



ULTIMATE HUD MANAGER (UHM)

V1.0

DOCUMENTATION

--OVERVIEW--

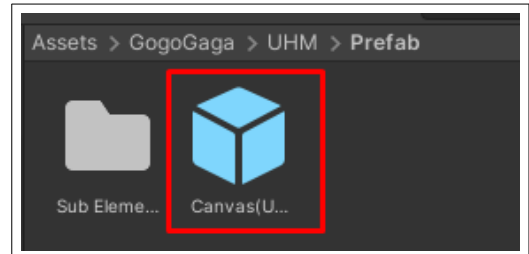
UHM is complete package for anything related to "HUD". Most games (especially First-Person and Third-Person games) needs HUD. UHM provide you with complete package to create basic and advance HUD, compact into simple and easy-to-use solution.

Indexing

- Basics: Page-3
- Mission Panel: Page-4
- Skill Bar: Page-7
- Primary Notification: Page-10
- Secondary Notification: Page-12
- Pickup Notification: Page-14
- Input Tool-tip: Page-15
 - Text Formatting: Page-16
- Damage/Heal Effect: Page-19
- Cross-hair: Page-20
- Health-bar: Page-23
- Fade Out/In: Page-25
- Way-point Marker: Page-26
 - Compass: Page-30
- Progress Tool-tip: Page-31

1. Basics:

Add **"Canvas(UHM)"** prefab to the scene. It is a singleton, so only one **instance** is required. Also it is a **"Don't Destroy On Load"**.



Use this **name-space** to access **"UHM"**:

```
using GogoGaga.UHM;
```

"Access" the singleton by writing:

```
UltimateHudManager.Instance
```

"Access" the singleton by

By **default** UHM will be using **main camera** (using the MainCamera tag, or what is returned by Camera.main), you can **change** that my.

```
SetMainCamera(Camera cam)
```

2. Mission Panel:

Creates a mission panel with a “**title**” and sub “**tasks**”.



Create/Show/Hide:

These functions will help you create the mission HUD.

```
CreateMission(string missionTitle)
```

```
CreateMission(string missionTitle, string[] tasks)
```

```
CreateMission(string missionTitle, TaskData[] taskDatas)
```

- **Mission Title:** What should be the title of the mission.
- **Tasks (string array):** It's an array of each task, by default each task is going to be in pending state.
- **Tasks (TaskData array):** It's an array of task data
 - **Task Text:** Name of the task.
 - **Task Progress Type:** Type of progress.

```
TaskData(string taskText  
         ,TaskProgressType progressType)
```

```
TaskProgressType  
{  
    pending,  
    failed,  
    skipped,  
    completed,  
}
```

This will hide the Mission Panel, and it will play a hiding animation as well. **clearMissionData** will tell if to remove all tasks when hiding.

```
HideMission(bool clearMissionData = false)
```

This will Show the mission panel if it is hidden.

```
ShowMissions()
```

Add/Remove Tasks:

If you want to Add/Remove mission after you have created mission:

```
AddMissionTask(string taskText)
```

```
AddMissionTask(string[] taskText)
```

```
AddMissionTask(TaskData taskData)
```

```
AddMissionTask(TaskData[] taskData)
```

Remove a task if you don't need it

```
RemoveMissionTask(int Index)
```

Index: The index of task (starts from 0, and the completed tasks are also included).

Change Task Status:

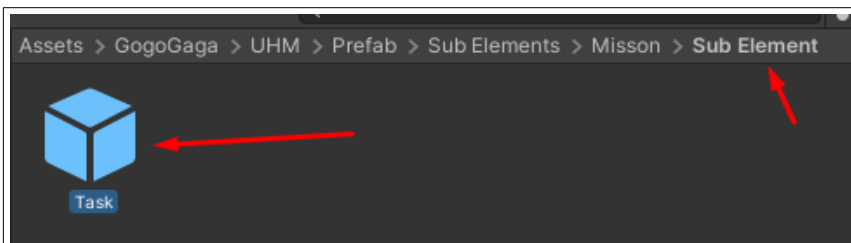
You can change task status to pending, failed, skipped, completed.

```
ChangeTaskStatus(TaskProgressType NewType,  
                 int Index = 0)
```

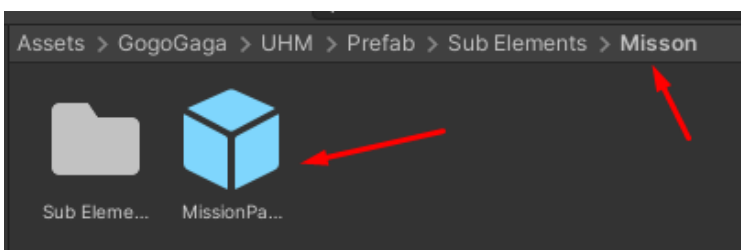
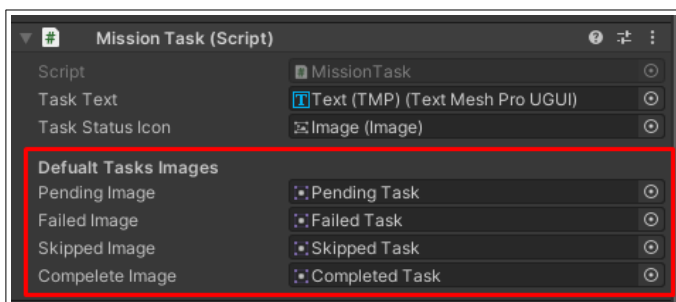
Note: On changing the task type to **“Completed”** will remove the tasks from the Mission Panel, but it will stay in the List and if you change it again to other task type for example skipped, It will reappear.

