

HUD Navigation System

HUD Navigation System



Unity Asset Store Link:

<https://assetstore.unity.com/packages/slug/103056>

Documentation:

<http://docs.sickscore.games/hud-navigation-system/>

Online Demo:

<http://demo.sickscore.games/hud-navigation-system/>

Youtube Channel:

https://www.youtube.com/channel/UCep_wCCGKovzb7pKZ2nS-TA

Getting Started

Installation



Import Asset

Import HNS into your existing project or start a new one. If you're updating from a previous version, it might be necessary to delete the old folder before you import the package.

You'll find all package contents within the folder *Sickscore Games > HUD Navigation System*.



Quick Setup



The built-in *Quick Setup* is the most simple way to get started with HNS.

You'll find the *Quick Setup* in the Unity main menu:

Window > Sickscore Games > HUD Navigation System > Quick Setup Window

Simply assign your Player Controller (the Transform, which represents your player) and your Player Camera (in most cases, this is the main camera). The Quick Setup will do the following tasks automatically for you:

1. Adds mandatory scripts to your player controller and camera
2. Creates a new GameObject with the *HUD Navigation System* component
3. Adds the *HUD Navigation Canvas* prefab to your scene



QUICK SETUP

Player Controller

ExamplePlayer

Player Camera

Main Camera (Camera)

START QUICK SETUP

Example Content



Example Content

You'll find all example contents within the folder *Sickscore Games > HUD Navigation System > _Examples*. Simply copy e.g. HUD prefabs and adjust things as you like.



Examples are a good starting point for customization.

- **Example Scenes:** It's always good to play around in our example scenes before you integrate HNS into your own project.
- **Example Settings:** Some *Element Settings* used in our example scenes, which are shared between multiple *HUD Navigation Elements*.
- **HUD Prefabs:** Here you'll find the example prefabs for the different features of HNS. These are used in our example scenes. It's a good practice to copy them and adjust them to your needs.
- **Scene Configs:** Used in our example scenes. You can always create your own *Scene Configurations* to adjust settings between different scenes.
- **Scripts:** Some example scripts used in our example scenes. It might be worth checking some of them for reference. You can find more example scripts within *_Examples > Example Scene > Scripts*.
- **Textures:** Some example textures used in our example scenes. You're allowed to use all textures and icons for your own projects aswell!

Support / Feedback



E-Mail Support

Just **drop us a message** with your **Invoice #**, a brief description of your issue and maybe some **screenshots and/or videos** as reference.



We receive a large amount of support mails, so please give us time to check your request and write back to you 😊



Support Forum

We're are currently working on our new website, which will also include a support forum and threads for all our assets. Stay tuned!



Feature Requests

Having thoughts on how to improve HNS even further? **Drop us a message** with a brief description of your idea and we'll check, if it fits into the system.

FAQ's

Can I use HNS for my VR/XR project?

Currently VR/XR is not supported, because the HNS canvas runs in screen space. We're already working on an addon for HNS, which already supports most of the features of HNS. A working beta will be released to the store once we've integrated all features of the main asset. Stay tuned!

Is this asset also suitable for 2D / 2.5D games?

2D is not supported "out-of-the-box", but we have a few customers, which successfully use HNS in their 2D/2.5D games. This mostly depends on the individual setup of your project. Please [contact us](#) for further advice, before you purchase HNS.

Can I get a free voucher for your assets?

It should be self-explanatory, but we can't provide everyone with free copies of our addons :) We have only a handful of free vouchers available each year, so we exclusively give them out to students (with legal documents / proof of study).

How to use HNS in an environment with multiple scenes?

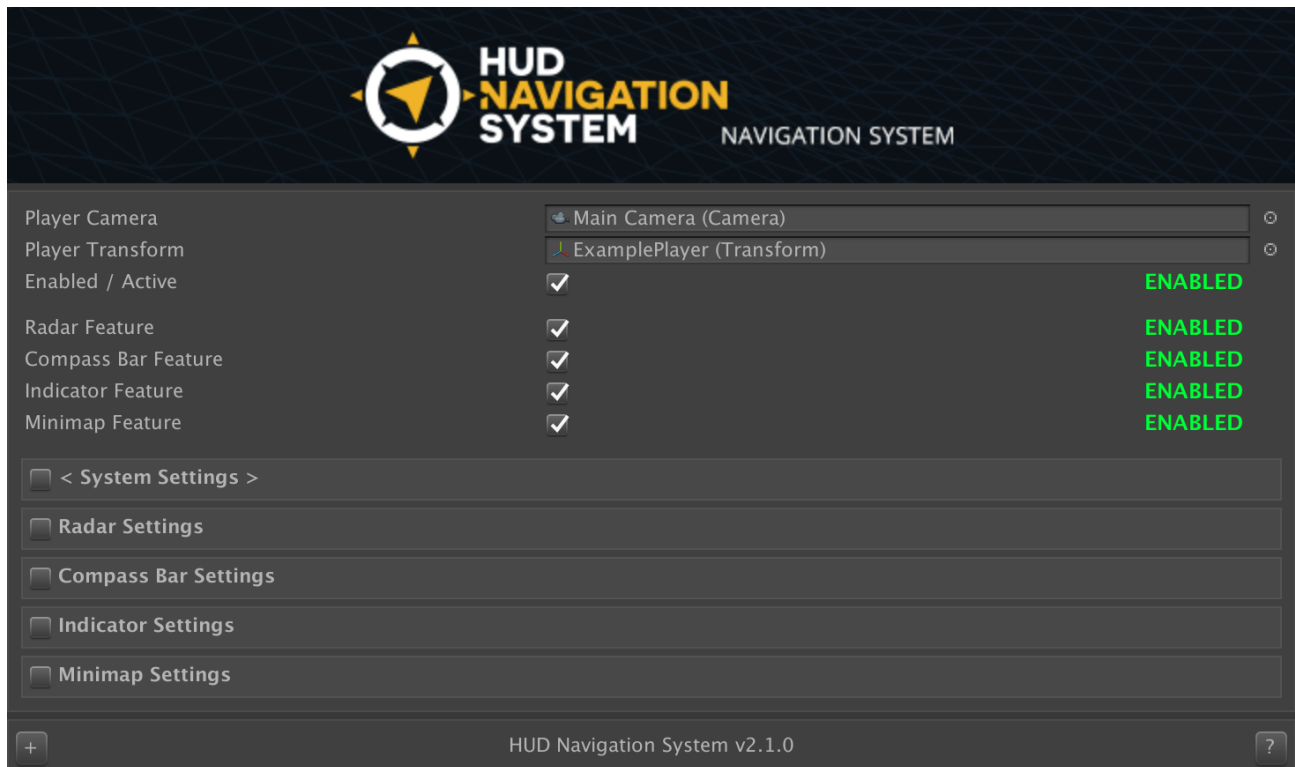
That's actually really easy to accomplish with the [Scene Manager](#). Just make sure you've enabled the *Don't Destroy On Load* [setting](#) on the *HUD Navigation System* and let the scene manager do the rest. For e.g. menu scenes, where you don't need HNS, simply disable the system on these scenes with the *Disabled In Scene* option.

