

# MALOKE GAMES

## ASSETS

### ANALOGICAL FLIGHT INSTRUMENTS

High Fidelity GUI

v 1.1

- 
- **Contact:** Maloke7-Games@yahoo.com.br
  - **Full Portfolio:** <https://maloke.itch.io/>
  - **AssetStore:** <https://assetstore.unity.com/publishers/26634>
-

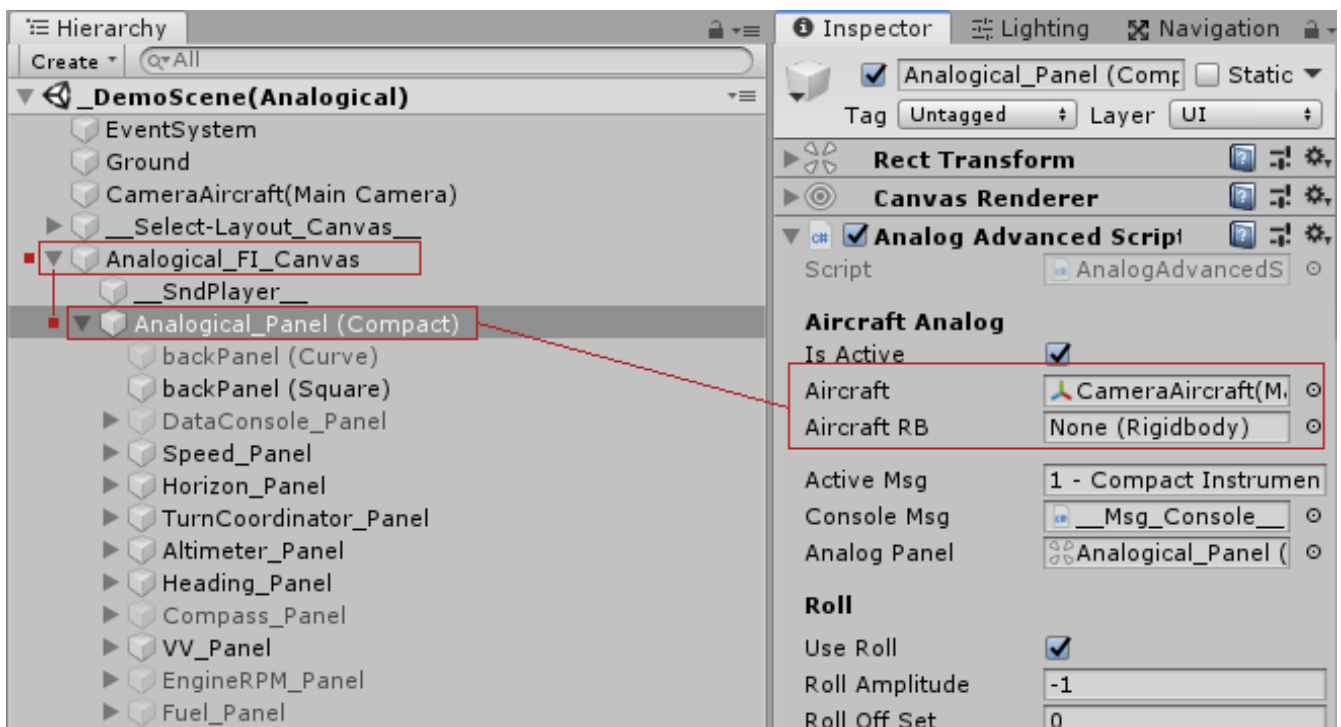
### ➤ Quick Instructions:

You can find a DemoScene with this asset configured and working straight away for 10 different layouts as examples (**or you can create your own design using the Template prefab!**). The demo uses a very basic camera movement script applied to the main camera that emulates an aircraft movement to help you visualize properly how it works.

If you want to use on your own project/aircraft you **just need to link a reference of your aircraft's Transform and/or Rigidbody to the main script**. If you leave these fields **empty** it will automatically look for the current **MainCamera** in the scene.

Just follow these 3 simple steps:

- 1- Drop on your scene any of the "xxx\_AFI\_CANVAS.prefab" prefab.
- 2- Locate the main script "**AnalogAdvancedScript**" as shown in the image:



3- Then simply drag your aircraft object to the field "**Aircraft**" and you are ready to go. If your aircraft has a **Rigidbody** then drag it too to the "**AircraftRB**" field. Using an **RB** will ensure more precision on some physics calculations, but the script can fully work using **Transform** only too.