Other Properties:

If you want to change the Default Sprites for the pending/skipped/failed/completed tasks, here's how



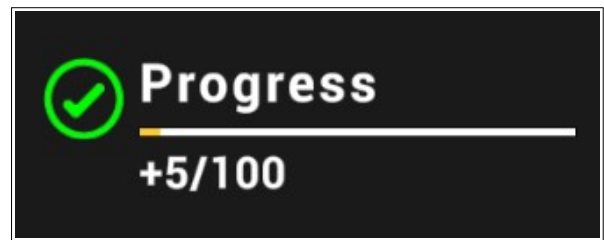
Find this Prefab and on **“Mission Task”** script you will find the default sprites field.



You will find the **Mission Panel** prefab here.

3. Skill Bar:

Creates a Progress bar with **Title**, **Image**, **Description**. You can use it for skill or Level progress.



Create/Show/Add/Remove:


These functions will help you create the Skill bar HUD

```
CreateSkillBar(SkillProgressProperties newSkillData,  
               bool ShowAfterCreating = false)
```

- **Show After Creating:** If true, it will show the progress bar right after creating.
- **New Skill Data:** The data structure with skill properties.
 - **Title:** The title of the progress bar, this should be unique always, this will be used to access this skill next time.
 - **Progress Value:** Upon creating this will be the starting value of the progress.
 - **Max Value:** This is the maximum value of progress.
 - **Description:** What should be the text below progress skill, if this is null/empty, by default the description will be " + progressValue / maxValue ".
 - **Icon Image:** This the Sprite shown right next to skill bar. If null, it will use the default Sprite.

```
SkillProgressProperties(  
    string title,  
    int progressValue,  
    int maxValue,  
    string description,  
    Sprite iconImage)
```

```
SkillProgressProperties(  
    string title,  
    int progressValue,  
    int maxValue,  
    string description,  
    Sprite iconImage)
```



```
SkillProgressProperties(  
    string title,  
    int progressValue,  
    int maxValue,  
    string description)
```

```
SkillProgressProperties(  
    string title,  
    int progressValue,  
    int maxValue)
```

This is how you will **add** the Progress to progressValue.

```
AddValueSkillProgress(string title, int add)
```

This is how you will **show** the Progress bar, use **newVal** if you want to update the skill progress.

```
ShowSkillProgress(string title, int newVal = -1)
```

If you want to **update** multiple properties in **SkillProgressProperties**.

```
UpdateSkillProgressData(SkillProgressProperties newData,  
                        bool ShowAfterUpdating = true)
```

If you want to **remove** any progress property.

```
RemoveSkillProgress(string Title)
```

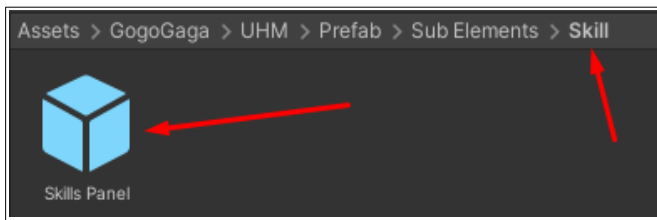
If you want to **remove all** progress properties.

```
RemoveAllSkillProgress()
```

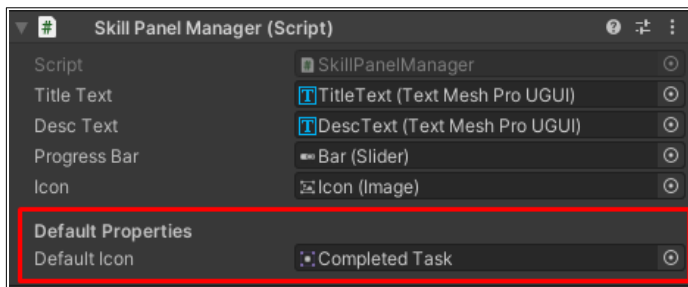
Other Properties:

This will **return progress value**.

```
GetSkillProgressValue(string title)
```