HUD Navigation System

Component Info

HUD Navigation System


This is the main component of HNS, which is used to enable/disable the different features of HNS and holds most of the global settings.



Settings

Settings

The *HUD Navigation System* component has a ton of settings, which offers a variety of possibilities for your project.

 We provide a tooltip for almost every setting, just hover them with your mouse pointer in the unity editor.

General Settings

- **Player Transform:** Assign your Player Controller (the Transform, which represents your player). You can also add the *HNSPlayerController* component to your player, to dynamically assign this setting.
- **Player Camera:** Assign your Player Camera (in most cases, this is your main camera). You can also add the *HNSPlayerCamera* component to your player camera, to dynamically assign this setting.
- **Enabled / Active:** If enabled, the system will start automatically. This setting can be changed at runtime to e.g. temporarily disable HNS.

System Settings

- **Rotation Reference:** HNS will use the rotation of this transform for it's calculations.
- **Update Mode:** Depending on your project, you might have to change this setting, if the icon positioning is not synced with your character movement.
- **Don't Destroy On Load:** By default, HNS will automatically persist between scene changes. You can globally disable this behavior with this setting. *Note: You then have to manually include HNS again, if you change scenes!*

Radar Settings

- **Radar Mode:** You can switch the mode to change the behavior of the radar. Select the mode, which fits best to your project.
- **Radar Zoom:** Controls the overall zoom of the radar. The higher the value, the higher the virtual drone floats above the player. :)
- **Radar Radius:** Radius of the radar in meters/units. Elements outside this radius will be displayed on the border of the radar.
- **Radar Radius (Border):** Border radius in meters/units. Elements outside this radius will be hidden on the radar.
- **Enable Height System:** Enable the height based system for each element inside the radar. If an element is physically above or below a certain distance, it will show an arrow above/below the element inside the radar.
 - **Min. Distance Above:** Minimum distance upwards to activate the element's ABOVE arrow.
 - **Min. Distance Below:** Minimum distance downwards to activate the element's BELOW arrow.
 - **Show Debug Gizmos:** If enabled, it will draw two planes above and below the player, which indicate the distance of the height system above and below the player.
 - **Gizmo Size:** Defines the size of the Debug Gizmos.
 - **Gizmo Color:** Defines the color and opacity of the Debug Gizmos.

Compass Bar Settings

- **Compass Bar Radius:** Radius of the compass bar in meters/units. Elements outside this radius will be hidden on the compass bar.

Indicator Settings

- **Indicator Radius:** Radius of the indicators in meters/units. Indicators of elements outside this radius will be hidden.
- **Indicator Hide Distance:** Controls the distance in meters/units at where indicators are automatically hidden, if your player comes closer to the element. Set the value to 0 to disable auto-hide on your indicators.
- **Enable Offscreen Indicators:** If enabled, indicators will stick to the borders of the screen, if they're currently not visible on screen.

- **Screen Border:** Controls the distance of your offscreen indicators to the screen borders. A higher value increases the distance to the border.
- **Enable Distance Scaling:** If enabled, the indicators will scale by distance within the defined scale radius.
 - **Scale Radius:** Indicators inside this radius will be scaled by distance. Value must be lower or equal to the actual indicator radius.
 - **Minimum Scale:** Controls the minimum scale of indicators affected by the distance scaling functionality.
- **Enable Distance Fading:** If enabled, the indicators will fade by distance within the defined fade radius.
 - **Fade Radius:** Indicators inside this radius will be faded by distance. Value must be lower or equal to the actual indicator radius.
 - **Minimum Opacity:** Controls the minimum opacity of indicators affected by the distance fading functionality. Set the value to 0 to completely fade-out indicators.

Minimap Settings

- **Minimap Profile:** Assign the map profile you want to use for the minimap. For more informations on how to generate a map profile, follow the instructions in the [map profile documentation](#).
- **Minimap Mode:** You can switch the mode to change the behavior of the minimap. Select the mode, which fits best to your project.
- **Minimap Scale:** Controls the overall scale of the minimap. Decrease the value to zoom out the minimap and show more area around the player.
- **Minimap Radius:** Radius of the minimap in meters/units. Elements outside this radius will be displayed on the border of the minimap.
- **Show Minimap Bounds:** If enabled, it will draw a cube which scales to the size of your minimap bounds. Can be useful to debug your minimap profiles.
 - **Gizmo Color:** Defines the color and opacity of the minimap bounds cube.
- **Enable Height System:** Enable the height based system for each element inside the minimap. If an element is physically above or below a certain distance, it will show an arrow above/below the element inside the minimap.
 - **Min. Distance Above:** Minimum distance upwards to activate the element's ABOVE arrow.
 - **Min. Distance Below:** Minimum distance downwards to activate the element's BELOW arrow.

