

# ProCOMPASS: Simple & Minimal Navigator

V 1.3.0

**ProCOMPASS: Simple & Minimal Navigator** is simply a compass and navigator tool that can be implemented in your game easily. Most importantly, you can add indicators to track some object's position on the compass. You can also set the behaviour of each indicator on compass on specific events according to your need. This tool also provides the opportunity to use minimap, which means you can draw a custom map of your scene and use it as a minimap on the compass. All you need to do is follow some simple steps. This tool is also lightweight because it is designed to do calculations as little as possible with efficient algorithms. There is a customizable viewing option that will allow you to customize the basic viewing system of the compass. You can also use multiple compass at the same time. Two different compass skins are included in this package. Even if the styles don't match your game, you can make your own skins by replacing your designed sprites with the one used in it and following the hierarchy structure.

## Key Features:

This tool is designed to be as simple as possible. The main features of **ProCOMPASS: Simple & Minimal Navigator** are:

- A basic rotating compass
- Object tracker (Indicators) with three customizable behaviour (***Clip Indicator***, ***Show direction when outside*** or show direction of the gameObject when not in viewing area, ***Do nothing*** or only show indicator when the object is in compass area) that is common for almost all sorts of game
- Minimap without Render Texture
- Customizable viewing options
- Multiple compass support
- Easy to use - just drag and drop prefabs to set it up


You can also use the features and do the operations of compass programmatically, such as adding indicators or removing them etc. by calling some methods only.

## What's new:

- New Layer Mask will allow you to show objects of specific layers on the compass
- Added an easy way to change the scale of indicators on the compass
- Fixed some minor issues

## How to use

1. Since the compass works with Unity UI only, you must create a Canvas if there isn't any. Right-click on the Hierarchy or go to **UI > Canvas**.
2. Go to **ProCOMPASS > Prefabs** in the Project tab and select one of those prefabs and then drag and drop that in the Canvas gameObject in Hierarchy tab. Modify the position and scale of Rect Transform to set it as you wish.

3. There should be a gameObject according to which the Compass will work. So, assign the player or main controller of the game in the **Player** field of the compass component. The basic compass is implemented.
4. To add indicators, select the gameObjects in the scene you want to indicate on the compass. ProCompass uses the **Tracker** component to indicate a gameObject. So click **Add Component**, type **Tracker** in the search box and then click on **Tracker**.
5. You will have to assign an indicator gameObject which will be indicating the position of this gameObject on the Compass. You will find some specific indicator prefabs in **ProCOMPASS>Prefabs>Indicators**. Assign one of them in the Indicator field of the **Tracker** component you added to the gameObject before. 
6. Select one of the activities depending on how the indicator of this gameObject will appear on some specific event. See [\[Tracker\]](#) for more details about these activities and other parameters of Tracker.
7. Go to Compass in the Canvas and then assign this gameObject in the **Trackers** list of the Compass component. Similarly, add as many Trackers as you need.
8. The compass with indicator has been implemented. Now it is time to customize it. You can change the parameters of the Compass and see how it works. See [\[Pro Compass\]](#) for more details about these parameters.
9. Click on the Play button of the editor to see if it works properly.
10. To set up a Minimap in the Compass, assign necessary objects to the parameters which you will find in the Minimap section of the Compass component. See [\[Minimap\]](#) for more details about these parameters.

## Public Methods

To add or remove indicator through scripts, follow these simple steps. Directly adding or removing trackers from the Trackers list of the compass may create inconsistency or throw exceptions in the runtime. Instead, there are methods by which you will be able to add or remove indicator of a specific gameObject easily. You can use **AddIndicator()** to add and **RemoveIndicator()** to remove. Both method takes the Tracker, which you want to add or remove, as parameter. However, destroying a gameObject which contains Tracker and was added to Compass, without removing it would do the similar job as you do by calling **RemoveIndicator()** method. So, there would be no problem doing it, but it will show a warning message in the Console. You can disable this by commenting out a line in the Compass script.