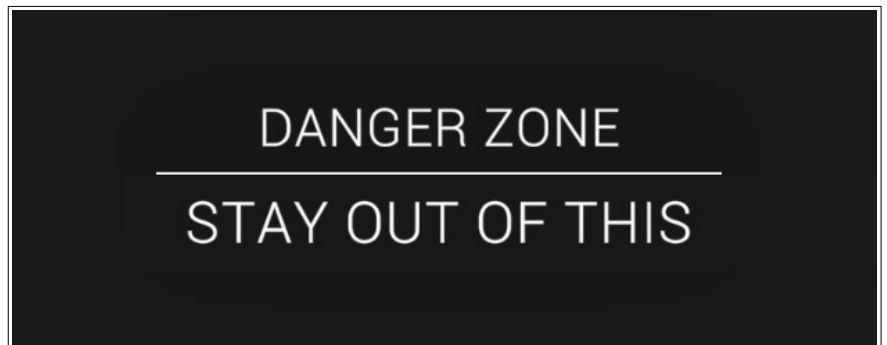
You can find the **Skill Panel Prefab** here



You can change the default **Skill Bar Icon sprite** on the prefab.

4. Primary Notification:

Shows a **notification** with **Main Title**, and **bottom Text**. You can use it for example to show name of an area. Or some event happening.



Create:

These functions will help you create the a primary notification.

```
CreatePrimaryNotification(string topTitleText, string bottomTitleText)
```

```
CreatePrimaryNotification(  
string topTitleText,  
string bottomTitleText,  
float StayTime)
```

```
CreatePrimaryNotification(  
string topTitleText,  
string bottomTitleText,  
float StayTime,  
float initialDelay)
```

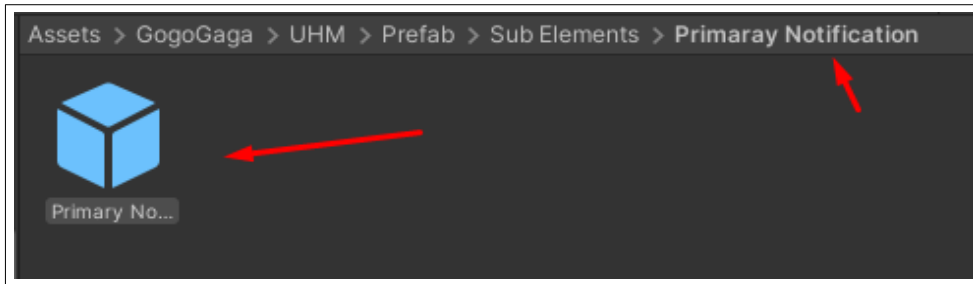
```
CreatePrimaryNotification(  
string topTitleText, string bottomTitleText,  
float StayTime, float initialDelay,  
Color topTextColor, Color bottomTextColor)
```

- **Top Title Text:** The title of top text.
- **Bottom Title Text:** The title of bottom text.
- **Stay Time:** Time in seconds for the title to stay on screen. By default it is 2.
- **Initial Delay:** Delay in seconds, by default it is 0.
- **Top Text Color:** By default it is white.

- **Bottom Text Color:** By default it is white.

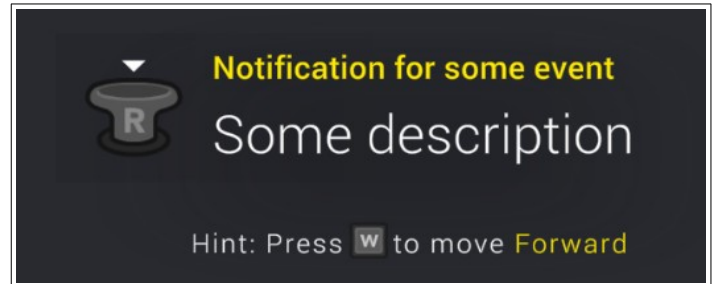
Other Properties:

You can find the prefab here.



5. Secondary Notification:

Create a notification for some event in game, **Title**, **Description**, **Icon**, And **Hint Text**.



Create/Show/Hide:

```
CreateSecondaryNotification( string topText,  
string bottomtext, float AutoHideDelay)
```

```
CreateSecondaryNotification(string topText,  
string bottomtext, string tooltip = "", Sprite icon = null)
```

```
CreateSecondaryNotification(string topText, string bottomtext,  
string tooltip, float AutoHideDelay, Sprite icon = null)
```

```
CreateSecondaryNotification( string topText,  
string bottomtext, string toolTip, float AutoHidedelay,  
Sprite icon, Color topTextColor, Color bottomTextColor)
```

- **Top Text:** Title of notification.
- **Bottom Text:** Description of the notification.
- **Tool tip Text:** Hint/tool-tip text of the notification. (How to add Input Icon in the hint, See Section “Input tool-tip→Text Formatting” Page 15).
- **Auto Hide Delay:** Should the notification be **auto-hide**? If not set the notification will stay on screen unless you hide it by calling Hide function.
- **Icon:** Notification sprite. If null there won't be any image with the notification