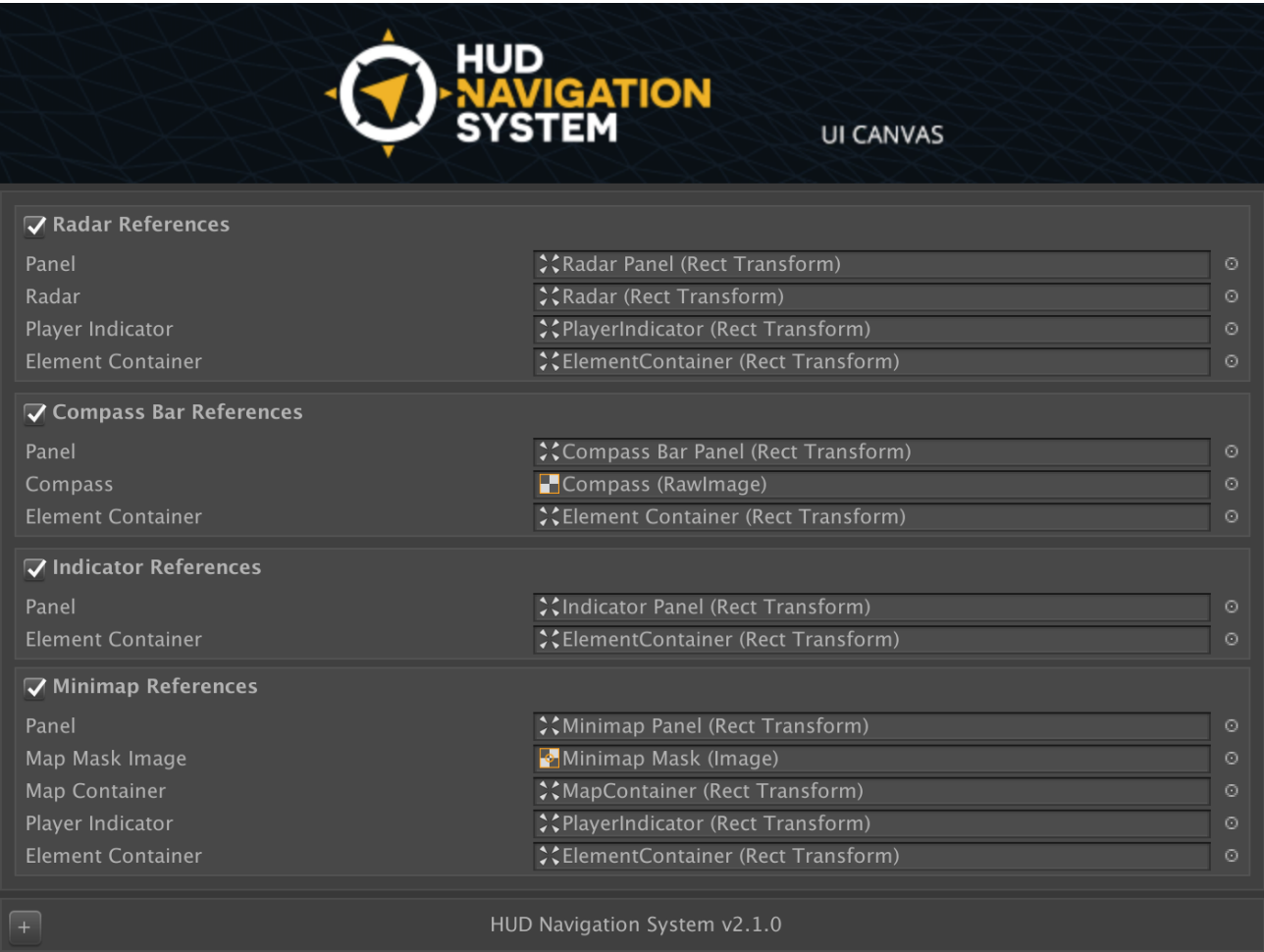
- **Show Debug Gizmos:** If enabled, it will draw two planes above and below the player, which indicate the distance of the height system above and below the player.
 - **Gizmo Size:** Defines the size of the Debug Gizmos.
 - **Gizmo Color:** Defines the color and opacity of the Debug Gizmos.

HUD Navigation Canvas

Component Info

HUD Navigation Canvas

This is the 2D canvas prefab, which holds the UI elements for each feature of HNS. This prefab can simply be copied and customized to fit your needs.




Settings / UI References

Settings

The *HUD Navigation Canvas* is controlled by the settings of the **HUD Navigation System** component. This component is used to link/assign the different *UI References* to the system.

UI References

HNS needs to know, where to find the different UI elements inside your UI Canvas. Each feature needs specific UI elements to work properly, so make sure to correctly assign them.

 The most simply way to get started is to duplicate the canvas prefab and customize it to your needs. You'll find the prefab in the prefabs folder:
Sickscore Games > HUD Navigation System > Resources > Prefabs.

Radar References

- **Panel:** The parent transform, which holds all UI components for this feature.
- **Radar:** The actual radar transform, which has the image of your radar.
- **Player Indicator:** A transform, which represents the icon/location of your player.
- **Element Container:** The container transform, which holds all instantiated elements.

Compass Bar References

- **Panel:** The parent transform, which holds all UI components for this feature.
- **Compass:** The actual compass transform, which has the image of your compass.
- **Element Container:** The container transform, which holds all instantiated elements.

Indicator References

- **Panel:** The parent transform, which holds all UI components for this feature.
- **Element Container:** The container transform, which holds all instantiated elements.

Minimap References

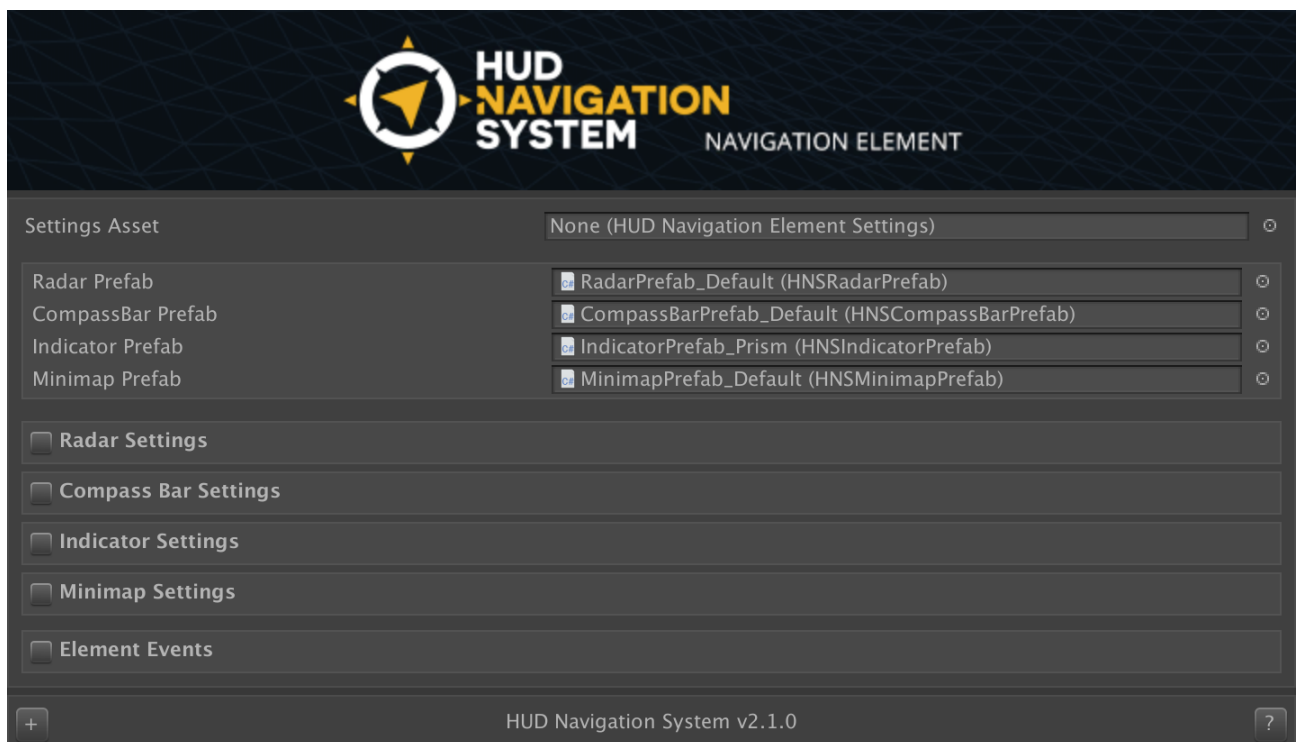
- **Panel:** The parent transform, which holds all UI components for this feature.
- **Map Mask Image:** The parent transform of the map container, which has an image mask component attached. The image mask will clip the minimap and hides the borders of the map texture.
- **Map Container:** The container transform for all instantiated map layers.
- **Player Indicator:** A transform, which represents the icon/location of your player.
- **Element Container:** The container transform, which holds all instantiated elements.

