MANAGERS HUB

The perfect solution for all your project.

Managers HUB is a simple and powerful tool that regroup all of the most use managers in the game industry! The hub has been designed to be extremely efficient and incredibly **easy to use** by any level programmer/designer. All managers have their own color assign to them and their data's, that way you know easily what you're editing and make it easier to understand! All setting(s) for each manager are use with Scriptable Object (Data) and all have their own custom interface, so you can create your game **simply and quickly**.

AVAILABLE MANAGERS:

- Audio Manager 🟦
- Pool Manager 🟧
- Scene Manager 🟥
- Language Manager 🟩

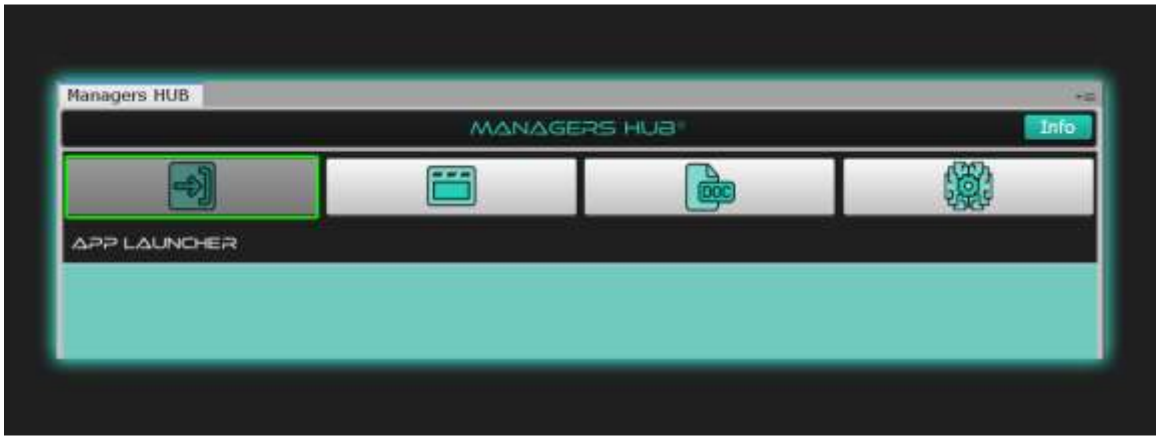
MANAGERS TO COME:

- Spawn 🟦
- Option 🟪
- Save 🟩
- Multiplayer 🟨
- Inventory 🟥

Thanks everyone for using the Managers HUB! If you have any problems, questions, comments or/and suggestions, you can contact me at:

THIERRY.RIOUX@BELIEVEIT.GAMES

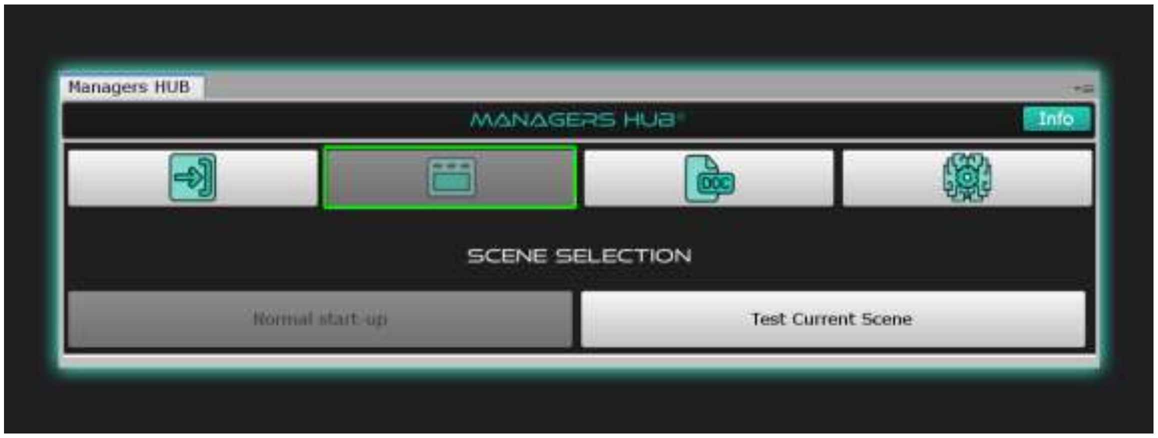
I'll answer as soon as possible!



APP LAUNCHER:
The AppLauncher button lets you access the AppLauncher scene really quickly. Just click it and Unity will change scene for you.

In AppLauncher Scene:	Load the previous scene you were working on (if exist). This way you can edit manager really quickly without losing what you were doing.
In Other Scene:	Load the AppLauncher scene and save current scene in memory.