To add an indicator, create a reference of the compass and call **AddIndicator()** method from it. Simply pass the Tracker of the gameObject to the method.

```

1 using UnityEngine;
2 public class AddIndicatorTest : MonoBehaviour {
3     public Compass compass;
4     private Tracker tracker;
5     void Start () {
6         tracker = GetComponent<Tracker> ();
7         compass.AddIndicator (tracker);
8     }
9 }

```

To remove an indicator, just call **RemoveIndicator()** but don't try to remove a Tracker which hasn't been added yet. Otherwise, it will show error message in the console.

```

1  using UnityEngine;
2  public class RemoveIndicatorTest : MonoBehaviour {
3      public Compass compass;
4      public Tracker tracker;
5      void Start () {
6          // Do other initializations and things
7          tracker = GetComponent<Tracker> ();
8          compass.AddIndicator (tracker);
9      }
10     // From somewhere, let's assume Death() method is called
11     void Death () {
12         compass.RemoveIndicator (tracker);
13     }
14 }

```

To get the viewing angle, call **DirectionAngle()** method which will return a float value.

```

1      public Compass compass;
2      void Update () {
3          Debug.Log (compass.DirectionAngle ());
4      }

```

## Pro Compass

The main script that does almost all the works is **ProCompass.cs**. You will find the tooltip by placing the mouse pointer on the field in Unity Editor.

**Player:** Player of the compass or the gameObject under which the compass will work. Assign the main player like FPSController etc. of your game to this field.

**Trackers:** List of trackers that will indicate their position in the compass.

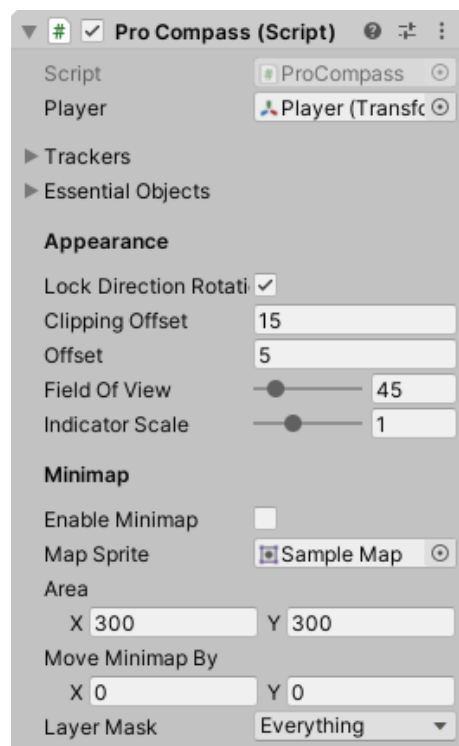
**Essential Objects:** Essential gameObjects in compass gameObject. Don't modify them or assign them properly.

### Appearance

**Lock Direction Rotation:** Lock rotation of the gameObjects in 'Directions' gameObject.

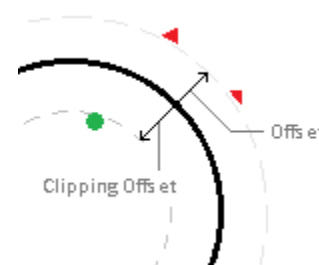
**Clipping Offset:** Offset of the indicators that are being clipped inside the compass.

**Offset:** Offset of the indicators outside compass. These indicators which are set to show direction outside compass will be clipped with offset.



**Field Of View:** Field of view of the compass can be called zoom in-out option. This defines the viewing area of compass.

**Indicator Scale:** Allows to set the scale of indicators on compass. This actually multiplies the scaling factor with the size of real prefab used in trackers. Set it to 1 to use the actual prefab size. The size can't be modified in play mode to maintain simplicity.



## Minimap

**Enable Minimap:** Turn on/off minimap.

**Map Sprite:** Sprite of the minimap. You must import the map texture as **Sprite (2D and UI)** as it will be used as a UI element.