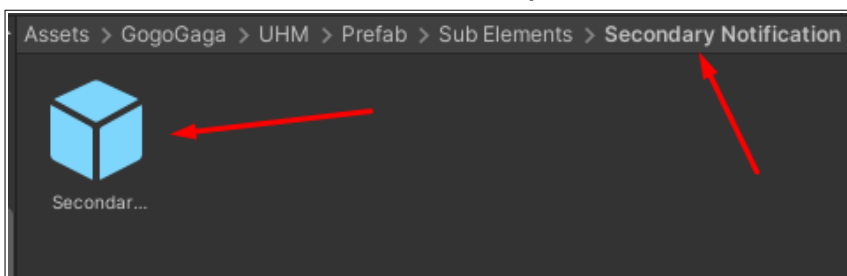
- **Top text Color:** The color of title text. By default it will be yellow.
- **Bottom text Color:** The color of the description text. By default it will be white.

If notification is not **auto-hide**, use this.

```
HideSecondaryNotification(float delay = 0)
```

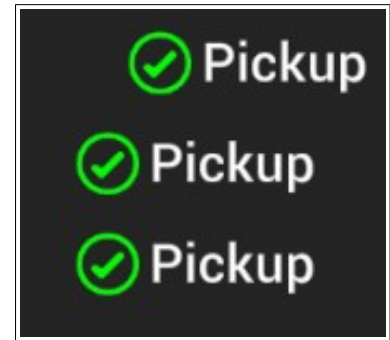
Other Properties:

You can access the notification prefab here:



6. Pickup Notification:

Simple notification for quick notification for player. Like if player picked up something etc.



Create:

Create a pick up notification.

```
CreatePickupNotification(string text,  
float delay = 0, float stayTime = 4)
```

```
CreatePickupNotification(string text,  
    Sprite icon = null, float delay = 0)
```

- **Text:** The text description of notification.
- **Icon:** Sprite of the notification. If null there won't be any image next to notification.
- **Delay:** If you want any delay before appearing of notification.
- **Stay time:** How long the notification stay. By default it 4 sec.

7. Input Tool-tip:

Simple notification for quick notification for player. Like if player picked up something etc.



Show/Hide:

Use these function to show input tool-tip.

```
ShowInputTooltipText(string preText, KeyCode key,
    string postText , float AutoHideAfter = -1)
```

- **Pre-Text:** Text before Input Icon.
- **Key:** Input key for icon.
- **Post-Text:** Text after input icon.
- **Auto Hide After:** This will auto hide the tool-tip. If not set the tool-tip will stay unless you hide it by calling.

Hide it using:

```
HideInputTooltip(float delay = 0)
```

```
ShowInputTooltipText(string text ,
    float AutoHideAfter = -1)
```

- **Text:** You can manually format the text string, explained below.

Text formatting:

Using "text mesh pro" we can have amazing text formatting.



















































You can find the documentation for that [here](https://docs.unity3d.com/Packages/com.unity.textmeshpro@4.0/manual/RichText.html).





































<https://docs.unity3d.com/Packages/com.unity.textmeshpro@4.0/manual/RichText.html>

For "Icons" in text you can use the following command inside you string text.
<sprite=value> where value can be from 0-94 mapped in chart below.

```
<sprite=0>
```

Dark color buttons

 Index: 0 Glyph ID: 0 Scale: 1	 Index: 10 Glyph ID: 10 Scale: 1	 Index: 20 Glyph ID: 20 Scale: 1	 Index: 30 Glyph ID: 30 Scale: 1	 Index: 40 Glyph ID: 40 Scale: 1
 Index: 1 Glyph ID: 1 Scale: 1	 Index: 11 Glyph ID: 11 Scale: 1	 Index: 21 Glyph ID: 21 Scale: 1	 Index: 31 Glyph ID: 31 Scale: 1	 Index: 41 Glyph ID: 41 Scale: 1
 Index: 2 Glyph ID: 2 Scale: 1	 Index: 12 Glyph ID: 12 Scale: 1	 Index: 22 Glyph ID: 22 Scale: 1	 Index: 32 Glyph ID: 32 Scale: 1	 Index: 42 Glyph ID: 42 Scale: 1
 Index: 3 Glyph ID: 3 Scale: 1	 Index: 13 Glyph ID: 13 Scale: 1	 Index: 23 Glyph ID: 23 Scale: 1	 Index: 33 Glyph ID: 33 Scale: 1	 Index: 43 Glyph ID: 43 Scale: 1
 Index: 4 Glyph ID: 4 Scale: 1	 Index: 14 Glyph ID: 14 Scale: 1	 Index: 24 Glyph ID: 24 Scale: 1	 Index: 34 Glyph ID: 34 Scale: 1	 Index: 44 Glyph ID: 44 Scale: 1
 Index: 5 Glyph ID: 5 Scale: 1	 Index: 15 Glyph ID: 15 Scale: 1	 Index: 25 Glyph ID: 25 Scale: 1	 Index: 35 Glyph ID: 35 Scale: 1	 Index: 45 Glyph ID: 45 Scale: 1
 Index: 6 Glyph ID: 6 Scale: 1	 Index: 16 Glyph ID: 16 Scale: 1	 Index: 26 Glyph ID: 26 Scale: 1	 Index: 36 Glyph ID: 36 Scale: 1	 Index: 46 Glyph ID: 46 Scale: 1
 Index: 7 Glyph ID: 7 Scale: 1	 Index: 17 Glyph ID: 17 Scale: 1	 Index: 27 Glyph ID: 27 Scale: 1	 Index: 37 Glyph ID: 37 Scale: 1	 Index: 47 Glyph ID: 47 Scale: 1
 Index: 8 Glyph ID: 8 Scale: 1	 Index: 18 Glyph ID: 18 Scale: 1	 Index: 28 Glyph ID: 28 Scale: 1	 Index: 38 Glyph ID: 38 Scale: 1	 Index: 48 Glyph ID: 48 Scale: 1
 Index: 9 Glyph ID: 9 Scale: 1	 Index: 19 Glyph ID: 19 Scale: 1	 Index: 29 Glyph ID: 29 Scale: 1	 Index: 39 Glyph ID: 39 Scale: 1	 Index: 49 Glyph ID: 49 Scale: 1