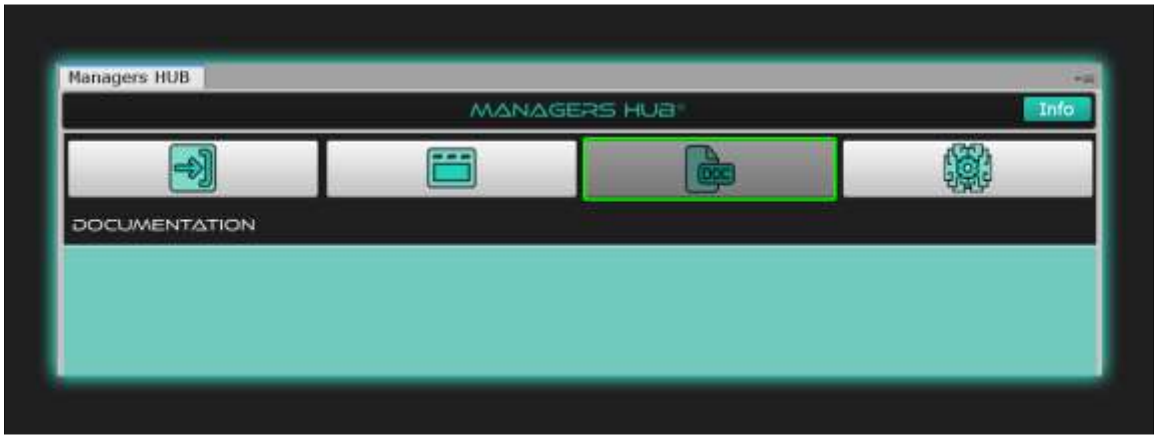
P.S. The AppLauncher will flash red if you're in the Manager Creator and not in the AppLauncher. Its to warn you that you need to be in the AppLauncher to create manager.



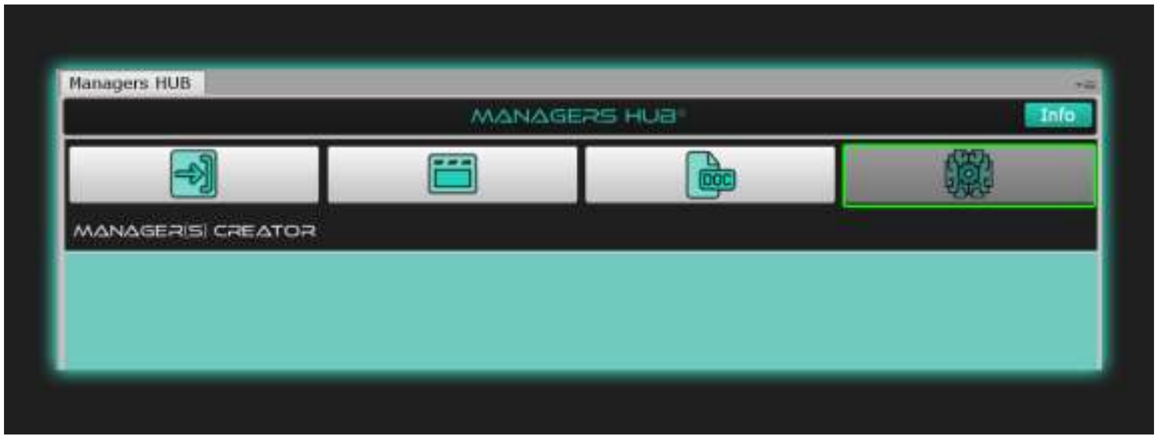
SCENE SELECTION:
The scene selection button lets you choose how Unity will launch your game in the editor (not in a build) when press PLAY.

Normal Start-up:	*Unity will start your game normally, like a build, with standard game-flow.*

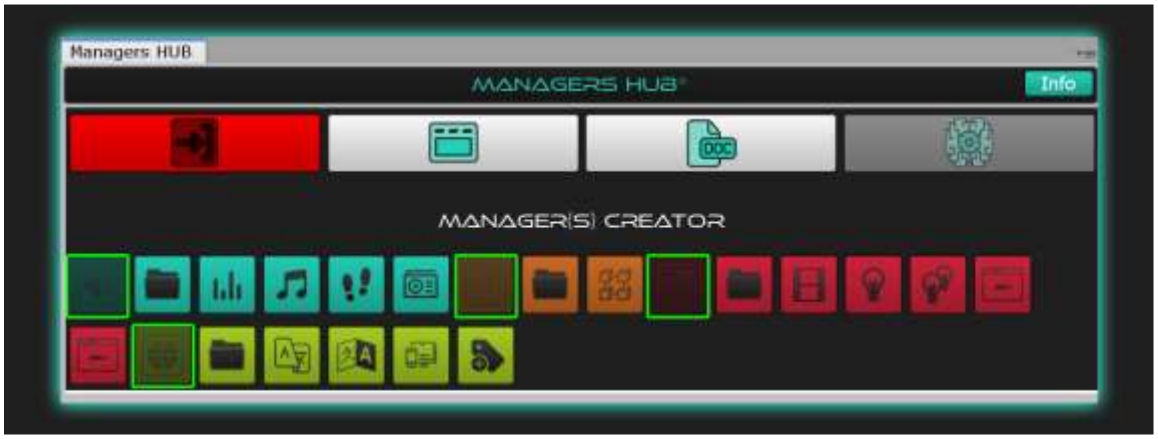
Test Current Scene:	*Unity will load the AppLauncher to activate managers and go directly to the scene you were working on, so you can test your scene without doing all the game-flow.*
WARNING:	*Test current scene is disable when you're in the AppLauncher.*



