow the target on the screen based on the next four settings:

- « Follow Target »
- « Follow Target Inverse »
- « Relative To Target »
- « Centered On Target »

With « Follow Target » or « Follow Target Inverse »



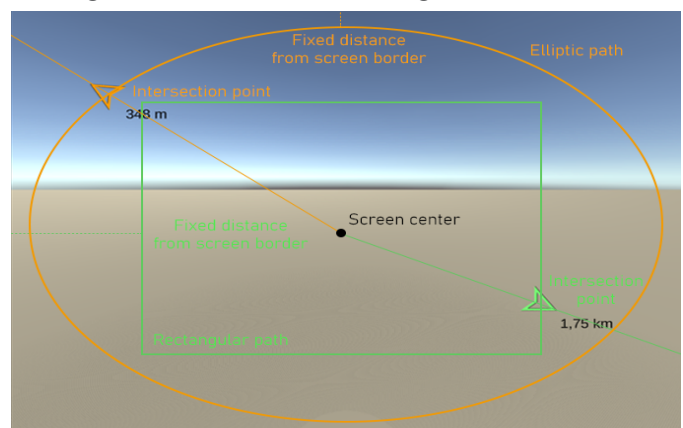
(example on the right) the icon position is calculated from the position of the target in the screen plus the normalized direction from the target to the screen center multiplied for the « Distance from target » value (or its opposite).



« Relative To Target » puts the icon in a **fixed position from the target**. That position can be set with the « Distance from target » and « Fixed angle » values.

« Centered On Target » sets the icon in the **same position as the target** on the screen.

If « Always » or « Off Screen » is selected, the icon will indicate the direction that goes from the screen center to the target position. If « Along Screen Border » is selected, the icon position it's calculated as



the **intersection point** between the right, up, left, or down screen line limit and the direction between the screen center and the target position.

Else if « Circular Rotation » is selected, the icon will **rotate around the screen center** following a circular or elliptic path based on the « Circle eccentricity » value (0 = circle, 1 = ellipse). « Distance from screen border » sets a fixed space between the icon and the screen limits.

The last two settings are « Move instantaneously? » and « Movement speed ». If the first is true, the icon position updates to the new one every frame. If false, the position is linearly interpolated using the last value.

The section **ANTI-OVERLAP** is accessible only when « Compass » or « Follow Target Inverse » is selected.

ANTI-OVERLAP SETTINGS:

☐ Start to avoid center at: 150

☐ Decentration: X 5 Y 5

The two parameters **avoid the icon from cover the target** when it's near the screen center.

« Start to avoid center at » is a pixel distance from the screen center. When the distance between the target position on the screen and the screen center is lower than this value the screen center coordinates are moved by the fixed amount « Decentration ».

The default values should be ok for almost every purpose (the script was principally tested with a 2560*1080 resolution).

COMPASS TYPE:

☒ Arrow ☐ Horizontal ☐ Vertical

The section **COMPASS TYPE** is accessible only when « Compass » or is selected.

COMPASS POSITION SETTINGS:

☐ Horizontal position from screen center: 0

☐ Vertical position from screen center: 0

- Rotation correction: X 0 Y 0 Z 0

« Arrow » is a **fake 3D compass that always points to the target** (the icon can rotate around its three axes) from a fixed position. The icon position in the screen can be set using the sliders « Horizontal position from screen center » and « Vertical position from screen center ». Both go from - 1 to +1. That value is multiplied respectively for half of the horizontal resolution and half of the vertical resolution. That means that positive values move the icon up and to the right, while negative ones move it down and to the left.

« Rotation correction » is a Vector3. The x, y, and z values rotate the icon on

its respective axes. Use them to correct the icon position till it points to the target (if the texture points to the right the correction should be [90, -90, 0]).