 Index: 50 Glyph ID: 50 Scale: 1	 Index: 60 Glyph ID: 60 Scale: 1	 Index: 70 Glyph ID: 70 Scale: 1	 Index: 80 Glyph ID: 80 Scale: 1	 Index: 90 Glyph ID: 90 Scale: 1
 Index: 51 Glyph ID: 51 Scale: 1	 Index: 61 Glyph ID: 61 Scale: 1	 Index: 71 Glyph ID: 71 Scale: 1	 Index: 81 Glyph ID: 81 Scale: 1	 Index: 91 Glyph ID: 91 Scale: 1
 Index: 52 Glyph ID: 52 Scale: 1	 Index: 62 Glyph ID: 62 Scale: 1	 Index: 72 Glyph ID: 72 Scale: 1	 Index: 82 Glyph ID: 82 Scale: 1	 Index: 92 Glyph ID: 92 Scale: 1
 Index: 53 Glyph ID: 53 Scale: 1	 Index: 63 Glyph ID: 63 Scale: 1	 Index: 73 Glyph ID: 73 Scale: 1	 Index: 83 Glyph ID: 83 Scale: 1	 Index: 93 Glyph ID: 93 Scale: 1
 Index: 54 Glyph ID: 54 Scale: 1	 Index: 64 Glyph ID: 64 Scale: 1	 Index: 74 Glyph ID: 74 Scale: 1	 Index: 84 Glyph ID: 84 Scale: 1	 Index: 94 Glyph ID: 94 Scale: 1
 Index: 55 Glyph ID: 55 Scale: 1	 Index: 65 Glyph ID: 65 Scale: 1	 Index: 75 Glyph ID: 75 Scale: 1	 Index: 85 Glyph ID: 85 Scale: 1	
 Index: 56 Glyph ID: 56 Scale: 1	 Index: 66 Glyph ID: 66 Scale: 1	 Index: 76 Glyph ID: 76 Scale: 1	 Index: 86 Glyph ID: 86 Scale: 1	
 Index: 57 Glyph ID: 57 Scale: 1	 Index: 67 Glyph ID: 67 Scale: 1	 Index: 77 Glyph ID: 77 Scale: 1	 Index: 87 Glyph ID: 87 Scale: 1	
 Index: 58 Glyph ID: 58 Scale: 1	 Index: 68 Glyph ID: 68 Scale: 1	 Index: 78 Glyph ID: 78 Scale: 1	 Index: 88 Glyph ID: 88 Scale: 1	
 Index: 59 Glyph ID: 59 Scale: 1	 Index: 69 Glyph ID: 69 Scale: 1	 Index: 79 Glyph ID: 79 Scale: 1	 Index: 89 Glyph ID: 89 Scale: 1	

There are also light color buttons with same indexing.



Example:

To achieve something like this:

The string would be

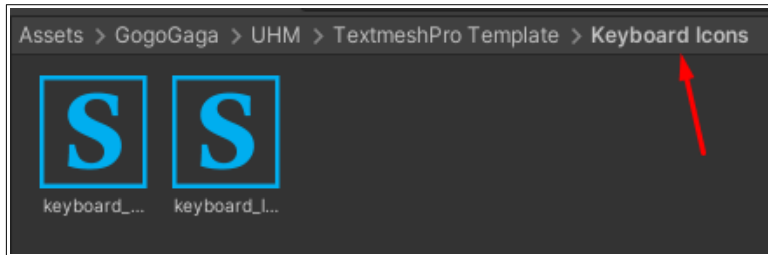
"Hint: Press <sprite=90> to move <color=yellow>Forward</color>"

Or there is a function that could make your life easier.

```
UltimateHudManager.Instance.GetInputIconCode(KeyCode.W)
```

This will **return** the **Tag**, like for example, for **W**. It will return <sprite=90>.