That's it, now you are ready to go!

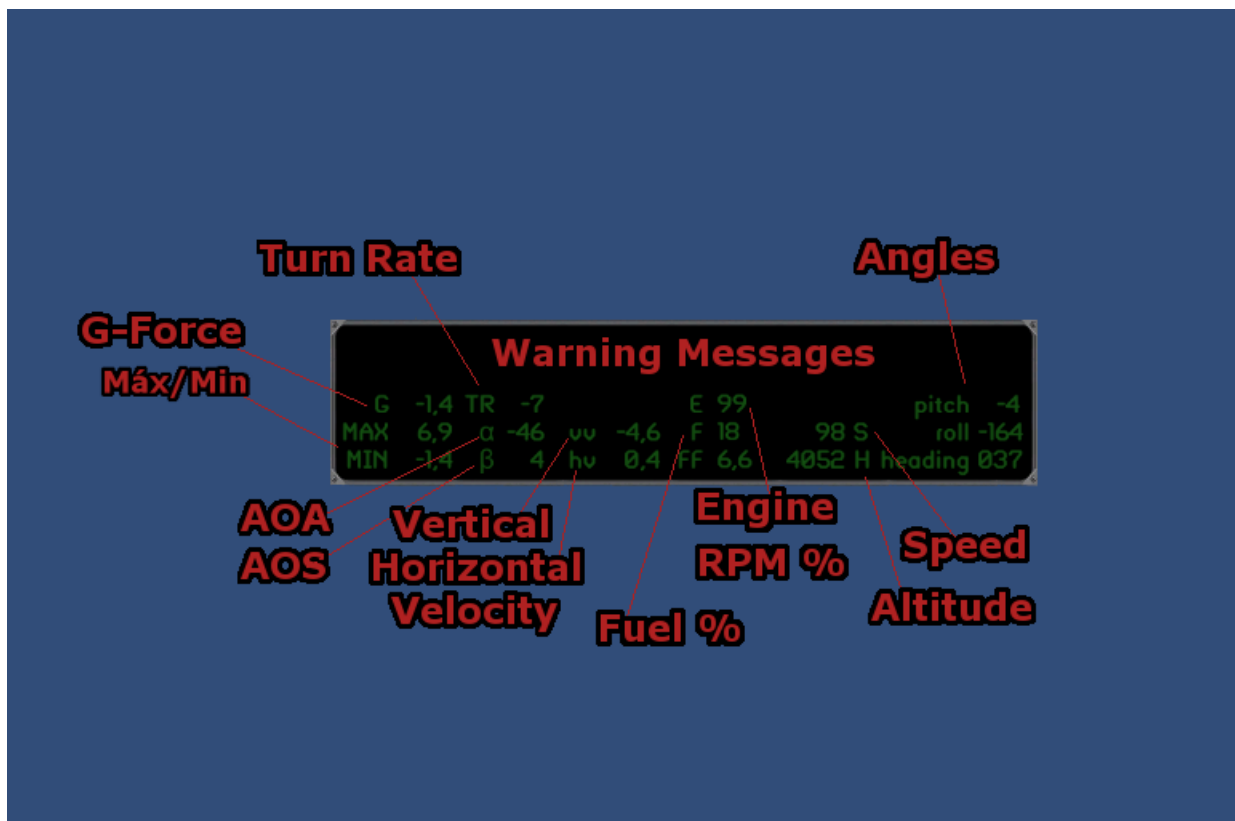
If you want to know more about this asset you can find extra information further on this document.

➤ Instruments Symbolology:



➤ Data Console:

Here you can visualize all the flight variables and also send messages to the console.



**Fuel %**



**Compass Heading**



**Engine RPM %**



**Speed**



**Pitch  
Angles**



**Roll  
Angle**

**Altitude**



**Artificial Horizon Line**

**Turn Rate**



**Slip Angle (AOS) /  
Horizontal Velocity**



**Heading**



**Vertical Velocity**

➤ **Instruments Main Layouts:**

This asset comes with premade layouts but you can use the Template Prefab to create your own. The panel background is a tileable image.