HORIZONTAL COMPASS POSITION SETTINGS:

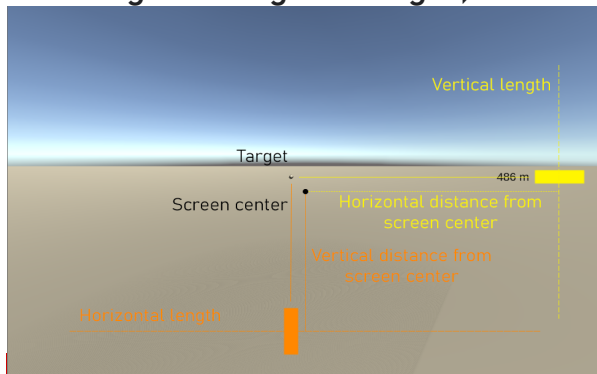
- Horizontal length: 0.005
- Vertical position from screen center: 0

VERTICAL COMPASS POSITION SETTINGS:

- Vertical length: 0.846
- Horizontal position from screen center: 0.851

If « Horizontal » or « Vertical » is selected, respectively the section **HORIZONTAL COMPASS POSITION SETTINGS** or **VERTICAL COMPASS POSITION SETTINGS** is accessible.

The first slider sets the length of the line on which the icon can move along showing the target's height, while the second the position in the screen from its center.



You can see in the image on the left two instances of DynamicIcons applied to the same target.

A horizontal compass is often used in videogames to combine the marker with the cardinal points.

SCALING SETTINGS section contains the parameters related to the icon size. « Max dimension » is the **maximum size the icon can reach** if « Scale size? » is true.

SCALING SETTINGS:

- Max dimension: 0.53
- Scale size? ☒
- f(d): [Graph showing a curve from 0 to 1]
- d[MAX]: 2000

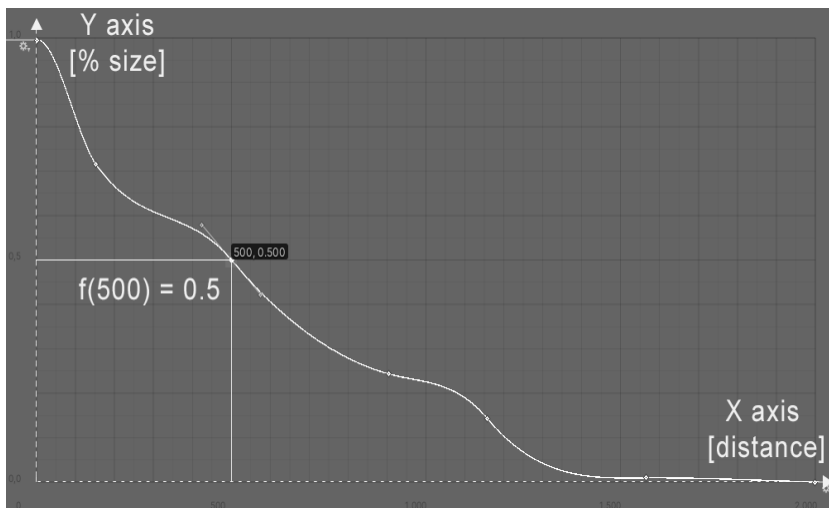
« f(d) » is a **custom curve used to scale the size of the icon** according to the distance from the target and « d[MAX] » is the maximum of its x-axis.

Understand what the curve « f(d) » does and how to set it up can be a little tricky, but it allows complete customization.

The x-axis of the curve is the distance between the main camera and the target GameObject. It goes from 0 to the « d[MAX] » value and, after that limit, the icon will be always deactivate to reduce the computational load.

The y-axis goes from 0 to 1 and multiplies the « Max dimension » value (the max size of the icon).

In the example above you can see that a distance of 500 (it's the magnitude of the direction from the camera to the target) $f(500) = 0.5$ and the icon reduces its size by 50% ($0.5 \equiv 50\%$).



If « Scale size? » is true, also the space between the target and the icon on the screen (set it up with « Distance from target ») is reduced and increased according to the distance in the same way.

Please note that at the maximum distance « d[MAX] » the curve

should value 0.

FADING SETTINGS contains the boolean value « Fade in/out? ».