DOCUMENTATION:
Open the documentation for the tools (You already know if you're here 😉)

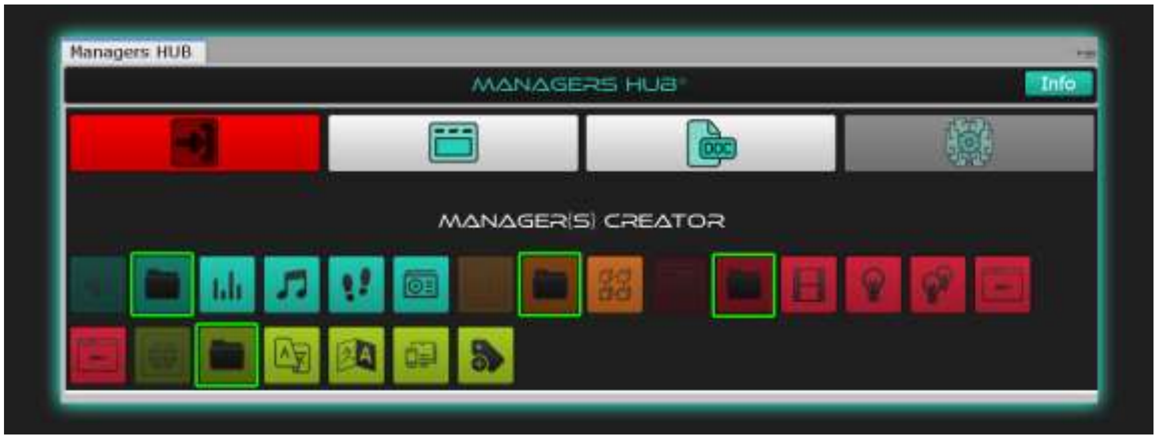


MANAGERS CREATOR:
The managers creator is the menu with all shortcut to create Managers, Data and access their folders.



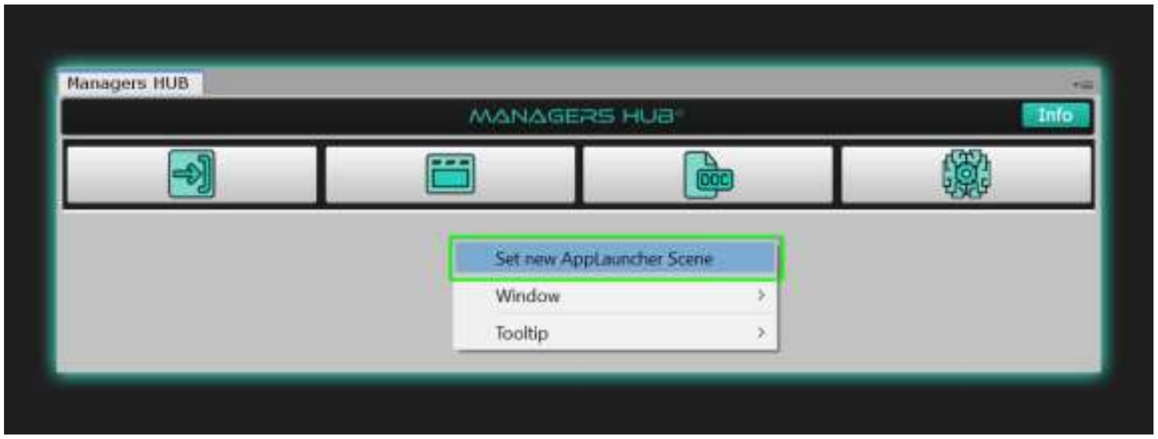
CREATORS

Those buttons are the Manager Creator. You need to be in the AppLauncher to enable those buttons. When you click on it, it will create a manager for you.