HUD Navigation Element

Component Info

HUD Navigation Element


Assign this component to each element you want to track with HNS. *HUD Navigation Elements* are designed to be instantiated at runtime, so you can add/remove new elements at any time. Once a GameObject with this component attached gets removed from the scene, HNS will automatically get notified and will remove the element from the collection.



Settings

Settings

Most of the settings on the *HUD Navigation Element* component are used to override the default settings defined by the **HUD Navigation System** component. This way, you can define different types of elements, which derive from the default settings, but behave different from each other.

 If you plan to create a lot of elements of the same type (e.g. item pick-ups), it's a good practice to use **HUD Navigation Element Settings** to share your settings between multiple elements and change them at a single place.

General Settings

- **Settings Asset:** (*optional*) Assign an **Element Settings** asset to configure this element based on the settings of the asset.
- **Radar / Compass Bar / Indicator / Minimap Prefab:** Assign an **UI Prefab** for each feature you want to use on this element.

Radar Settings

- **Hide In Radar:** If enabled, the element will be hidden within the radar.
- **Ignore Radius:** If enabled, the element will always be visible within the radar, even if it's outside the defined radius. Useful for e.g. quest markers!
- **Rotate With GameObject:** If enabled, the element will rotate within the radar, depending on it's GameObject's rotation.
- **Enable Height System:** If enabled, the radar height system will be activated on this element.

Compass Bar Settings

- **Hide In Compass Bar:** If enabled, the element will be hidden within the compass bar.

- **Ignore Radius:** If enabled, the element will always be visible within the compass bar, even if it's outside the defined radius. Useful for e.g. quest markers!
- **Enable Distance Text:** If enabled, the distance text will be activated on this element. It will show the current distance to the player next to the element.
 - **Text Format:** Defines the format of the distance text. "{0}" will be automatically replaced with the actual distance value.

Indicator Settings

- **Show Indicator:** If enabled, the indicator for this element will be visible. Make sure, the indicator feature is enabled on the **HUD Navigation System** component.
- **Show Offscreen Indicator:** If enabled, the offscreen indicator for this element will be visible.
- **Ignore Radius:** If enabled, the indicator for this element will always be visible, even if it's outside the defined radius. Useful for e.g. quest markers!
- **Ignore Hide Distance:** If enabled, the indicator for this element will ignore the defined hide distance. Useful for e.g. item pick-ups!
- **Ignore Distance Scaling:** If enabled, the indicator for this element will not be affected by the distance scaling setting.
- **Ignore Distance Fading:** If enabled, the indicator for this element will not be affected by the distance fading setting.
- **Enable Distance Text:** If enabled, the distance text will be activated on this indicator. It will show the current distance to the player next to the indicator.
 - **Show Offscreen Distance:** If enabled, it will also show the distance text on the offscreen indicator.
 - **Onscreen / Offscreen Text Format:** Defines the format of the onscreen / offscreen distance text. "{0}" will automatically be replaced with the actual distance value.

Minimap Settings

- **Hide In Minimap:** If enabled, the element will be hidden within the minimap.
- **Ignore Radius:** If enabled, the element will always be visible within the minimap, even if it's outside the defined radius. Useful for e.g. quest markers!
- **Rotate With GameObject:** If enabled, the element will rotate within the minimap, depending on it's GameObject's rotation.

- **Enable Height System:** If enabled, the minimap height system will be activated on this element.

Element Settings



Element Settings

If you plan to create a lot of elements of the same type (e.g. item pick-ups), it's a good practice to use *Element Settings* to share your settings between multiple elements and change them at a single place.

The settings of the *Element Settings* asset are identical to the settings of the *HUD Navigation Element* component. You can find the [documentation here](#).



To create new *Element Settings*, right click in your project window and select:
Create > Sickness Games > HUD Navigation System > New Element Settings



Copy Settings

It's easy to create an *Element Settings* asset from an existing *HUD Navigation Element*. Simply create a new *Element Settings* asset and assign the GameObject with the existing *HUD Navigation Element* component to the *Copy From* field and click on *Copy Settings*. All existing settings on the *Element Settings* asset will be replaced!