FADING SETTINGS:	
<input checked="" type="checkbox"/> Fade in/out?	<input checked="" type="checkbox"/>
<input type="checkbox"/> Min distance to be visible:	<input type="text" value="50"/>
<input type="checkbox"/> Max distance to be visible:	<input type="text" value="500"/>
<input type="checkbox"/> Instantaneous fading?	<input type="checkbox"/>
<input type="checkbox"/> Fading speed:	<input type="text" value="10"/>

If it's true, the icon will be visible only when the distance between the camera and the target is higher than « Min distance to be visible » and lower than « Max distance to be visible ».

Please note that the icon will be always not visible when the distance it's over the « d[MAX] » value (so remember to respect the following inequality: « Min distance to be visible » < « Max distance to be visible » < « d[MAX] »).

If « Instantaneous fading? » is true, the icon appears and disappears immediately.

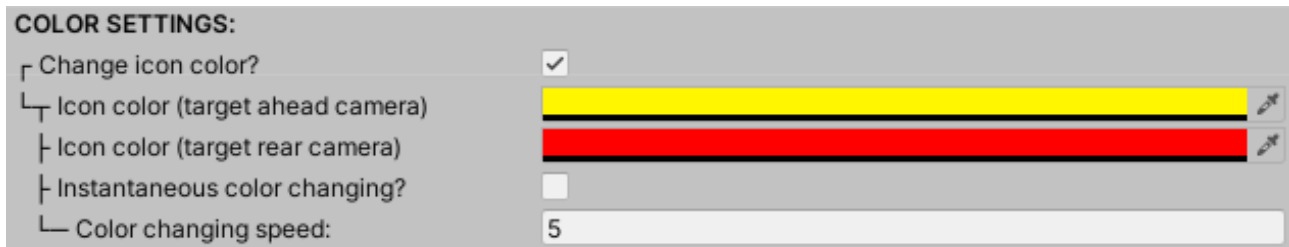
If it's false, the icon changes its transparency according to the « Fading speed » value.

ROTATION SETTINGS:	
<input checked="" type="checkbox"/> Point to target?	<input checked="" type="checkbox"/>
<input type="checkbox"/> Instantaneous rotation?	<input type="checkbox"/>
<input type="checkbox"/> Rotation speed:	<input type="text" value="30"/>

If « Point to target? » is true, the right axis of the icon's RawImage points to the target, and the parameter « Instantaneous rotation? » of the ROTATION SETTINGS section will be available.

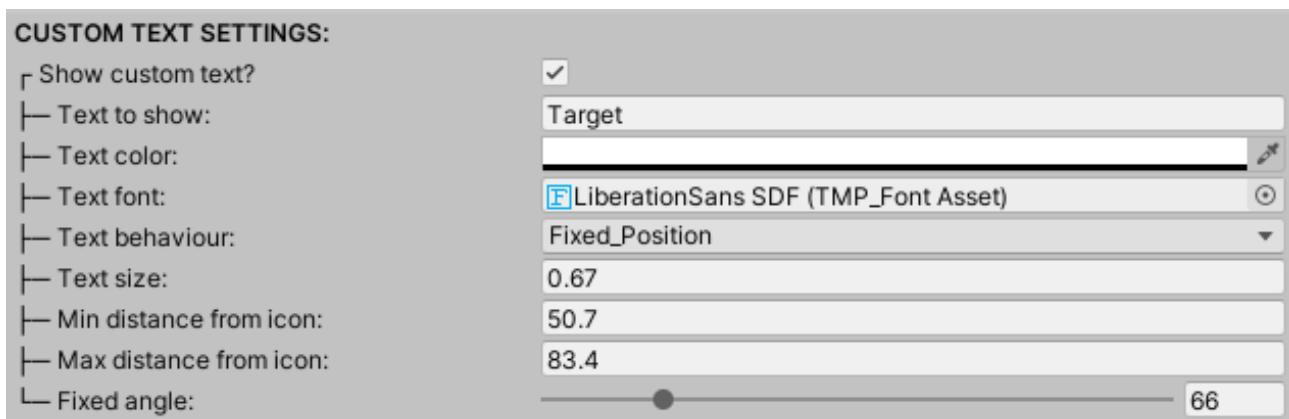
If the last boolean variable is false, the icon rotates according to the « Rotation speed » value.