You can find the TextMeshPro sprite assets here.

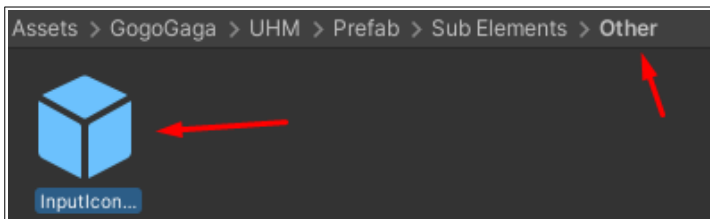


Input Key-code Icons:

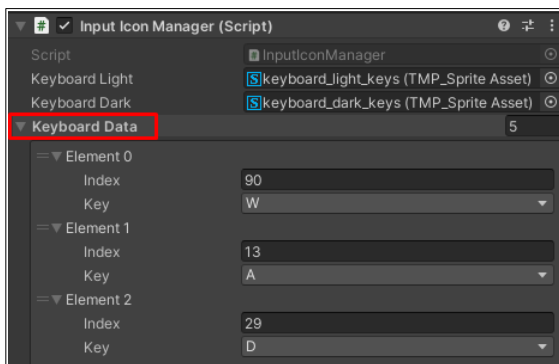
For each key you can assign an icon from the given list



This prefab has a list of key-codes.



We have added basic key-codes and there indexing, if you need more you can add those from the table given above.



8. Damage/Heal Effect:

Damage Effect

```
ShowDamageEffect(float startingDelay, bool loop)
```

```
ShowDamageEffect(float startingDelay, float displayTime = 0.2f)
```

```
ShowDamageEffect(bool loop = false)
```

Heal Effect

```
ShowHealEffect(float startingDelay, bool loop)
```

```
ShowHealEffect(float startingDelay, float displayTime = 0.2f)
```

```
ShowHealEffect(bool loop = false)
```

-
- **Starting Delay:** Delay before starting effect, by default 0
 - **Loop:** If you want to loop or not, By default it will play once.
 - **Display Time:** If not loop, how long should the effect be. By default it is 0.2 sec.

9. Cross-hair:

UHM has build in **3** cross hair type. With **Idle, Highlight, Hit, and Shoot** effect.



Create/Change:

You can select from given

```
CreateCrossHair(int CrossHairTypeFromAvaliable)
```

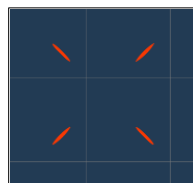
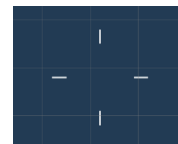
Use your own

```
CreateCrossHair(CrosshairGraphics crossHairPrefab)
```

Set color of cross hair using

```
SetCrossHairColor(Color newColor)
```

```
SetCrossHairHitColor(Color newColor)
```



Or both at once

```
SetCrossHairColors(Color newColor,Color newHitColor)
```

Effects:

```
IdleEffectCrossHair()
```

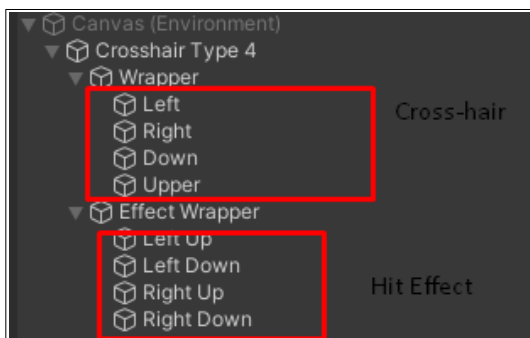
```
HighlightEffectCrossHair()
```

```
ShootEffectCrossHair()
```

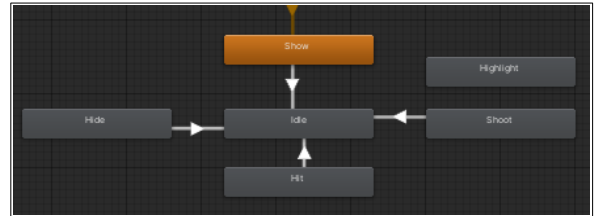
```
HitEffectCrossHair()
```

Create custom cross-hair:

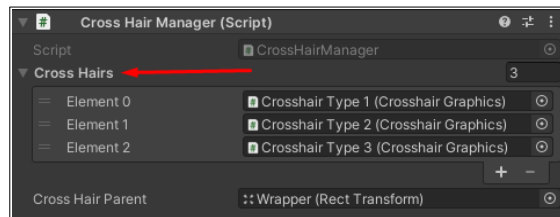
You can duplicate one of these prefab to create your own cross hair



If you want to make new animation for custom cross-hair, don't edit the ones already assigned. Create/Duplicate new animations and animation-controller and keep the structure of animation-controller same as shown here.