HUD NAVIGATION SYSTEM

ELEMENT SETTINGS

Copy From

HUD Element Pivot

Copy Settings

Radar Prefab

None (HNS Radar Prefab)

CompassBar Prefab

None (HNS Compass Bar Prefab)

Indicator Prefab

IndicatorPrefab_PickUpItem (HNSIndicatorPrefab)

Minimap Prefab

None (HNS Minimap Prefab)

☒ Indicator Settings

Show Indicator



Show Offscreen Indicator



Ignore Radius



Ignore Hide Distance



Ignore Distance Scaling



Ignore Distance Fading



☒ Enable Distance Text

Show Offscreen Distance



Onscreen Text Format

{0}m




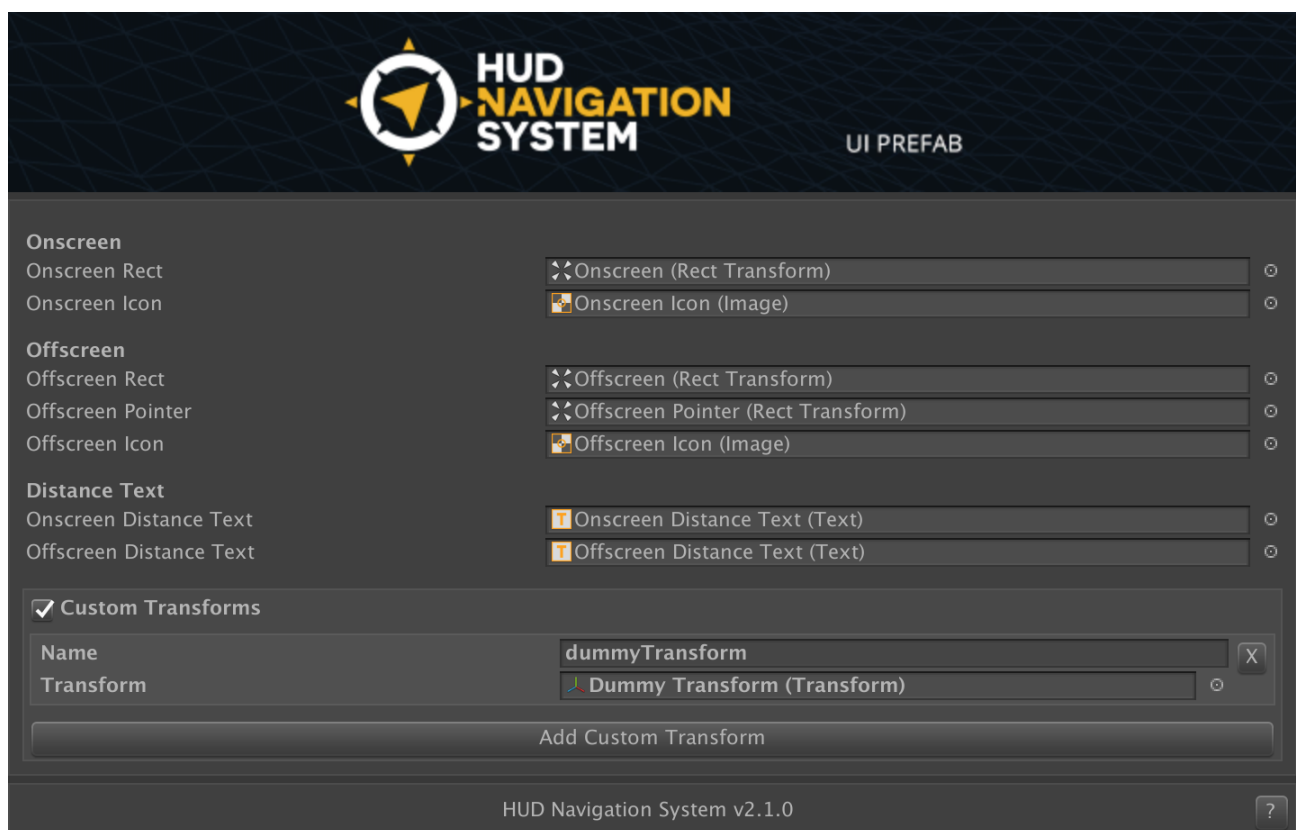
HUD Navigation System v2.1.0



UI Prefabs

HNS needs to know, where to find the different UI elements on your marker. Each feature needs specific UI elements to work properly, so make sure to correctly assign them.

-  The most simply way to get started is to duplicate any UI prefab and customize it to your needs. You'll find the prefab in the prefabs folder:
Sickscore Games > HUD Navigation System > _Examples > HUDPrefabs.



HNS Radar Prefab

- **Icon:** The main icon for this element
- **Arrow Above:** The icon used as the upwards arrow for the height system
- **Arrow Below:** The icon used as the downwards arrow for the height system

HNS Compass Bar Prefab

- **Icon:** The main icon for this element
 - **Distance Text:** The text component used for the distance text.
-

HNS Indicator Prefab

- **Onscreen Rect:** The parent transform of the onscreen indicator
 - **Onscreen Icon:** The main onscreen icon for this element.
 - **Onscreen Distance Text:** The text component used for the onscreen distance text.
 - **Offscreen Rect:** The parent transform of the offscreen indicator
 - **Offscreen Pointer:** The pointer image used for the offscreen indicator.
 - **Offscreen Icon:** The main offscreen icon for this element.
 - **Offscreen Distance Text:** The text component used for the offscreen distance text.
-

HNS Minimap Prefab

- **Icon:** The main icon for this element
 - **Arrow Above:** The icon used as the upwards arrow for the height system
 - **Arrow Below:** The icon used as the downwards arrow for the height system
-

Custom Transforms

Custom Transforms are an easy way to add custom logic to *HUD Navigation Elements*. We use this mechanic in our example scenes to create the item pick-up.



Custom Transforms can be used to include custom logic to *HUD Navigation Elements*, such as item pick-ups, interactions, ...

Settings

- **Name:** Enter a unique name, which will be used to access this transform in code
- **Transform:** Assign the transform with your custom logic

Code Example

Show/hide a custom icon on the indicator, if an element is within the defined radius. How to get this example to work:

1. Add a custom transform to your indicator *UI Prefab*
2. Your *Custom Transform* must be named "*customIcon*" (see code)
3. Create a new C# script and copy&paste the code below
4. Select the *HUD Navigation Element* or **Element Setting** you want to use
5. Assign these methods to their corresponding **Events**:

OnEnterRadius: OnIndicatorEnterRadius()