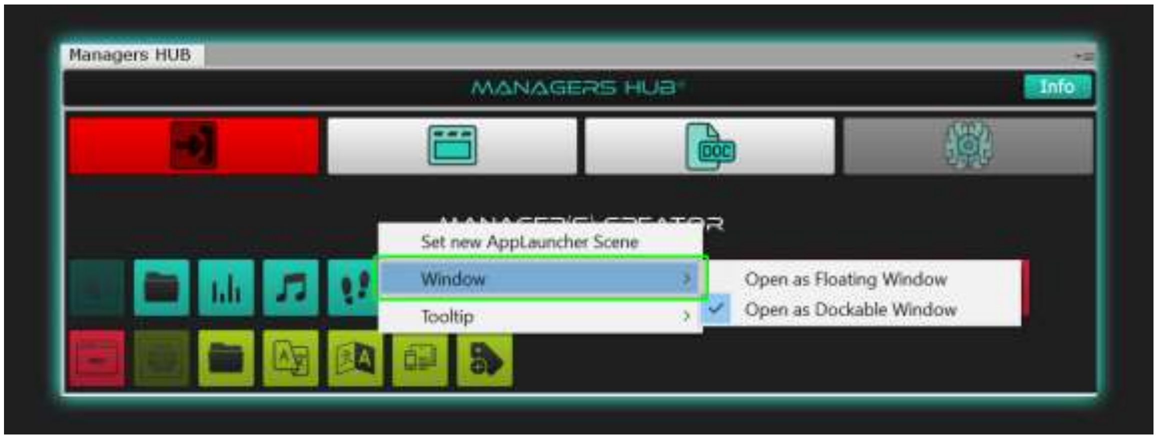
FOLDERS

Those buttons are the Manager Folder. When you click on them, Unity will select the folder in the inspector so you can access all data for a specific manager quickly.



SET NEW APPLAUNCHER SCENE

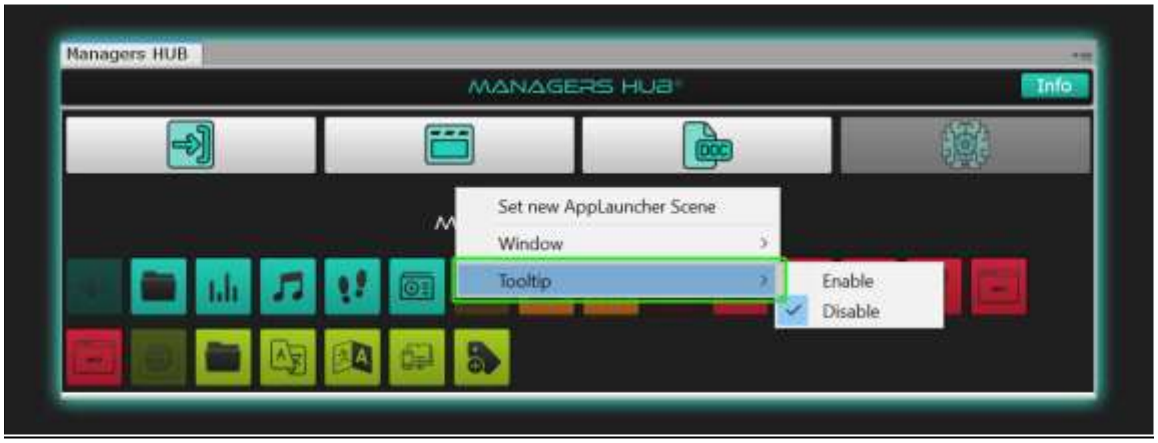
Right click on the HUB and select Set New AppLauncher Scene to set your own AppLauncher scene. By default, it takes the AppLauncher scene of the HUB.



WINDOWS

Right click on the HUB and select Window to have 2 options

Floating Window:	*Normal window that can be move anywhere on your screen.*
Dockable Window:	*Window that can be dock within the Unity inspector.*



TOOLTIP

Right click on the HUB and select ToolTip to have 2 options

Enable:	*Will show documentation directly on the HUB.*
Disable:	*Will hide documentation from the HUB.*

P.S. The 'Info' button in the right corner of the HUB do the same things.

AUDIO MANAGER

The audio manager is a very simple audio extension that will lets you resolve your audio needs in second! You can play music, sound, create a radio [like in racing game], make foot sound depend on the ground type and your movement, etc.

ENUMS use in the Audio Manager

eAudioM_GroundType:	*Contains all ground type for your games.*
eAudioM_FootSoundType:	*Contains all the movement for your foot sounds.*
eAudioM_FootSoundDir:	*Left or Right foot.*
eAudioM_AudioSelector:	*If your foot sound uses AudioData or AudioClip.*

MUSIC SYSTEM

The music system lets you play, stop, pause and loop music from anywhere in the game.

Setting(s)

NULL

Function(s)

Music_PlayOnce(AudioClip aClip) ->

Will play an audio clip once on the music audio source.

Music_Play() ->

Press play on the music audio source.

Music_Stop () ->

Press stop on the music audio source.

Music_Pause () ->

Press pause on the music audio source.

Music_SetClip (AudioClip aClip, bool aAutoPlay) ->

Set the new clip for within the music audio source with aClip and play it if aAutoPlay = true.

Music_SetLoop (bool aWantToLoop) ->

Set the music audio source loop option to true if aWantToLoop = true, set it to false if aWantToLoop = false.