**Area:** The area that the map will cover in the scene. Shouldn't be set to zero.

**Move Minimap By:** Calibrate or move the position of the minimap on the compass with respect to the player.

**Layer Mask:** You can choose to show indicators of gameObjects of specific layers on compass. For example, you can see normal enemies on your compass in normal mode. But in night vision or special mode you can see hidden enemies. You can implement it with the layer mask feature.

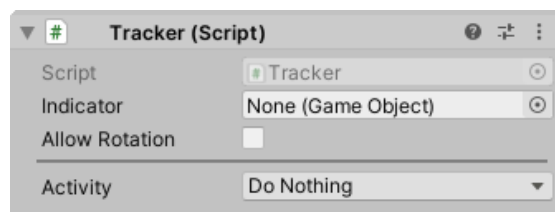
**Editor Helper:** While implementing **ProCompass** script in the inspector, there appears an editor helper if any problem detected. This feature doesn't affect the value of any variables. For example, if the value of an important variable of the compass is not set, there appears a button (**Show error and warnings**) at the end of the **ProCompass** component, suggesting you what to do. You can also hide the help box by clicking the button named '**Hide Help**' under the help box. All these things automatically disappears if no problems are found.

## Tracker

The **Tracker** identifies a gameObject and helps **ProCompass** to show its position and rotation on the compass.

**Indicator:** Basically, it is a gameObject that will be shown as an indicator on the compass. Simply put an indicator prefab on it.

**Allow Rotation:** Allowing rotation means the indicator on compass will rotate as the gameObject rotates. Disabling this will make the indicator stay fixed, no matter how the compass rotates.



**Activity:** There are three types of action you can perform with the indication, on a specific event.

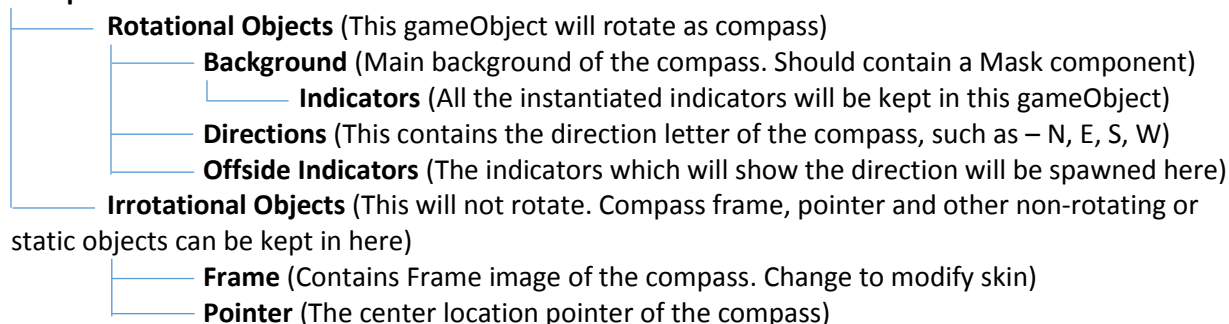
1. **Do Nothing:** This will show the indicator only if the gameObject appears in the field of view of the compass.
2. **Clip Indicator:** This will clip the indicator inside compass if the gameObject goes outside the field of view.

3. **Show Direction When Outside:** When a gameObject goes outside the compass, an alternative indicator is shown outside to indicate its direction. You must set a direction indicating gameObject as direction indicator.

## Hierarchy Structure and Customizations

The compass maintains a basic hierarchy structure for less design complexity. You can modify the appearance of the compass easily with this. Just replace your designed skin sprites with the one used in it. But you shouldn't change the structure, as changing may cause inconsistency. Some of these gameObjects are assigned to 'Essential Objects' section of ProCompass component.

### Compass



## Contact and Support

For any kind of suggestions, queries and help, you can contact me through email. If you like this asset, don't forget to write a review on the Asset Store, so that more amazing creators like you find this asset helpful for their projects.

**Arnab Raha**

Email: [arnabraha501@gmail.com](mailto:arnabraha501@gmail.com)

Thank you for choosing ProCOMPASS.