**\* Compact or Essential Instruments Panel \***



**\* Full Panel \***





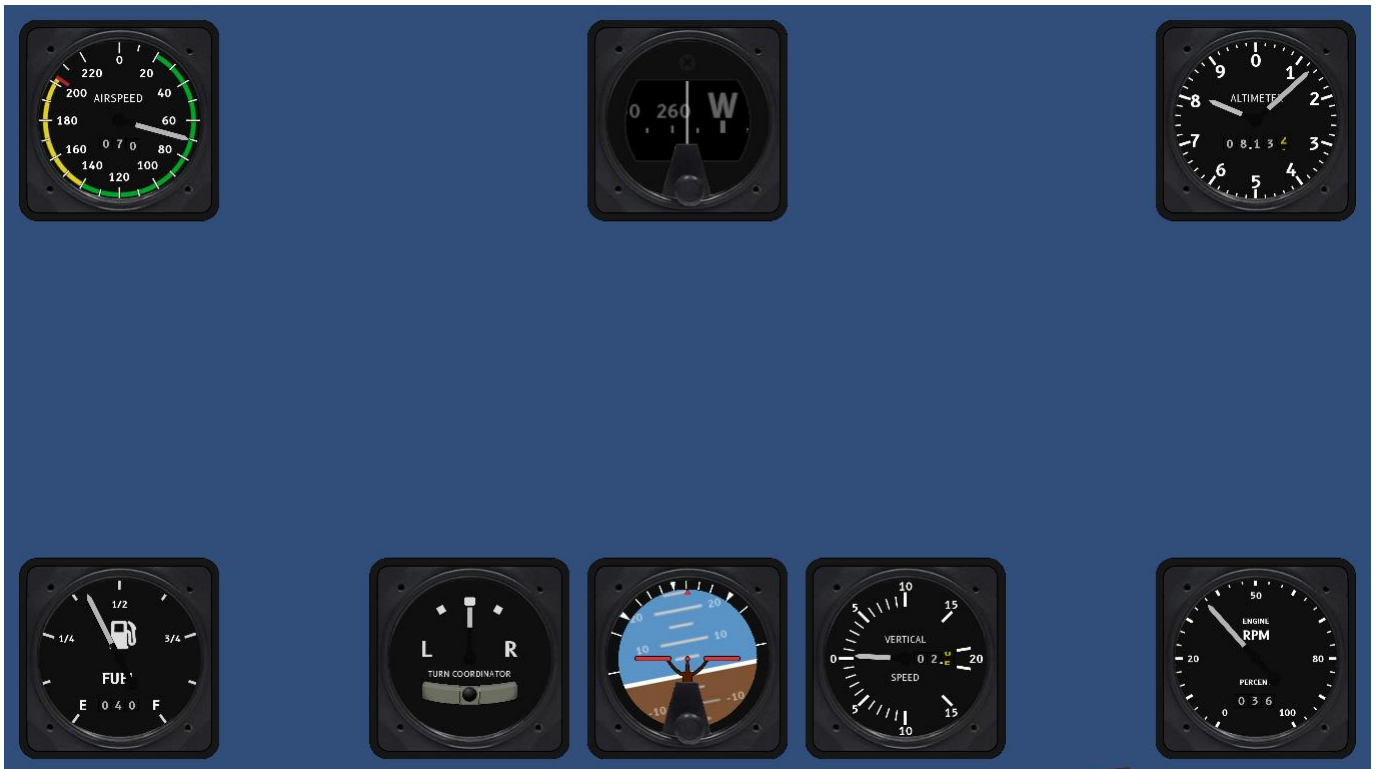
**\* Full Panel - Square \***



**\* Simple Panel \***



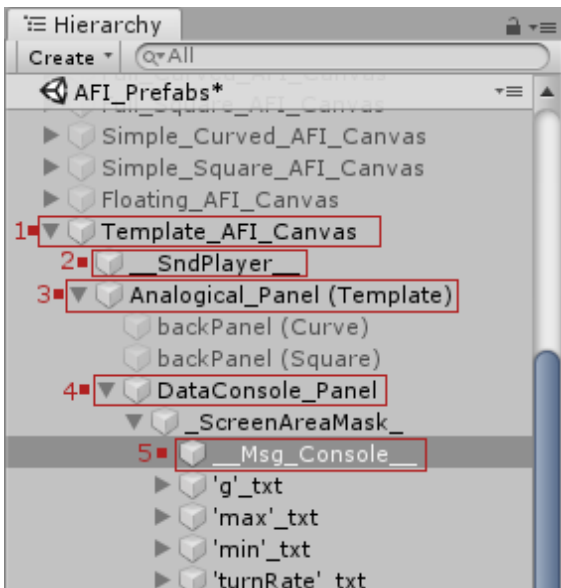
**\* Floating Mode \***



**\* Full Template Instruments \***



## ➤ Main Components:



1- The MainCanvas of this Asset.