The last section is related to the icon is COLOR SETTINGS. If « Change icon color? » is false, the icon keeps a single custom color. If it's true there will be two color fields: « Icon color (target ahead camera) » and « Icon color (target rear camera) ».



The first one is the color that the icon has when the target is in front of the main camera. When the target is behind the camera the icon changes its color to the second one.

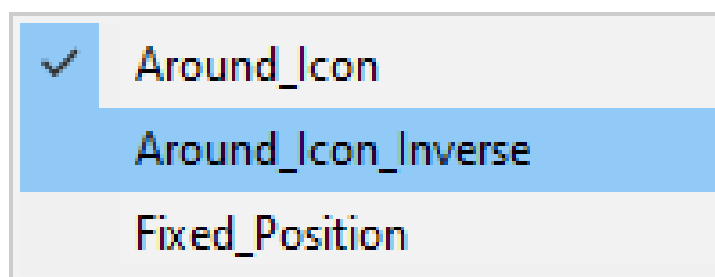
If « Instantaneous color changing? » is false, the icon changes its color according to the « Color changing speed » value.



CUSTOM TEXT SETTINGS its the first section related to the texts settings. If « Show custom text? » is true, **a personalized text will be visible near the icon.**

You can write the text to display inside the « Text to show » field and choose its color (« Text color »), font (« Text font »), and size (« Text size »).

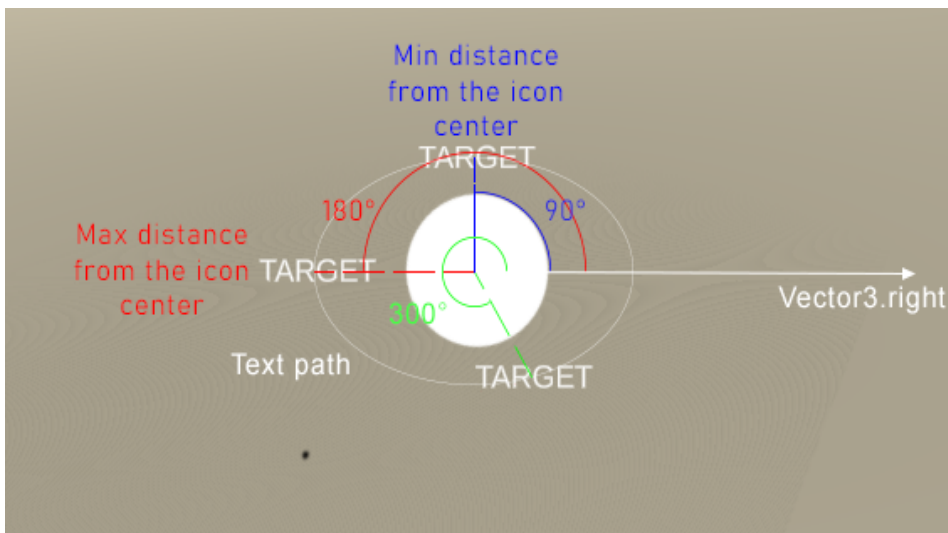
There are three different « Text behavior » options:



- « Around Icon »
- « Around Icon Inverse »
- « Fixed Position »

The first two let the text dynamically move around the icon.

The text position is the intersection point between an ellipse having the icon as its center and a line passing through the target and icon position on the screen.



« Fixed Position » does something similar, but the icon will stay in a fixed position relative to the icon because the angle in degrees between the `Vector3.right` and the line is set it up using the

« Fixed angle » slider.

The text width and height are different, so you can modify the elliptic path around the icon using « Min distance from icon » and « Max distance from icon » values, as you can see in the image above.