SOUND SYSTEM

The sound system lets you play sound anywhere in the game with all setting already pre-set. You can play sound in easy mode or in perf mode (similar to the pool manager but with sound)

Setting(s)

[Amount of perf sound(s)]	*The amount of perf sounds to init when the scene starts.*

Function(s)

SoundEasy_Play (AudioData aSound, Vector3 aPosition) ->

> Instantiate a sound with aSound settings at the aPosition position, play it and then destroy it.

SoundEasy_PlayAndReturn (AudioData aSound, Vector3 aPosition) ->

> Instantiate a sound with aSound settings at the aPosition position, return the Transform of the sound to the caller, play it and then destroy it.

SoundEasy_PlayAsAChild (AudioData aSound, Vector3 aPosition, Transform aParent) ->

> Instantiate a sound with aSound settings at the aPosition position, set it as a child of the aParent transform, play it and then destroy it.

SoundPerf_Play (AudioData aSound, Vector3 aPosition) ->

> Pick a soundPrefab already instantiate from a list, set the new sound with aSound settings at the aPosition position, play it and then put it back in the list.

SoundPerf_PlayAndReturn (AudioData aSound, Vector3 aPosition) ->

> Pick a soundPrefab already instantiate from a list, set the new sound with aSound settings at the aPosition position, return the Transform of the sound to the caller, play it and then put it back in the list.

SoundPerf_PlayAsAChild (AudioData aSound, Vector3 aPosition, Transform aParent) ->

> Pick a soundPrefab already instantiate from a list, set the new sound with aSound settings at the aPosition position, set it as a child of the aParent transform, play it and then put it back in the list.

RADIO SYSTEM

The radio system lets you play music with multiple station that the player (or you) can change anytime in game. It can also display the music in play with a custom UI, like in racing game. Only set the Music Data for the station you want, and the manager will mix it for you.

Setting(s)

[Radio station(s)]	*All the radio stations you want in your game (radio data)*
[Radio display]	*The prefab UI that contains the script RadioDisplay.cs on it.*

Function(s)

Radio_ResumeMusic() ->

Resume the music that was playing last time we exit this station.

Radio_PauseMusic() ->

Pause the music that is playing right now.

Radio_NextMusic() ->

Will change for the next music in the current radio station.

Radio_PreviousMusic() ->

Will change for the previous music in the current radio station.

Radio_NextStation(bool aAutoResume) ->

Will change for the next radio station. If(aAutoResume), will play the music automatically.

Radio_PreviousStation(bool aAutoResume) ->

Will change for the previous radio station. If(aAutoResume), will play the music automatically.

Radio_SelectStation(RadioData aRadioStation, bool aAutoResume) ->

Will change for the selected radio station (aRadioStation). If(aAutoResume), will play music automatically.

FOOT SYSTEM

The foot sound system lets you play foot sound depend on your movement and on which type of ground you're moving on.

Setting(s)

[FootSound Data]	*The data that contains all the foot sounds.*
[How use]	*Select if you want the foot system to use Sound data or Audio clip. If you select audio clip, you will need to set the volume for each movement type (Walk, Run, Crouch, Crawl).*

Function(s)

FootSound_Play(Vector3 aPosition, eAudioM_GroundType aGroundType, eAudioM_FootSoundType aMovementType, eAudioM_FootSoundDir aFootDir) ->

Instantiate a foot sound at the position aPosition depend on which ground you are on aGroundType, on which movement you're doing aMovementType and if it's the left or right foot that hit the floor aFootDir.

GROUND SELECTOR

The ground selector is the component to add on the floor you're going to walk on.

Setting(s)

[Ground Type]	*The type of ground this floor is.*
[How use]	*Select the ground type and set the layer of this gameObject to the same as the one of FootSound Caller (see next script).*

FOOT SOUND CALLER

The foot sound caller is the component that need to be set on the same gameObject as the player animator.

Setting(s)

[Ground Layer]	*The layer that every ground that will use the ground selector.*
[Left foot transform]	*The transform of the left foot of the player, use to detect ground.*

[Right foot transform]	*The transform of the right foot of the player, use to detect ground.*
[Distance from ground]	*The max distance the foot of the player can check the ground from is position.*
[How use]	*Set the component on the same gameObject as the player animator and set the properties. Open an animation of this animator, add an event when the player foot touch the ground and select the right function (left or right foot).*

OPTION SYSTEM

The option system lets you divide all of your sound between categories. Its to be able to manage their volume by code.

Setting(s)

[Master Group]	*The Audio Mixer group that contains the master volume.*
[Music Group]	*The Audio Mixer group that contains the music volume.*
[SFX Group]	*The Audio Mixer group that contains the SFX volume.*
[Voice Group]	*The Audio Mixer group that contains the voice volume.*

Function(s)

Option_SetMasterGroupVolume(float aVolume) ->

Will change the volume of everything in the game with the new one aVolume.