2- Contains script "**SndPlayer**" that manages the sounds.

3- The main script " **AnalogAdvancedScript** " which controls the whole Instruments and calculations.

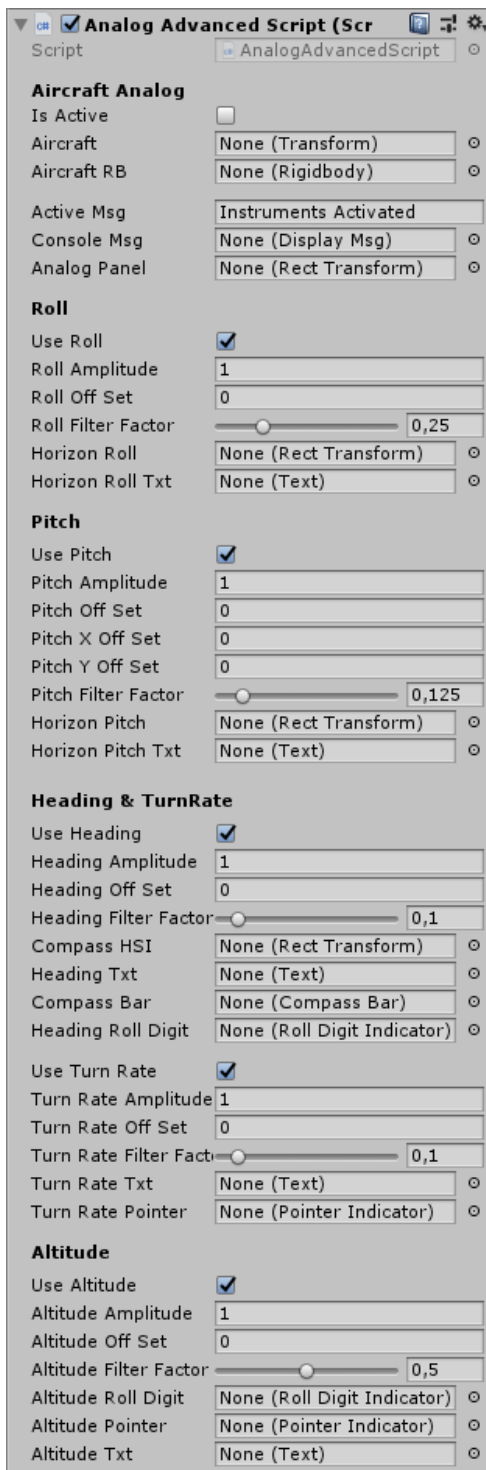
4- The DataConsole which displays all flight variables at runtime and console messages. Can be used as an extra instrument or just as an extra help while tweaking values.

5- Contains the script "**DisplayMsg**" which is responsible to send/show messages to the text console of the DataConsolePanel.

*\*Some of these components are used "under the hood" by the asset and do not require any setup or configuration... but fell free to use them on your project if you like!*



## ➤ AnalogAdvancedScript - Structure in Editor:



- "**isActive**" determines if the script should be active.

- "**Aircraft**"(*Transform*) and "**AircraftRB**"(*RigidBody*) are the references to your flying gameObject which will be used to calculate all the values displayed on the instruments.

- **ActiveMsg** is the *string* displayed on the console when this instruments panel is activated.

- **AnalogPanel** is an internal reference for the *panel* itself.

---

Each section below corresponds to a **Flight Variable** and the following pattern applies to all of them:

-The *bool* values "**useXXX**" determine if that variable will be calculated by the script.

- The "**xxxAmplitude**" is a value multiplied to that variable after calculations and before showing on the GUI. It can work as a **Scale** or as a **unit conversion factor**.

- The "**xxxOffset**" is a value added to the variable after calculation. It can be used to **unit conversion** or to fix **objects orientation** properly.