You can use newly created cross-hair by adding to the array like this



Now you can use this with respective index.

```
CreateCrossHair(int CrossHairTypeFromAvaliable)
```

Or you reference newly created cross-hair prefab and use this function.

```
CreateCrossHair(CrosshairGraphics crossHairPrefab)
```

10. Health-bar:

Health bar with extras health.



Create:

Create health bar using.

```
CreateHealthBar(HeathData heathData)
```

```
HeathData(  
    float health,  
    float maxHealth,  
    float extraHealth = 0,  
    float maxExtraHealth = 0)
```

Health-Data is structure with following parameters.

Add/Set:

Add some positive or negative value.

```
AddHealth(float add)
```

```
SetHealth(float newHealth)
```

```
SetMaxHealth(float newMaxHealth)
```

For extra health:

```
AddExtraHealth(float add)
```

```
SetMaxHealth(float newMaxHealth)
```

```
SetExtraMaxHealth(float newExtraMaxHealth)
```

Set color for health bar:

```
SetHealthbarColor(Color newColor)
```

```
SetExtraHealthbarColor(Color newColor)
```

```
SetHealthbarColor(Color upperBarColor,Color lowerBarColor)
```

Hide:

```
HideHealthBar()
```

 Just hide the health panel.

```
RemoveHealthBar()
```

 Hide and reset (data) the health panel.

11. Fade Out/In:

Show/Hide:

```
ToggleFade(float FadeDuration = 0.3f)
```

```
ToggleFade(Color color, float time = 0.3f)
```

```
FadeInToBlack(float time = 0.3f)
```

```
FadeInToWhite(float time = 0.3f)
```

```
FadeInToColor(Color fadeColor, float time = 0.3f)
```

Fade out

```
FadeOut(float time = 0.3f)
```

Note: By default fade out panel will be displayed over top of every other UHM element, if you want to change that you can be using following features:

```
MoveFadeToBottom()
```

```
MoveFadeToTop()
```

12. Way-point Marker:

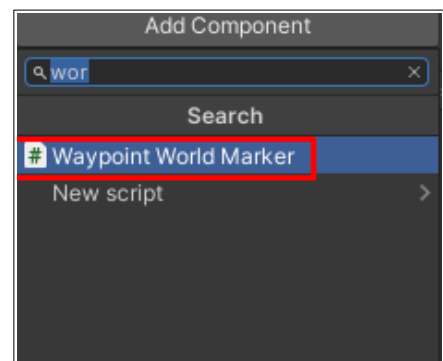


You can add way-point marker to game-object. Way-point marker will appear as a **marker GUI point hovering over** the target and pointing to the direction of object and as an **GUI indicator in the compass**.

Note: Way-point will be using main camera to measure direction/distance.

Create:

You can just add **“Waypoint World Marker”**
You can tweak **“Waypoint Marker Properties”**
which are explained in next section.



OR

Through UHM prefab.

This will create and return a Way-point Marker at some position, with default properties.

```
WaypointWorldMarker CreateWaypoint(Vector3 position)
```




You can use custom properties.

```
CreateWaypoint(WaypointMarkerProperties properties, Vector3 position)
```

Or you can use a prefab to instantiate.

```
CreateWaypoint(WaypointWorldMarker marker, Vector3 position)
```

These are the properties for a way-point marker.

<u>Commons</u>		
Title	Title	
Position Offset	X 0	Y 0 Z 0
UI Offset From Edge	X 0	Y 0
Hide When Close Distance	0	
Hide Fading Smoothing	<input type="range"/> 1	
<u>On screen</u>		
Show On Screen	<input checked="" type="checkbox"/>	
On Screen Image	None (Sprite) 	
On Screen Image Color	<input type="color"/>	
On Screen Size	<input type="range"/> 1	
On Screen Rotation	<input type="range"/> 0	
<u>Off screen</u>		
Show Off Screen	<input checked="" type="checkbox"/>	
Off Screen Image	None (Sprite) 	
Off Screen Image Color	<input type="color"/>	
Off Screen Size	<input type="range"/> 1	
Off Screen Rotation	<input type="range"/> 0	
Rotate To Target	<input checked="" type="checkbox"/>	
<u>Text properties</u>		
Text Mesh Text Preset	None (TMP_Font Asset) 	
Show Distance Text On Screen	<input checked="" type="checkbox"/>	
Show Distance Text Off Screen	<input type="checkbox"/>	
Distance Text Color	<input type="color"/>	
Show Title Text On Screen	<input checked="" type="checkbox"/>	
Show Title Text Off Screen	<input type="checkbox"/>	
Title Text Color	<input type="color"/>	
<u>other properties</u>		
Show As World Marker	<input checked="" type="checkbox"/>	
Show As Compass Marker	<input checked="" type="checkbox"/>	

Remove:

You can just destroy Way-point World Marker Game-object that is returned.