Option_SetMusicGroupVolume(float aVolume) ->

Will change the music volume of the game with the new one aVolume.

Option_SetSFXGroupVolume(float aVolume) ->

Will change the sounds volume of the game with the new one aVolume.

Option_SetVoiceGroupVolume(float aVolume) ->

Will change the voices volume of the game with the new one aVolume.

AUDIO DATA

Audio data is the Scriptable Object that lets you set a sound and every setting already done to it, so when you want to play it, it way easier!

[1] You need to select if the sound is an audio clip or a MultiLanguage Data (useful for voice in different language).

[2] You need to set the audio (audio clip or data) and set the base volume for it.

[3] Choose if you want to activate advanced settings. If activate, you will now have the option to change Pitch, Spatial Blend(2D / 3D) and Stereo Pan (Left / Right)

[4] Choose if you want to activate delay option. If activate, you can set a delay that the sound needs to wait before he gets played.

[5] Choose if you want to activate the variant settings. If activate, you can set a variant for volume and pitch. Variant is a random gap that the manager will take from to set the sound settings. (WARNING: Will overlap Advanced Settings)

MUSIC DATA

Music data is the Scriptable Object that lets you set a music with all is information (song name, artist, album,) so he can be display to the player.

[1] You need to set the audio clip for this music and set is volume.

[2] You need to set is artist and song name and is album cover art. (for radio display, you can skip if you don’t use radio display).

FOOTSOUND DATA

FootSound data is the Scriptable Object that lets you set all sounds for every ground type and movement available in your game.

[1] You need to select if you want your data to use AudioData(with pre-set settings) or Audio clip.

[2] Select the ground you like to edit and the movement of your choice.

[3] Drop as many sounds you want for those choice make above. Those are the sounds that will be taking randomly when the player moves.

RADIO DATA

Radio data is the Scriptable Object that lets you set all the music inside a radio station and set is option.

[1] You need to enter a name for your radio station.

[2] Choose if you want your station to shuffle all the music inside when the game start.

[3] Drop as many MusicData you want for this radio station.

RADIO DISPLAY (SCRIPT)

Radio display is the script that we put on in the custom UI to show the radio information to the player. You need to have this script on it and set correctly to make it work.

Setting(s)

[Radio Display]	*The GameObject that contains all the radio display component.*
[Artist Name Display]	*The TextMeshPro_UGUI that will show the artist name on screen.*
[Song Name Display]	*The TextMeshPro_UGUI that will show the song name on screen.*
[Album Cover Display]	*The Image that will show the album cover on screen.*

POOL MANAGER

The pool manager is designed to increase performance when you need to instantiate a lot of objects. Your objects are instantiated at Start then stored in separate pools for further use. It can really come handy for breakables, or particle systems. All pooled objects are set within a custom interface to make it **easy** for you to use.

ENUMS use in the Pool Manager

NULL

BASE SYSTEM

The base system lets you instantiate pooled object with a Pool Data, already pre-set, from anywhere in the game. WARNING: Never destroy a gameObject from the pool, only disable it when you're done with it.

Setting(s)

NULL

Function(s)

InitPools (PoolData aPoolData) ->

Initialise a new pools object with all the objects inside the aPoolData.

ClearOlderPools () ->

Remove all object pools currently in game (don't destroy object, they will get destroy .

UseObjectFromPool (GameObject aPrefab, Vector3 aPosition, Quaternion aRotation) ->

Select the gameObject you ask for (aPrefab) in the pooled list, set is position at aPosition and is rotation at aRotation. Spawn it and return it to the caller.

UseObjectFromPool<T>(GameObject aPrefab, Vector3 aPosition, Quaternion aRotation) ->

Select the gameObject you ask for (aPrefab) in the pooled list, set is position at aPosition and is rotation at aRotation. Spawn it and return the component of type T from the gameObject.

POOL DATA

PoolData is the Scriptable Object that lets you set all the objects you want to be instantiate within the pool manager, and their options.

[1] You need to add as many prefabs as you want by clicking the button “Add a Prefab in the pool“.

[2] Select the prefab you want to edit.

[3] Set is prefab, the amount you want to instantiate in game, and if you want that the manager instantiates more copy if there’s to many in use at the same time o this prefab (Auto-Create when exceeds)

SCENE MANAGER

The scene manager make scene loading, unloading, and reloading easy. It gives you the options to make your loading screens as you wish, without any code… **simple and easy!**

ENUMS use in the Scene Manager

eSceneM_SceneCallType:	*If you load the scene with a path or a Scene Data.*
eSceneM _FadeDir:	*Fade in or Fade out.*
eSceneM _Music:	*The music option for the scene loading.*
eSceneM _LoadingScreen:	*The type of loading you want (Default, overwrite or fade).*
eSceneM _LoadingType:	*With tip or without tip.*
eSceneM _TipOption:	*Random or select tip.*
eSceneM _TipGroup:	*All the group available for tips when select randomly.*
eSceneM _TextType:	*Use for tip, string or Multilanguage data (TextData).*

BASE SYSTEM

The base system lets you load scene with different effects.