- Note that for **Pitch** variable we have two extra **Offset** options in **X** and **Y**. These are visual offsets in the position of the Pitch Angles or Horizon Line graphics used on the GUI and not for the actual Pitch value itself.

- All the "**xxxFilterFactor**" values are used to smooth the value shown (*works as a lowpass filter*). If set to **1** it will **show direct value** and will have no filtering at all. If set too close to **0** it **will take more time to reach the actual value**.

- The "**HorizonXXX**" and other **RectTransform** entries are references to the GUI object used to represent that variable visually.

- The "**CompassBar**" is a reference to the script that controls the compass sliding position to match visually the current heading.

- All "**xxxTXT**" are references to a ui *text* component that represent the variable in text format on the HUD.

**Speed**

Use Speed ☒

Speed Amplitude 1

Speed Off Set 0

Speed Filter Factor 0,25

Speed Needle None (Needle Indicator) ○

Speed Roll Digit None (Roll Digit Indicator) ○

Speed Pointer None (Pointer Indicator) ○

Speed Txt None (Text) ○

**Vertical Velocity**

Use VV ☒

Vv Amplitude 1

Vv Off Set 0

Vv Filter Factor 0,1

Vv Needle None (Needle Indicator) ○

Vv Arrow None (Arrow Indicator) ○

Vv Roll Digit None (Roll Digit Indicator) ○

Round VV ☒

Show Decimal VV ☒

Round Factor VV 0,1

Vertical Speed Txt None (Text) ○

**Horizontal Velocity**

Use HV ☒

Hv Amplitude 1

Hv Off Set 0

Hv Filter Factor 0,1

Hv Needle None (Needle Indicator) ○

Hv Arrow None (Arrow Indicator) ○

Round HV ☒

Show Decimal HV ☒

Round Factor HV 0,1

Horizontal Speed Txt None (Text) ○

**G-Force**

Use G Force ☒

G Force Amplitude 1

G Force Off Set 0

G Force Filter Factor 0,25

G Force Txt None (Text) ○

Max G Force Txt None (Text) ○

Min G Force Txt None (Text) ○

**AOA, AOS and GlidePath**

Use Alpha Beta ☒

Alpha Amplitude 1

Alpha Off Set 0

Alpha Filter Factor 0,25

Alpha Needle None (Needle Indicator) ○

Alpha Arrow None (Arrow Indicator) ○

Alpha Txt None (Text) ○

Beta Amplitude 1

Beta Off Set 0

Beta Filter Factor 0,25

Beta Needle None (Needle Indicator) ○

Beta Arrow None (Arrow Indicator) ○

Beta Txt None (Text) ○

Use Glide Path ☒

Glide Path Filter Fact 0,1

Glide X Delta Clamp 600

Glide Y Delta Clamp 700

Glide Path None (Rect Transform) ○

- The "**XXNeedle**" is a reference to the script that controls the visual indication of that value using the angle of the UI needle instrument.

- The "**XXPointer**" is a reference to the script that controls the visual indication of that value using the angle of the UI pointer instrument.

- The "**XXArrow**" is the same as the needle and pointer, but it uses a reference to the script that controls the visual indication using an arrow translation instead of an angle.

- The "**RoundXX**" option makes that value be rounded by steps using the "**RoundFactorXX**". If round factor is set to "**10**" the value will be shown in steps like "**10, 20, 30, ...**". If set to "**0.1**" it will be shown like "**0.1, 0.2, 0.3, ...**".

- The bool "**ShowDecimalXX**" determines if the GUI will show the decimal point or not for this variable.

---

- The **AOA** (*Angle of Attack*), **AOS** (*Angle of Slip*) and the **GlidePath** (*also called Velocity Vector or flight path vector FPV*) are specific terms on aviation and engineering that are a bit complex to explain here, so here is some Wiki links with more detailed explanation about them:

- [https://en.wikipedia.org/wiki/Head-up\\_display#Displayed\\_data](https://en.wikipedia.org/wiki/Head-up_display#Displayed_data)
- [https://en.wikipedia.org/wiki/Angle\\_of\\_attack](https://en.wikipedia.org/wiki/Angle_of_attack)
- [https://en.wikipedia.org/wiki/Slip\\_\(aerodynamics\)](https://en.wikipedia.org/wiki/Slip_(aerodynamics))

- The "**Glide XY Delta Clamp**" determines how much the **Glide Path** indicator will be clamped from the center position. If you do not wish to clamp it inside the HUD, then you can just set these to some big value.