DISTANCE TEXT SETTINGS:	
Show distance?	<input checked="" type="checkbox"/>
Distance unit:	Kilometers
Unit size:	1
Text color:	<div style="background-color: black; width: 100px; height: 15px;"></div>
Text font:	None (TMP_Font Asset)
Text behaviour:	Fixed_Position
Text size:	0.52
Min distance from icon:	45.9
Max distance from icon:	63.7
Fixed angle:	<div style="display: flex; align-items: center;"> <div style="flex-grow: 1; border: 1px solid black; position: relative;"> <div style="position: absolute; left: 0; top: 0; bottom: 0; width: 10px; background: linear-gradient(to right, black 49%, white 49%, white 51%, black 51%);"></div> </div> <div style="width: 50px; text-align: center;">0</div> </div>

The DISTANCE TEXT SETTINGS is the last main section. The text positioning works as the previous one. The only difference is if « Show distance text? » is true, **the distance between the main camera and the target GameObject will be displayed** around the icon.

<input type="checkbox"/>	Meters
<input checked="" type="checkbox"/>	Kilometers
<input type="checkbox"/>	Feet
<input type="checkbox"/>	Miles

« Distance unit » automatically converts the distance from the Unity standard uniy (1 Unity unit = 1 meter) to meters, kilometers, feet and miles. If you need to use a different conversion (1 Unity unit = x meters) you can set it with « Unit size ».

■ HOW DYNAMIC ICONS WORKS:

When the game starts, the icons textures and texts are initialized.

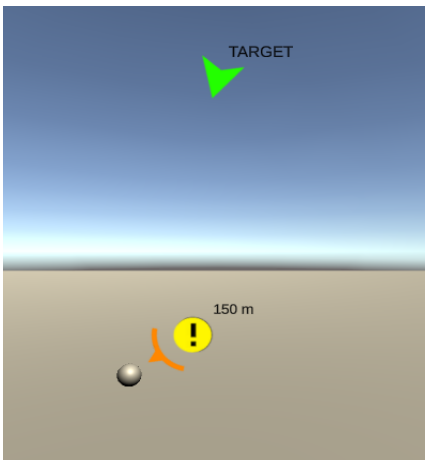
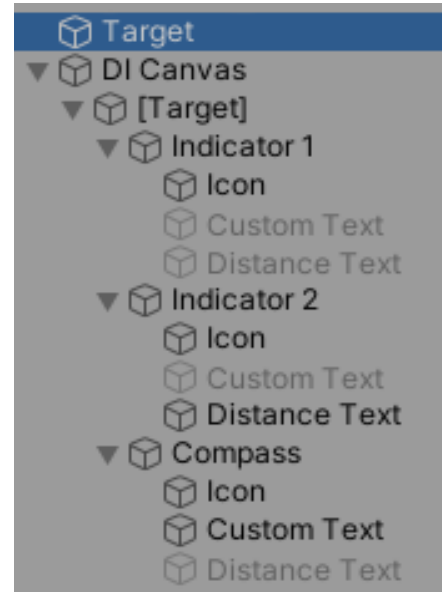
Every GameObject that has at least one DynamicIcons as component checks if its target GameObject has a DIManager as a component. If not, a new one is added to the target.

During the Awake() function routine, the DIManagers tries to generate a new UI canvas, called "DI Canvas", if it doesn't exist.

Then an empty GameObject is added as a child and named after the target.

Every DynamicIcons instance creates a second empty GameObject named as the « Script identifier » string you have chosen (by default is "Marker").

Those GameObject are ordered following the chosen «Visibility order» (starting from -1, icons with lower number display under the icons with a higher number).



Finally, inside every last empty GameObject, there are the icon RawImage, the custom text TextMeshPro, and the distance text TextMeshPro.

The example on the right shows a GameObject ("Target") with three instances of DynamicIcons ("Indicator 1", "Indicator 2", and "Compass").

As you can see in the example the auto-generated DI Canvas contains an empty GameObject called [Target] that contains, in turn, the icons and texts of every instance separately.

The image on the left shows you the final results.

Two icons (the yellow and orange ones) point to the target GameObject, while the third (the green one) is a fake 3D compass.

Using multiple instances of DynamicIcons on the same GameObject allows you to create more complex behaviors.