OnLeaveRadius: OnIndicatorLeaveRadius()

```
1  using UnityEngine;
2  using SickscoreGames.HUDNavigationSystem;
3
4  public class CustomTransformCallbackScript : MonoBehaviour
5  {
6      public void OnIndicatorEnterRadius (HUDNavigationElement element, Naviga
7      {
8          ShowCustomIndicatorTransform(element, type, true);
9      }
10
11
12     public void OnIndicatorLeaveRadius (HUDNavigationElement element, Naviga
13     {
14         ShowCustomIndicatorTransform(element, type, false);
15     }
16
17
18     void ShowCustomIndicatorTransform (HUDNavigationElement element, Navigat
19     {
20         // we're only interested in indicator events
21         if (!type.Equals(NavigationElementType.Indicator))
22             return;
23
24         // check if indicator exists
```

```
25     if (element.Indicator != null)
26     {
27         // show/hide our custom icon, if it exists
28         Transform customIcon = element.Indicator.GetCustomTransform("customI
29         if (customIcon != null)
30             customIcon.gameObject.SetActive(visible);
31     }
32 }
33 }
```

Events



Events

Events give you the ability to directly interact with HUD Navigation Elements when certain events happen during runtime. This way you could for example play an audio track, if an element (e.g. enemy) appears on screen.

```
1 public void OnElementReady(HUDNavigationElement element)
2 {
3     // Fires directly after the element got initialized and is ready.
4     // It's safe to interact with the element after this event has fired.
5 }
```

```
1 public void OnAppear(HUDNavigationElement element, NavigationElementType t
2 {
3     // Fires directly when the element appears on screen.
4     // This event will fire for each feature individually!
5     // Use the type parameter to check, which feature fired this event.
6 }
```

```
1 public void OnDisappear(HUDNavigationElement element, NavigationElementTyp
2 {
3     // Fires directly when the element disappears from the screen.
4     // This event will fire for each feature individually!
5     // Use the type parameter to check, which feature fired this event.
6 }
```

```
1 public void OnEnterRadius(HUDNavigationElement element, NavigationElementT
2 {
3     // Fires directly when the element enters the defined radius.
4     // This event will fire for each feature individually!
5     // Use the type parameter to check, which feature fired this event.
6 }
```

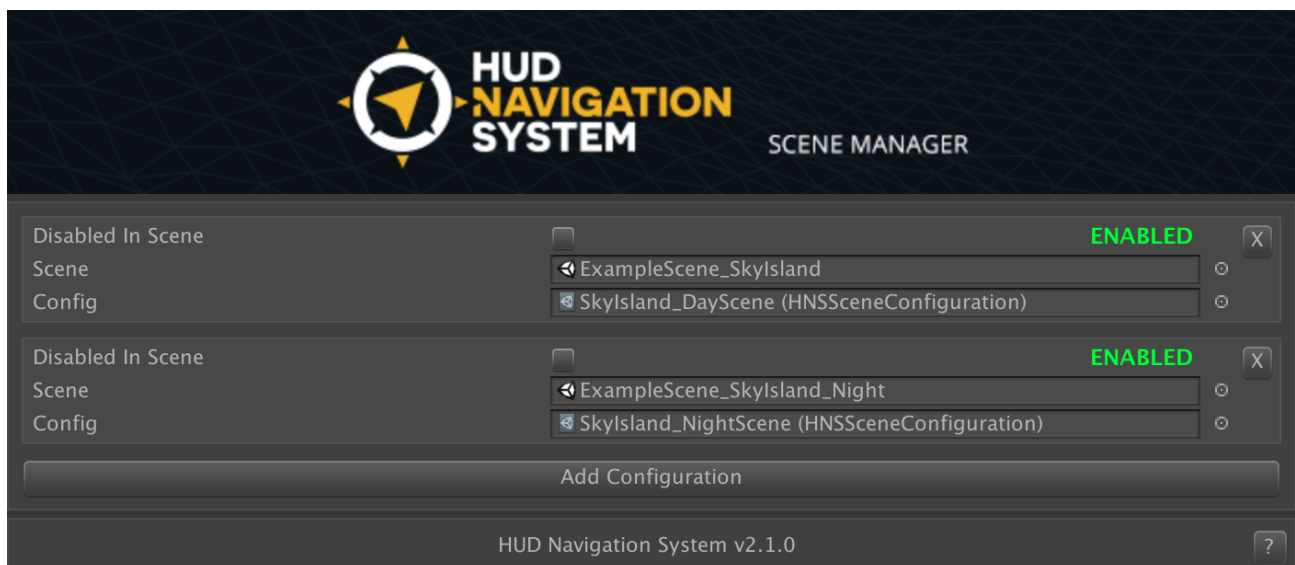
```
1 public void OnLeaveRadius(HUDNavigationElement element, NavigationElementT
2 {
3     // Fires directly when the element leaves the defined radius.
4     // This event will fire for each feature individually!
5     // Use the type parameter to check, which feature fired this event.
6 }
```

Scene Manager

Component Info

HUD Navigation Scene Manager

The *Scene Manager* is an essential component, if you're working within an environment with multiple scenes. With the *Scene Manager*, you can simply create **Scene Configurations** for your different scenes, or disable HNS completely on e.g. menu scenes.



Settings



Settings

- **Disabled In Scene:** If checked, HNS will be completely disabled within this scene. Useful for e.g. menu scenes.
- **Scene:** Assign the scene you want to configure.
- **Config:** Assign the [Scene Configuration](#) you want to use for this scene.