Engine and Fuel	
Use Engine	<input checked="" type="checkbox"/>
Engine Amplitude	1
Engine Off Set	0
Engine Filter Factor	<input type="text" value="0,0125"/>
Engine Pointer	None (Pointer Indicator) <input type="button" value="o"/>
Engine Roll Digit	None (Roll Digit Indicator) <input type="button" value="o"/>
Engine Txt	None (Text) <input type="button" value="o"/>
Use Fuel	<input checked="" type="checkbox"/>
Fuel Amplitude	1
Fuel Off Set	0
Fuel Filter Factor	<input type="text" value="0,0125"/>
Fuel Pointer	None (Pointer Indicator) <input type="button" value="o"/>
Fuel Roll Digit	None (Roll Digit Indicator) <input type="button" value="o"/>
Fuel Txt	None (Text) <input type="button" value="o"/>
Fuel Flow Txt	None (Text) <input type="button" value="o"/>
External Controllers	
Engine Target	<input type="text" value="0,75"/>
Follow Max Speed	0
Engine AS	None (Audio Source) <input type="button" value="o"/>
Min Pitch	0,25
Max Pitch	2
Fuel Target	<input type="text" value="0,8"/>
Maxfuel Flow	1
Idlefuel Flow	0,25
Flight Variables - ReadOnly!	
Speed	0
Altitude	0
Pitch	0
Roll	0
Heading	0
Turn Rate	0
G Force	0
Max G Force	0
Min G Force	0
Alpha	0
Beta	0
Vv	0
Hv	0
Engine	0
Fuel	0
Fuel Flow	0

- The **Engine** and **Fuel** variables are a **percentage** from 0 to 1 (empty/full).

- On **External Controllers** you can manage the current **Engine RPM** and **Fuel quantity** using the Target sliders (from 0 to 1 for empty/full).

- If **follow MaxSpeed** is set to **Zero**, then you can manually control the Engine RPM. If you set it to any value, the RPM will synchronize automatically to correspond 100% to that airspeed and interpolates it to zero when stopped.

- If the **EngineAudioSource** is not null, then the source **audio pitch** will automatically follow from **Min** to **MaxPitch** value and thus simulating the engine RPM sound accordingly.

- The **IdleFuelFlow** and **MaxFuelFlow** determine how much **% of the fuel is consumed for one minute**. The **IdleFuelFlow** determines *the minimum consumption*, while **Max** determines consumption **when Engine RPM is at 100%** and interpolates it in-between.

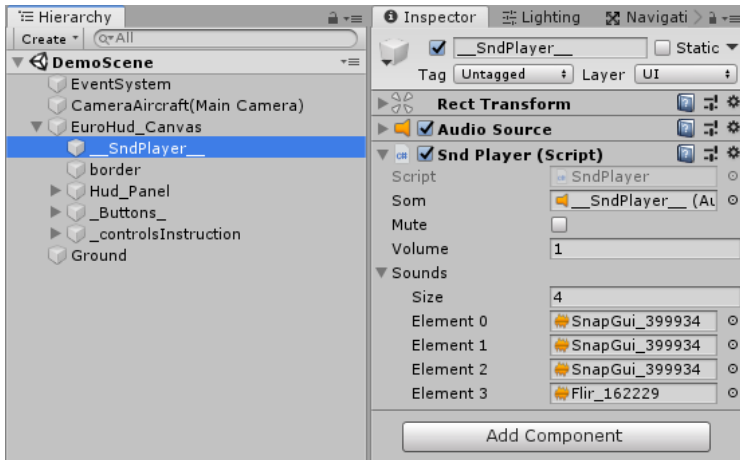
---

- On the **Flight Variables** section you can see all current values for all the variables in real time inside the Editor. This is just for debug or tweeking and are **ReadOnly!**

- You can also visualize these variables during runtime using the **DataConsole**.