The GameObject created for every target gets automatically the component "DIDestroyer". It's a simply destruction function that activates when the reference to the target doesn't exist anymore. It avoids seeing the target icons and texts after it got removed from the game.

You can always add a new instance of DynamicIcons to a GameObject during

the gameplay or add to the game a prefab that contains one or more DynamicIcons as a component.

Please note that a new icon added to an already existent target will be always displayed over all the other ones.

If you have multiples DynamicIcons added to a single GameObject, and all of them use similar settings, one can become the master script for all the others.

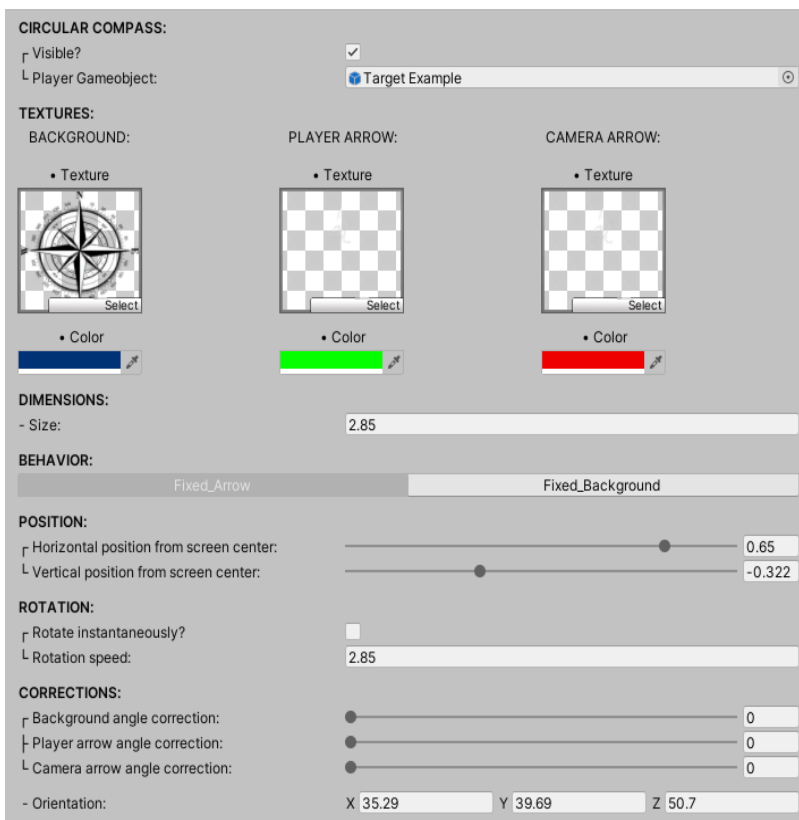
The next parameters are copied from the master script ones:

- **ICON STYLE**
- « Active? »
- « Circle eccentricity »
- « Start to avoid center at »
- « Decentration »
- « Scale size? »
- « f(d) »
- « d[MAX] »
- « Fade in/out? »
- « Min distance to be visible »
- « Max distance to be visible ».

Most important, the curve « f(d) » is evaluated only one time reducing the computation cost.

■ HOW TO SET UP DI WORLD COMPASS SCRIPT:

The first section **CIRCULAR COMPASS** contains the options for the classic compass.



If « Visible? » is true, the compass will be visible and the next settings accessible.

It uses three textures: a background and two arrows to show the player's forward axis and the camera's one (by default the global Z-axis is the north direction).

In the left image, you can easily see the chosen textures and their colors (the color is applied on the texture only if it's white). **Remember to use squared texture.**

The « Player GameObject »

field can be left empty (if you don't have a player model or don't want to show where it's looking at).

« Size » is a float value that you can use to increase or decrease the compass dimensions.

The compass has two behaviors. The first one keeps the camera arrow in a fixed position and orientation and only the background and player's arrow texture rotate. Instead, the second one keeps the background will stay in a fixed position and orientation while the two arrows rotate.

The compass is anchored on the screen center and with the next two sliders, you can move it horizontally and vertically.