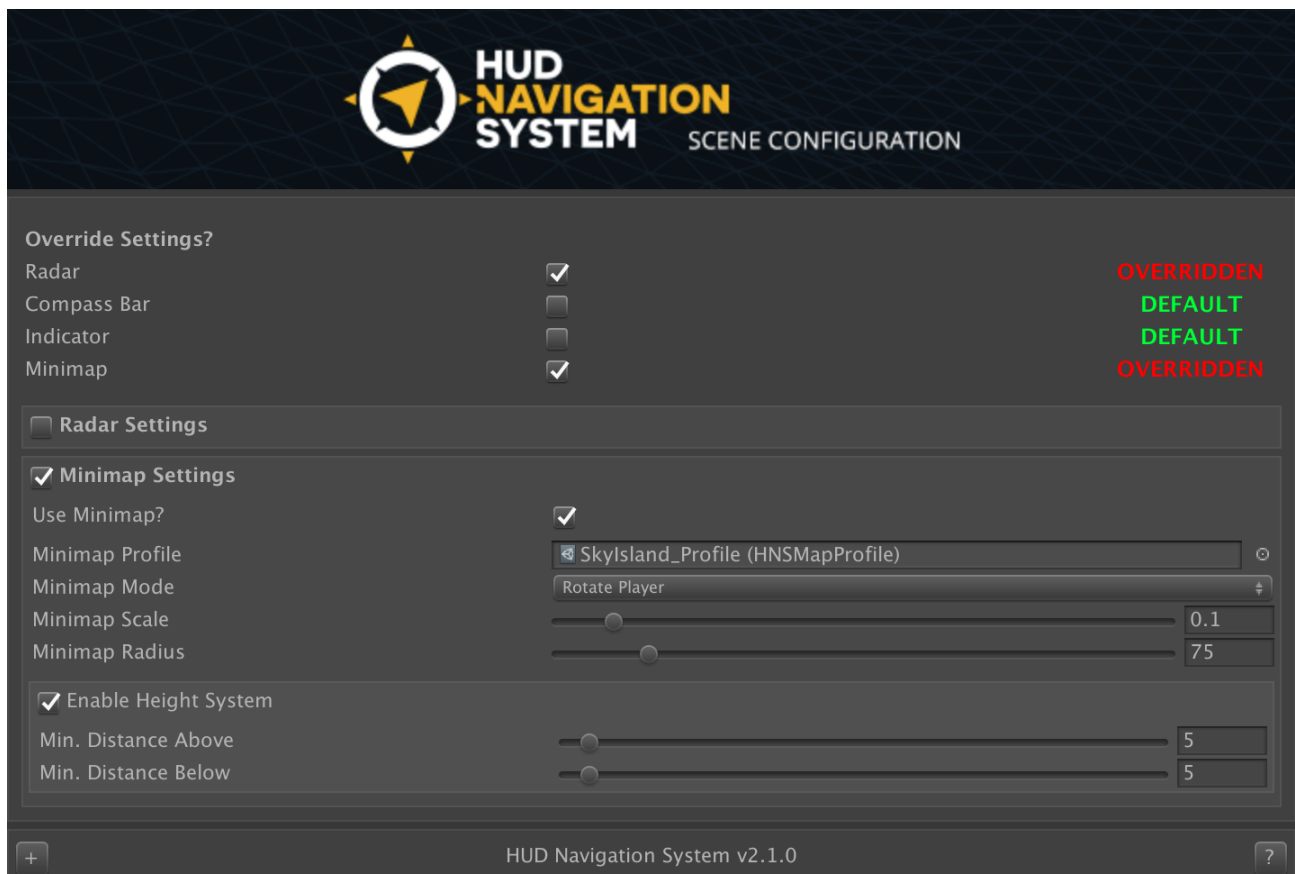
Take a look at the [Example Scenes](#) to see the *Scene Manager* in action!

Scene Configurations

Scene Configuration

Scene Configurations are used by the **Scene Manager** to apply different settings for each scene. With *Scene Configurations* you can override every setting of HNS and completely change the behavior for individual scenes.

-  To create a new *Scene Configuration*, right click in your project window and select:
- Create > Sickscore Games > HUD Navigation System > New Scene Configuration*

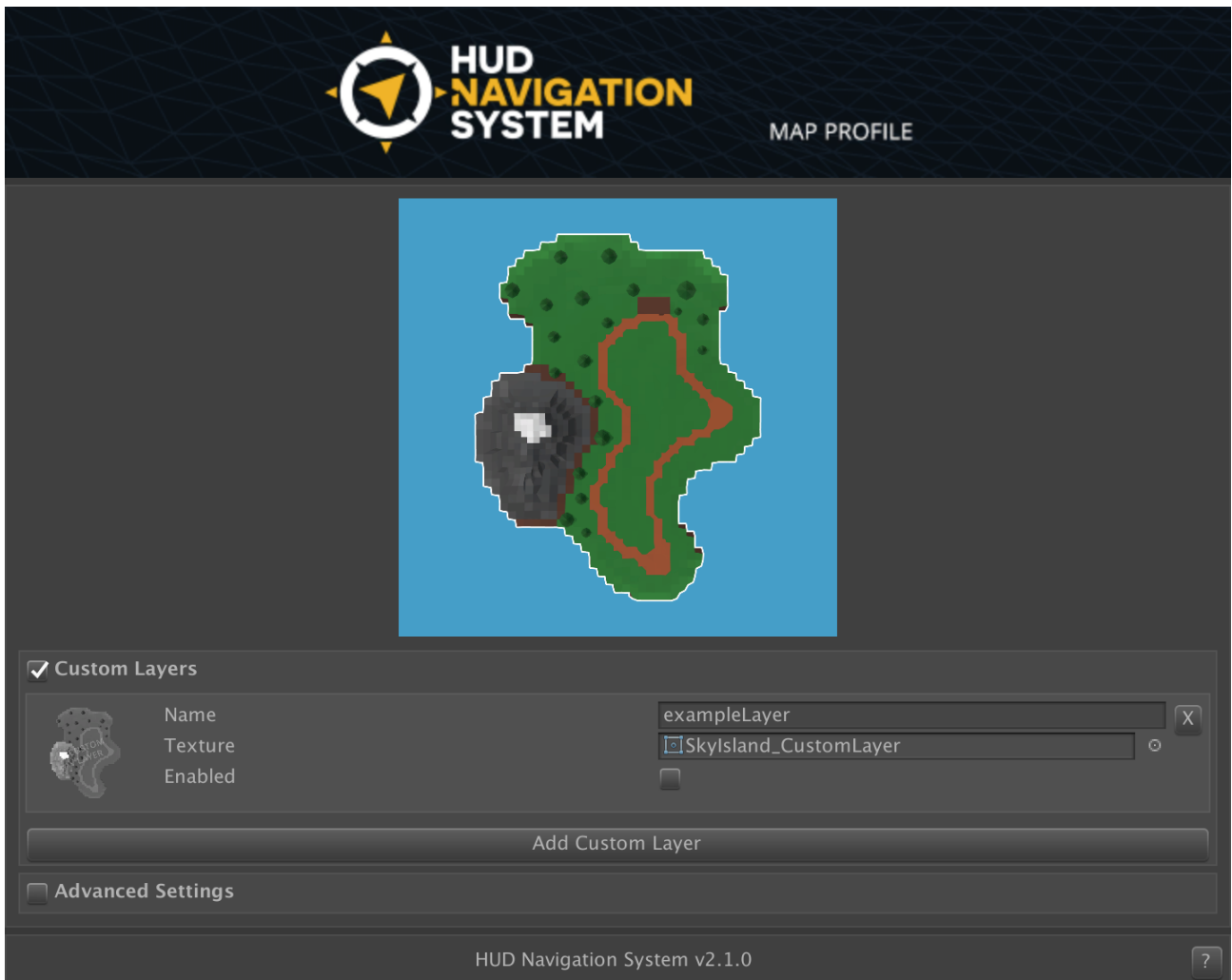


Map Profiles

Component Info

Map Profile

Map Profiles store the rendered (mini)map texture as well as it's corresponding map bounds. To create a new *Map Profile*, you have to use the [Map Texture Creator](#).