Setting(s)

[Default Loading Screen]	*The default loading screen that will be use when the Scene Data use the default option.*
[Min loading time]	*The minimum time the loading screen need to be show before is remove from screen. Even if the loading is done.*

Function(s)

CallLoadScene(SceneData aScene) ->

This function loads the next scene, you only need to give a SceneData (aScene) in argument, and the manager will load it for you with all your options previously set.

CallLoadScene(string aScenePath) ->

This function loads the next scene with basic option, you only need to set the path(aScenePath) of the scene you want (can be good for debug, but you SceneData if you can)

TIP & TRICK SYSTEM

Tip & trick lets you set a custom text within a loading screen. It can be chosen randomly or with a data to show the player the tip you want them to know!

Setting(s)

[Tips group]	*The group tips data that contains all the tips set in their groups (for the random option)*

Function(s)

NULL

FADE SYSTEM

The fade system lets you change scene with a fade (in/out) effect. When you choose this kind of loading, there's no "minimum loading time", when the loading screen is done, the screen fades out automatically.

Setting(s)

NULL

Function(s)

NULL

OVERWRITE SYSTEM

The overwrite system lets you have a custom loading screen for every scene, you just need to set a LoadingScreenData and the manager will create it for you. (Need this system for the "overwrite" option in the Scene Data.

Setting(s)

NULL

Function(s)

NULL

SCENE DATA

SceneData is the Scriptable Object that lets you set all option for the loading aspect of the scene.

---

[1] You need to set the Scene that you want to load. If you select a scene that isn't in the Project Setting of Unity, a warning will popup to tell you so, and ask you if you want to cancel your action or if you want the HUB to set it for you in the Project setting. That way, you won't get error because you try to load a scene that isn't set in the project setting.

---

[2] You will be asked to choose a Loading Screen option: **Default**, **Overwrite** and **Fade**.

Default:	*Load the default loading screen set on the Scene Manager inspector, useful for normal scene loading.*
Overwrite:	*Ask you a LoadingScreenData that will overwrite the default Loading Screen for this scene only. Can be useful if you want an epic loading screen for a boss battle for example.*
Fade:	*Will bypass the default loading screen, the minimum loading time set on the Scene Manager, and will create a fade effect (in/out) between the two scenes. When the scene is done loading, the fade out effect will start automatically. Can select the fade effect color.*

---

[3] If you selected Default or Overwrite option above, you will have the option if you want Tip & trick or not. If yes, two new options will be show to you: **Random** or **Select**.

Random:	*Select a group where the random tip will be grab, and the manager will select one of them randomly on play.*
Select:	*Select a Tip & Trick Data that have a tip on it, and the manager will show this tip to the player on play.*

---

[4] You can activate Manage Music. If you activate it, a message will be show to you that you need the Audio Manager to use this option and 3 options will be give to you.

With Music:	*Start a music (set an audio Clip) automatically when the scene is loading.*
Without Music:	*Do nothing, simple as that!*
Keep Old Music:	*Keep the music from the last scene running.*

---

[5] You can activate Manage Pool. If you activate it, a message will be show to you that you need the Pool Manager to use this option. Next step, you only need to set a Pool Data to spawn object when the scene is finished to load.

LOADING SCREEN DATA

Loading screen data is the Scriptable Object that lets you set a loading screen prefab. Those are used to be sure that you set a prefab with the right script on it for the Scene manager to create the loading screen.

---

[1] You need to set the prefab that contains the loading screen (need the script: loadingScreenObj.cs on it).

TIP DATA

Tip data are the Scriptable Objects that lets you set a tip & trick for your custom loading screen.

---

[1] You need to select if your tip is use with a string or with Multi-Language data (need language manager).

string:	*Just write your tip & trick.*

Multi-Language data:	*Set a Multi-Language data that contains your tip & trick in different languages.*

GROUP TIPS DATA

Group tips data are the Scriptable Objects that lets you set all your tip & trick by group and then set this data in the Scene Manager so he can select them randomly.

[1] Simply select a group you want to edit, and add all your Tip Data you want in. Do this for all wanted groups.

LOADING SCREEN OBJ (SCRIPT)

This script is used to set a new loading screen. Set this script on an object [normally with UI element on them] and set the element ask for. Create a prefab with it so you can set it in a Loading Screen Data.

[*Doesn't have a custom interface, so it easier for you to add element*]

[Tip Box]	*The box that contains the tip & trick (can be invisible)*
[Tip Txt Display]	*The TextMeshPro_UGUI element that will show the tip & trick.*

LANGUAGE MANAGER

Make your app/game internationally by translating it into multiple languages fast and easy. Supports all languages supported by Unity and all settings are done within a custom interface to make it **easy** for you to use!