Or You can remove all the way-points at once by using.

```
RemoveAllWaypoints()
```

Properties:

Common:

- **Title:** Text you want to show below marker. Except compass.
- **Position Offset:** If you want to have offset from spawn position. Except compass.
- **UI offset:** If you want offset from the edges for the marker UI.
- **Hide When Close Distance:** When the you (main camera) is near to that distance to the marker, the marker UI will fade-off.
- **Hide Fade Smoothing:** Fade out smoothing, when close to the "Hide When Close Distance" the object will fade from 0-1 alpha.

On Screen (When the marker position is in camera view):

- **Show On Screen:** Show the marker UI when on the way-point is on screen?
- **On Screen Image:** Sprite when marker is on screen. If null it will use the default (shown in next section).
- **On Screen Image Color:** Color of the sprite on screen.
- **On Screen Size:** Size of sprite on screen.

- **On Screen Rotation:** If you want rotation offset for on screen image.

Off Screen (When the marker position is not in camera view):

- **Show Off Screen:** Show the marker UI when on the way-point is off screen?
- **Off Screen Image:** Sprite when marker is off screen. If null it will use the default (shown in next section).
- **Off Screen Image Color:** Color of the sprite off screen.
- **Off Screen Size:** Size of sprite off screen.
- **Off Screen Rotation:** If you want rotation offset for off screen image.
- **Rotate To Target:** If turned on, while the object is off screen, the marker sprite will rotate to point in the direction on target.

Off Screen

- **TextMeshPro Text Preset:** If you want to use a custom TextMeshPro Font asset.
- **Show Distance On Screen:** Show distance on screen?
- **Show Distance Of Screen:** Show distance off screen?
- **Distance Text Color:** Color for the distance text.
- **Show Title On Screen:** Show title on screen?
- **Show title Of Screen:** Show title off screen?
- **Title Text Color:** Color for the title text.

Other Properties

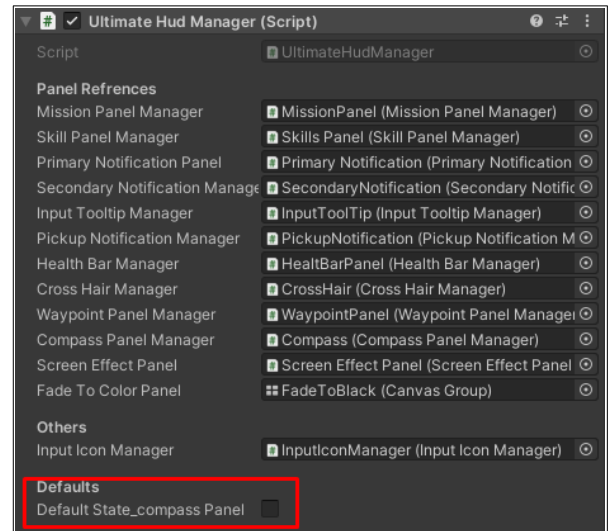
- **Show As World Marker:** Show the UI for world marker?
- **Show As Compass Marker:** If turned on, the marker will be shown in the compass at the top.

Compass:

You can turn on compass by default on UHM prefab

Or you can show/hide compass at runtime

```
ShowCompass(bool show = true)
```



13. Progress Tool-tip:



You can use progress tool-tip to show some kind of unlocking progress.

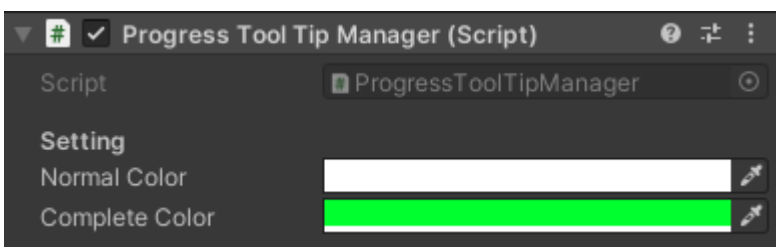
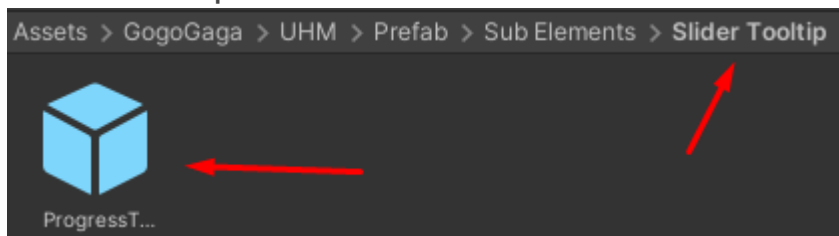
Create:

```
ShowProgressTooltip(float progressTime , string titleText,  
                    bool autoHideOnComplete = true)
```

- **Progress Time:** Time it will take to complete the progress.
- **Title text:** Text on the right side of progress.
- **Auto Hide On Complete:** If false the panel will not hide on completing the progress.

Properties:

You will find prefab here:



You can change color for when the progress is complete.