Custom Layers

You can add *Custom Layers* to any *Map Profile*, which you can access by code at runtime.



Custom Layers can be used to include custom logic to your (mini)map, such as day/night mode, cave systems, heat maps, etc...

Settings

- **Name:** Enter a unique name, which will be used to access this layer in code
- **Texture:** Assign your custom layer texture. Make sure it has the same dimensions as the map texture!
- **Enabled:** If checked, the custom layer is enabled and can be accessed by code.

Code Example


Toggle a custom layer's visibility by pressing the "F" key.

```
1  using UnityEngine;
2  using SickscoreGames.HUDNavigationSystem;
3
4  public class CustomLayerExample : MonoBehaviour
5  {
6      private HUDNavigationSystem _HUDNavigationSystem;
7
8      void Start ()
9      {
10         _HUDNavigationSystem = HUDNavigationSystem.Instance;
11     }
12
13
14     void Update ()
15     {
16         // check if the 'F' key was pressed
17         if (Input.GetKeyDown(KeyCode.F) && _HUDNavigationSystem.currentMinimap
18         {
19             // get custom layer from the current map profile
20             GameObject customLayer = _HUDNavigationSystem.currentMinimapProfile.
21             if (customLayer != null)
22                 customLayer.SetActive(!customLayer.activeSelf);
23         }
24     }
25 }
```

Create Profiles

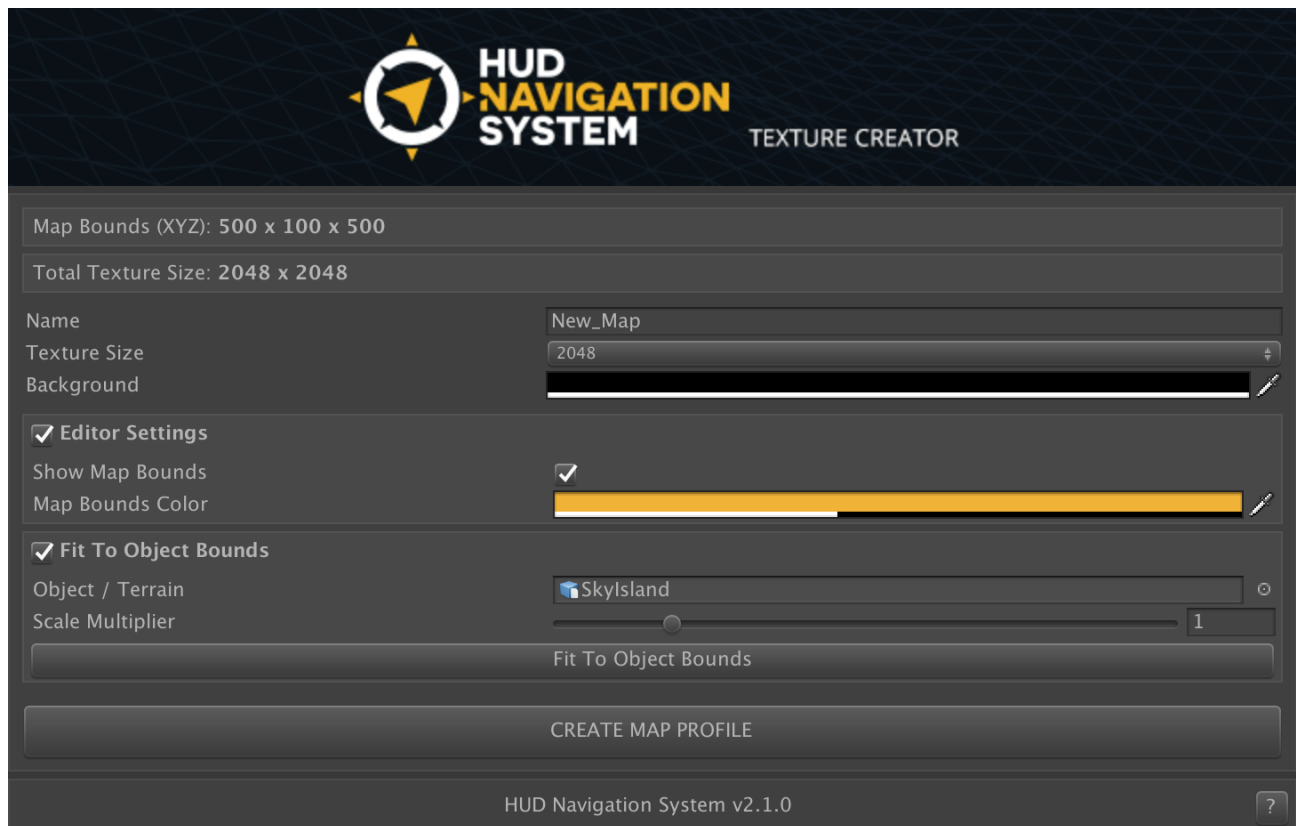
Map Texture Creator

The *Map Texture Creator* is used to render a static texture of your whole map and create a **Map Profile** out of it. The final texture can be edited in an image program to customize the look and feel of your (mini)map.

 Make sure to hide all elements, which should not appear on the final map texture, before you create the *Map Profile*.

You'll find the *Map Texture Creator* in the Unity main menu:

Window > Sickness Games > HUD Navigation System > Map Texture Creator





Usage

Once you open the *Map Texture Creator*, a new GameObject called '*HNS MapTextureCreator*' will be automatically added to the scene, which holds all necessary components.

The process of creating a new *Map Profile* is pretty straightforward:

1. Enter a name for your new MapProfile
2. Select a fitting texture size.
3. Adjust the bounding box of your map. Use the box handles to modify the size within the scene view.
4. (optional) Assign your terrain / level mesh to automatically adjust the size of the bounding box. With the *Scale Multiplier* you can manually fine-tune the size of the generated bounding box.

Once the *Map Profile* is created, it will be highlighted in the project window and is ready to be assigned to the **HUD Navigation System** component or to a **Scene Configuration**.