ENUMS use in the Language Manager

eLanguageM_LanguageType:	*The type use in the MultiLanguage Data (Text, Audio or Image)*
eLanguageM _Platform:	*All platform available (Xbox one, Ps4, Switch, Stadia, Android, Xbox360, Xbox ProjectX, etc.)*

BASE SYSTEM

The base system lets you have multiple language for your game super easily. It works with Sound, String and sprite!

Setting(s)

[AvailableLanguageData]	*The data that lets you edit all the language available in your games. You can add or remove them within the Language Manager inspector. Those language will change the language for each MultiLanguage_Data, so set them at the start of the project if you can. Adding them won't do anything bad, removing some will remove all your text in each MultiLanguage_Data of the language removed.*
[Default Option Language]	*Choose if you want the manager to detect the device current language and set it as the default language or if you want a precis language for default option.*
[Default Language]	*The language that will be takin by default (if the platform language doesn't exist for your game and the above option was on Device Language)*
[Default Platform]	*Unity can't detect precis platform, so you need to set it by yourself. It's the default platform the game will start on. Change it for each build depend on platform.*
[Default SpriteAsset]	*The default SpriteAsset use for Tag system. The default one is use when the platform ask*

	doesn't exist or if you use Multiple TagCall option.

Function(s)

Language_SetCurrentLanguage(SystemLanguage aLanguage) ->

Set the new language of the game with aLanguage.

Language_ResetWithDefaultLanguage() ->

Will set the new language of the game with the default one set in the Language Manager (setting above).

Language_ResetWithDeviceLanguage() ->

Will set the new language of the game with the device one (if exist for your game).

TAG SYSTEM

The tag system work with **TextMeshPro** sprite. It will change the sprite that need to be show depend on the platform you're on. See TextMeshPro documentation to see how **SpriteAsset** work. Really useful for UI that show button for example.

(Xbox -> "Press A to start", PlayStation -> "Press X to start", Pc -> "Press Enter")

Setting(s)

[Tag Option]	*Choose if you want the tag system will use **multiple SpriteAsset** or **multiple TagCall. Multiple SpriteAsset** will change the entire SpriteAsset for the one of your current platforms, but you need to set all your SpriteAsset with all the same TagCall. If you select **multiple TagCall**, you will only have one SpriteAsset for all your platform, but you will need to set all theirs TagCall unique, so it won't mixt up.*
[SpriteSheetPlatformData]	*[Only need to be set if you use Multiple SpriteAsset] Set the SpriteSheetPlatformData that contains all your SpriteAsset for each platform you use in your game.*
[TagPlatformData]	*[Only need to be set if you use Multiple TagCall] Set the TagPlatformData that contains all the tag set for each platform you use in your game.*

Function(s)

Tag_SetCurrentDevice(eLanguageM_Platform aPlatform) ->

Set the new platform in use with aPlatform.

MULTI-LANGUAGE DATA

Multi-Language data are the Scriptable Objects that lets you set text, audio or image with different variant for all language available for your game.

---

[1] Choose if you want your data use Text, Audio, or Image.

---

[2] Select the language that you want to edit.

---

[3] Set the value of this language

[Text -> string] [Audio -> AudioClip] [Image -> Sprite]

AVAILABLE LANGUAGE DATA

AvailableLanguage data are used to select which language your game is available in. Can't edit directly this data, you need to use it in the Language Manager inspector. [READ ONLY]

TAG PLATFORM DATA

Tag platform data are used to set different tag version for one common tag Call, that way, you only need to call one tag in your text and the manager will change the tag depend on the current platform. Those tags are use with TextMeshPro SpriteAsset.

HOW IT WORKS

To call a tag into a string, you need to write: <tag=YourTagCall>

[1] Write down a Tag Call and click the button "Add Tag Call" to add this call into the data.

[2] Select the Tag Call that you want to edit (if empty, go back to step one)

[3] Just write down the tag depend on the platform.

Example:

Common Tag Call ->	*Interact*
Xbox One ->	*xboxOne_A*
PlayStation 4 ->	*ps4_X*

SPRITE SHEET DATA

Sprite sheet data are used to set different sprite sheet for each platform available in your game. When you use this option, you need to be sure that all sprite assets are set with the same tag call for each sprite, for all platform.

[1] Just drop a sprite asset for each platform available for your game.

TEXT READER TMPRO (SCRIPT)

This script is used on every TextMeshPro_UGUI that will use a Multi-Language data. The script will take the good text for the current language and convert the SpriteAsset tag for the good one depends on the current platform.

Setting(s)

[Multi-Language Data]	*The Multi-Language data that contains the text you want to show.*

Function(s)

SetNewText(**MultiLanguageData** aTextData) ->

Set a new Multi-Language data aTextData, so it would change text on screen.