ROTATION settings allow setting the speed rotation of the compass. If « Rotate instantaneously? » is true, the compass or the arrows will point to the camera or player model direction immediately, else you can use the « Rotation speed » value to set the speed of the linear interpolation.

The three sliders in CORRECTION is used to rotate the three texture around its Z-axis and correct its orientation. Those settings are useful if the textures are not perfectly oriented (by default, north is up).

« Orientation » is a Vector3 and its values are used to rotate all the compass around the X, Y, and Z axes. You can use it to change the texture's perspective.

The second and last section **HORIZONTAL COMPASS** contains the options for the horizontal compass.

If « Visible? » is true, the horizontal compass will be visible and the next settings accessible.

This compass uses three textures. The first one is the background texture, the second texture is the cardinal points texture, and the last is a texture that is visible over the other two.

You can leave empty the unutilized texture fields.

You can set the compass dimensions in the homonym section. As you can see in the image on the right, the compass can adapt its length to the screen width. In that case, the horizontal compass width is a percentage of the screen width. Otherwise, you have to set a fixed width for the compass. The height is ever a fixed value.

« Texture stretching » value is used to stretch the cardinal points texture (if it's 0, all the texture is visible inside its rect).

HORIZONTAL COMPASS:

- Visible? ☒

TEXTURES:

- BACKGROUND:**
 - Texture: [Select]
 - Color: [Color picker]
- CARDINAL:**
 - Texture: [Select]
 - Color: [Color picker]
- OVERLAY:**
 - Texture: [Select]
 - Color: [Color picker]

DIMENSIONS:

- ☒ Adapt to screen?
- ☐ Fixed height: 29.3
- ☐ Relative width: 0.424
- ☐ Texture stretching: 0

POSITION:

- Vertical position from screen center: 0.806

ROTATION:

- ☐ Rotate instantaneously?
- ☐ Rotation speed: 4.83

CORRECTIONS:

- ☐ Background relative height: 0.424
- ☐ Background relative width: 1.018
- ☐ Overlay relative height: 1.18
- ☐ Overlay relative width: 0.631
- North correction: 0.585

As for the previous compass, even **the horizontal one is anchored to the screen center**. However, you can't move it horizontally but only vertically. Rotation settings set the speed at which the cardinal texture will move horizontally.

Background and overlay texture dimensions are calculated as a percentage of the cardinal texture size.

You can use the last parameter "North Correction" to correct the compass orientation.

■ HOW DI WORLD COMPASS WORKS:

DIWorldCompass generates a hierarchy as the Dynamic Icons script. The GameObjects that contain the RawImages of the circular and horizontal compasses are inside the "DI Canvas" with the other icons.

The script works only if you add it to the main camera.

The circular compass gets the rotation around the Y-axis of the main camera transform.

That value is used to orientate the compass background or the arrows (in the example the camera arrow is a texture that shows a fake area visible from the camera, but you can use every texture you want to), depending on the chosen behavior.



The upper image shows an example of both circular and horizontal compasses.

You can rotate the circular compass around the global axis to change its perspective, you can correct the north position using a slider (if the image is not perfectly oriented, it's a fast way to correct it) and the compass can update its orientation instantaneously or slowly.

The horizontal compass is anchored to the screen center and it can be moved up or down. Its width can automatically scale with the screen width or maintain a fixed size.

N NE E SE S SW W NW **The horizontal compass itself must be a texture with all the cardinal points.**

The UV-Rect horizontal value can be modified to show only a portion of the texture (that gives a stretched effect to the texture) or all the texture.

As the circular compass, the north position can be easily corrected using a slider option in the editor.

The other two textures of the horizontal are facultative, but they allow you to add an image under and one over the compass texture.

■ CONTACTS:

If you have any problem or question, send an e-mail to procedural.imagination@gmail.com.

Write a Review



Dynamic Icons - Easily create your waypoints, indicator, and compasses

★★★★★ Click to rate

Please, if you liked Dynamic Icons, [leave a review](#). Your feedback will help me to improve the tool and the

package will get higher visibility on the Asset Store. :)