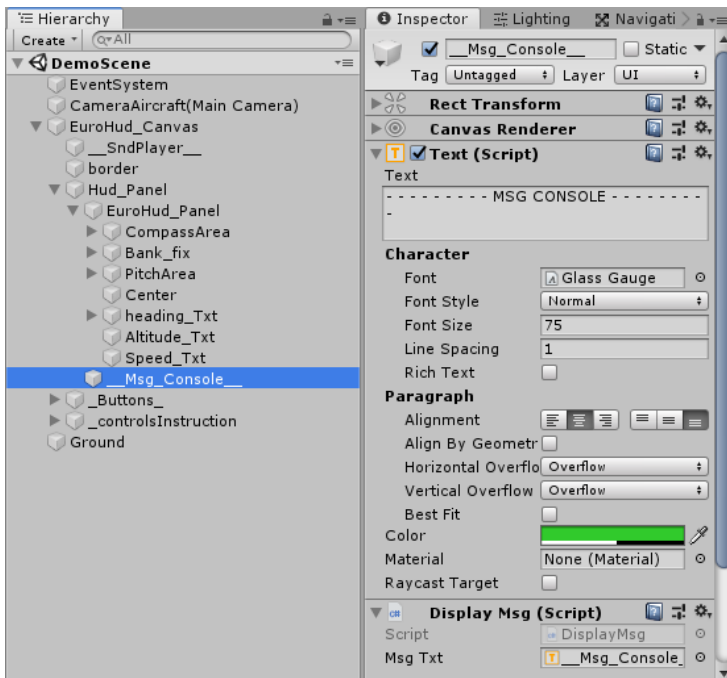
## ➤ Secondary Components (Sound and Message Console) :

If you wish extra functionalities, you can make use of these components by script calling statics methods:



```
-public static void play(int index);  
-public static void play(AudioClip clip, float volume = 1f);
```

(Plays the sound listed on array Sounds with index position or the audioclip itself)

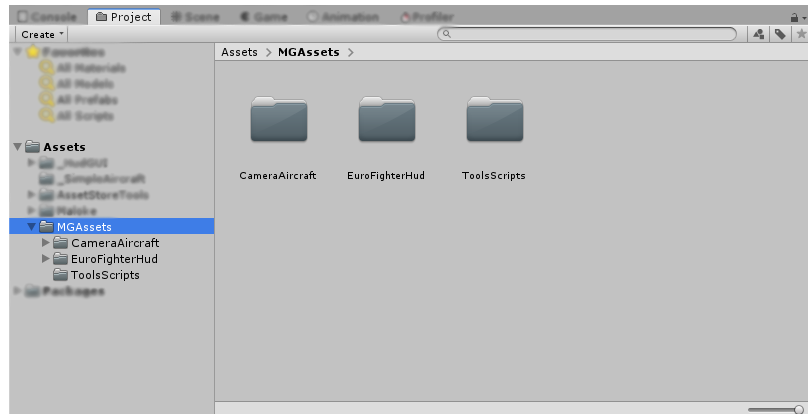


```
-public void displayMsg(string msg = "");  
-public void displayQuickMsg(string msg = "");  
-public static void show(string msg = "", float timed = 0);
```

Displays a string message on the bottom of the HUD for an amount of seconds (quick is 5s).

### ➤ Asset's Folders Organization:

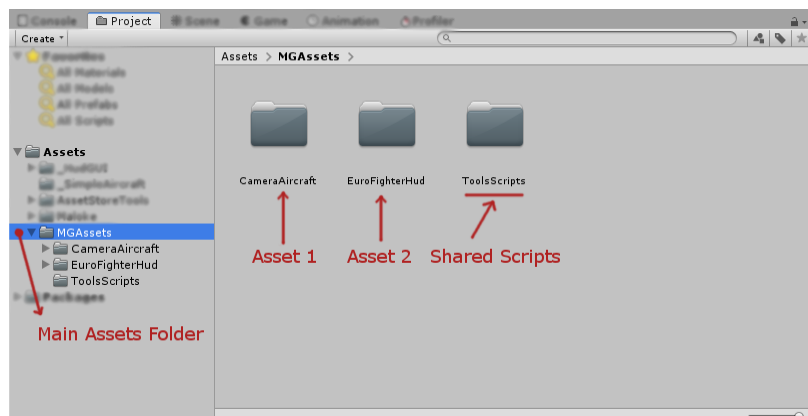
All assets and packages from MalokeGamesAssets will be downloaded/unpacked to a folder called **"MGAssets"** inside the Unity's **"Assets"** root:



Inside **"MGAssets"** you will find a separate folder for each asset package and all their specific resources (like data, scripts, textures, sprites, prefabs, demo scenes and so on...) will be found inside and organized in their respective subfolders.

Notice that some assets may use "under the hood" some general scripts and shared functionalities, so for this reason (and to avoid duplicity or accidental deletion) you will find all this shared tools inside a folder called **"ToolsScripts"**.

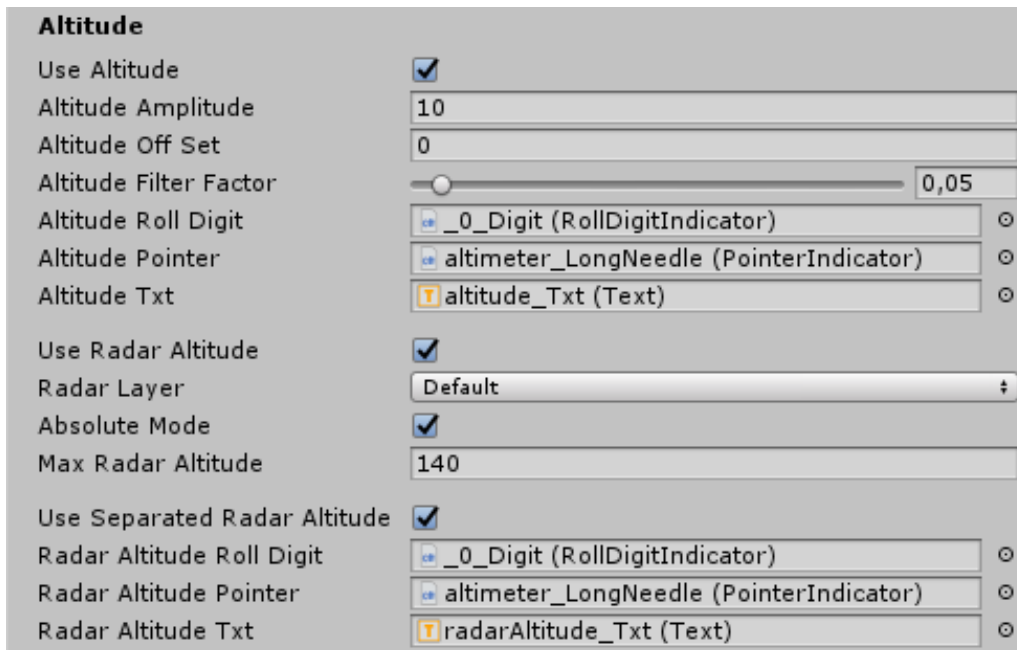
Feel free to explore and use them on your projects too, they are simple but handy!





### ➤ Update 1.1 - Radar Altimeter + Bevel Frame:

A new instrument has been added. The Radar Altimeter will display the actual distance the aircraft is from the floor. See below the explanation of its factors:



The screenshot shows a configuration window titled "Altitude". It contains two sections of settings. The first section includes: "Use Altitude" (checked), "Altitude Amplitude" (10), "Altitude Off Set" (0), "Altitude Filter Factor" (0,05), "Altitude Roll Digit" (\_0\_Digit (RollDigitIndicator)), "Altitude Pointer" (altimeter\_LongNeedle (PointerIndicator)), and "Altitude Txt" (altitude\_Txt (Text)). The second section includes: "Use Radar Altitude" (checked), "Radar Layer" (Default), "Absolute Mode" (checked), "Max Radar Altitude" (140), "Use Separated Radar Altitude" (checked), "Radar Altitude Roll Digit" (\_0\_Digit (RollDigitIndicator)), "Radar Altitude Pointer" (altimeter\_LongNeedle (PointerIndicator)), and "Radar Altitude Txt" (radarAltitude\_Txt (Text)).

- The "**UseRadarAltitude**" bool will enable the script calculation for ground height.
- The layer "**RadarLayer**" can be used to ignore or include some objects like trees.
- If "**AbsoluteMode**" is enabled the displayed height will be calculated independent of the aircraft attitude, otherwise, it will behave realistically detecting obstacles straight down relative to the aircraft attitude.
- "**MaxRadarAltitude**" indicates the maximum range the radar can detect the ground. If no obstacles are found within this range, the instrument's needle will display the máximo value.
- The "**UseSeparatedRadarAltitude**" bool will enable the extra instrument dedicated to showing the ground height. If this is disabled, then the default altimeter will display the ground height instead of the altitude.

This update also included alternative bevel frames for the instruments:

