Finally, a framework for **all non-gameplay aspects** of your game production!

ARSENAL

01 INPUT

An advanced input system, easy to use, without any package dependency. Handles keyboard, mouse, gamepad and touch inputs. Any ingame action can be bound to a set of buttons/axes.

02 AUDIO

A series of utilities built on top of Unity's native AudioMixer system. Among these, Soundbank assets let you create and manage a variety of sound effects for your game.

03 TWEENS

The Tweener components give you total control over how any value can be animated. Custom Tweeners and Gauges can also be created, thanks to an easily expandable API.

04 THEMES

Reskinning your whole UI is now instantaneous: you can bind any element or group to a Theme asset, which is easy to edit and contains all relevant data for your content's appearance.

05 UI

All the most common components of a game's UI are available as highly customizable prefabs with all kinds of behaviours: panels, buttons, selectors, tab system, page system...

06 CURSOR

The main cursor (usually the player's mouse) can be fully customized, and also provides a seamless compatibility with joysticks or virtual sticks in addition to regular mouse movements.

07 LOCALIZATION

Store your game's texts in a CSV or TSV file, using the spreadsheet template included with the package, and Arsenal will handle the rest. Dynamic strings are also supported.

08 DIALOGUES

The package comes with a complete dialogue system built on top of the localization system. This includes branching, choices, events, speed manipulation and skipping dialogues.

09 CONTEXTS

Using a simple visual editor, you can dictate which game actions are locked or unlocked during different parts of the game, such as exploration, combat, menu or dialogues.

10 SAVE

Arsenal includes a full save system that can be customized at will. Game data is serialized in a JSON file, but savefiles are also readable and editable in the form of a ScriptableObject.

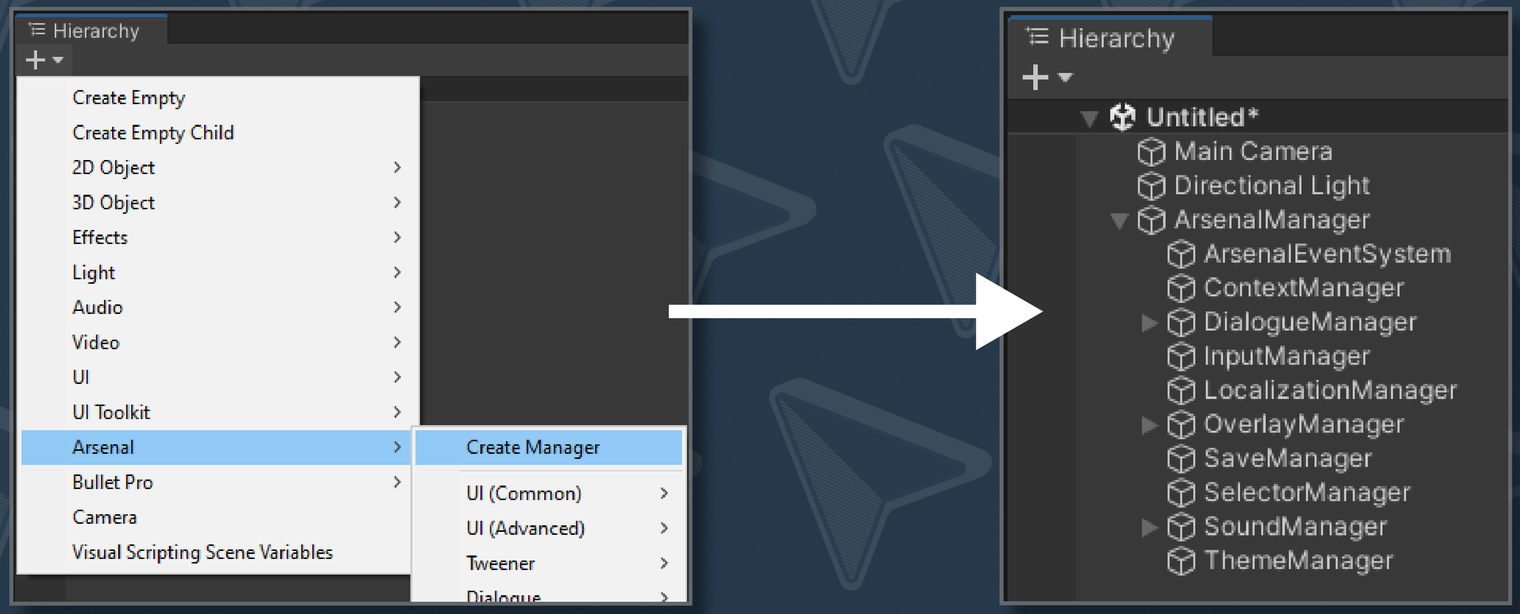
11 SETTINGS

As the sample use case for the save system, a complete functional "Settings" panel is available in the package. It gives control over fullscreen mode, resolution, language and audio volume.

12 AND MORE!

The list isn't even complete! Also, the package comes with an extensive documentation (user manual + scripting API reference), and support is available through the Discord community.

Step 1: create the Arsenal Manager in your scene.

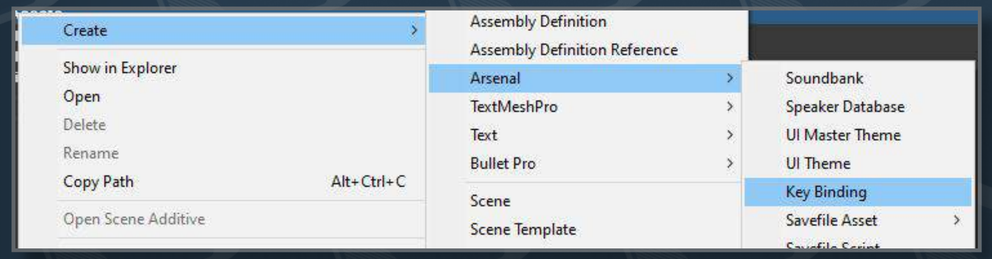


Step 2: make your game!

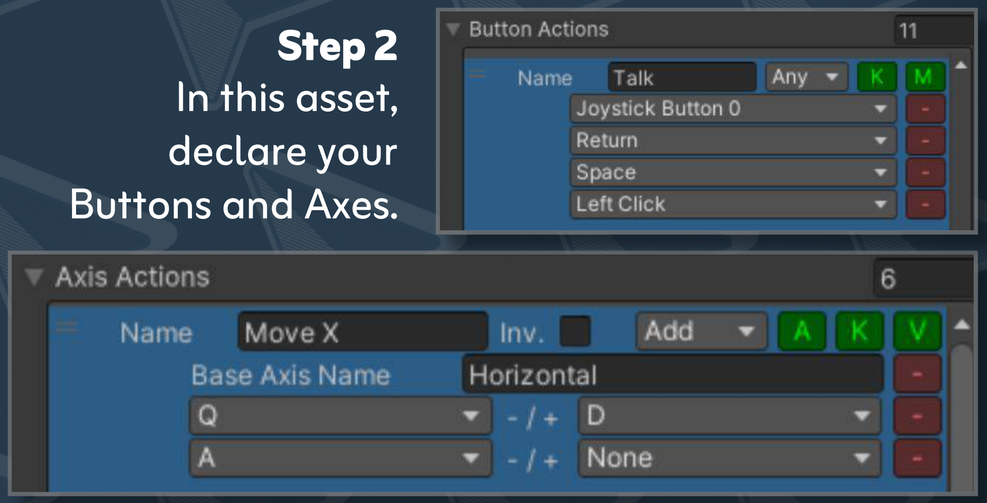
INPUT

ARSENAL

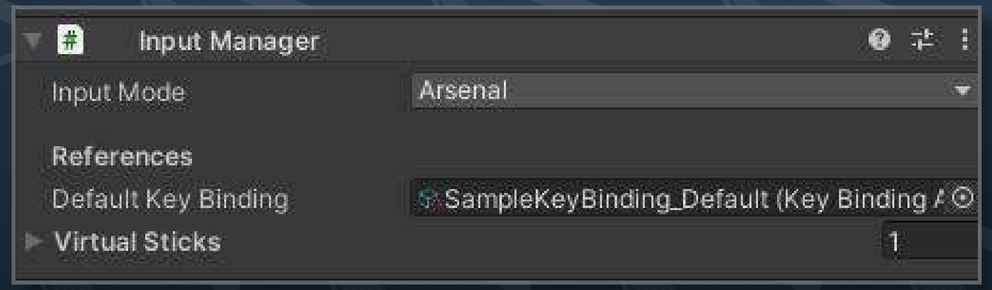
Step 1
In your Project folder, create a KeyBinding asset.



Step 2
In this asset, declare your Buttons and Axes.



Step 3
Drag the KeyBinding asset to the Input Manager.



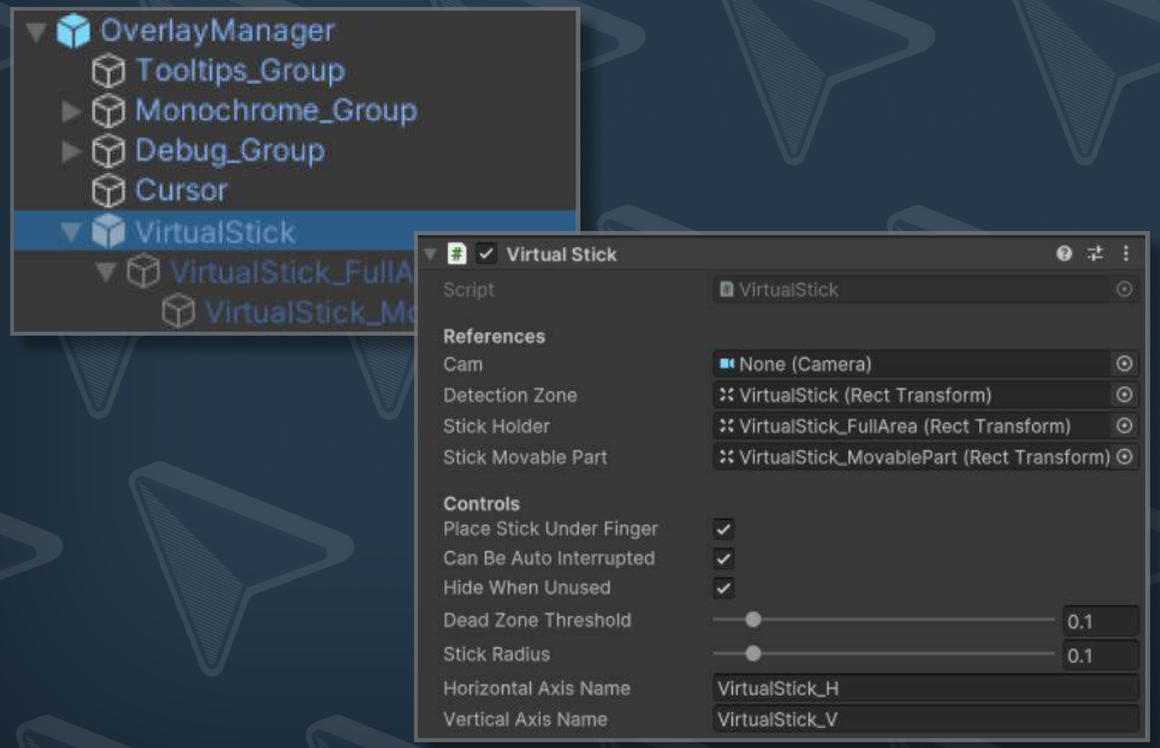
Step 4
You're all set!

```
PlayerInputManager input = PlayerInputManager.instance;

bool keyIsPressed       = input.GetButton("Talk");
bool keyBecamePressed   = input.GetButtonDown("Talk");
bool keyBecameReleased  = input.GetButtonUp("Talk");
float joystickValueX    = input.GetAxis("Move X");
```

INPUT (TOUCH)

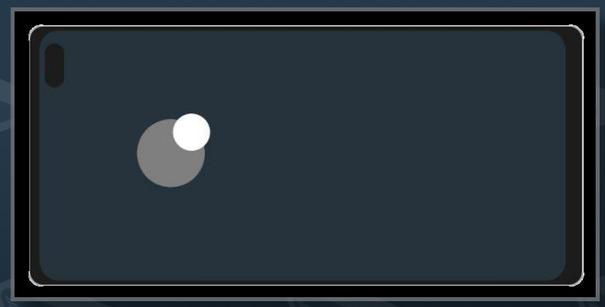
ARSENAL

Using touch input? Play with the Virtual Stick prefab!



Your KeyBinding assets also accept Virtual Sticks.

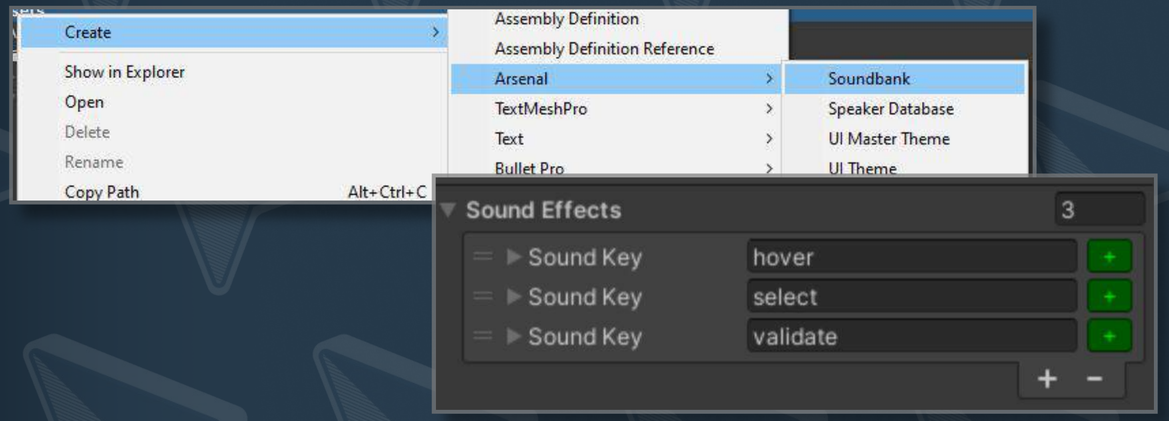
Results are immediate and customizable.



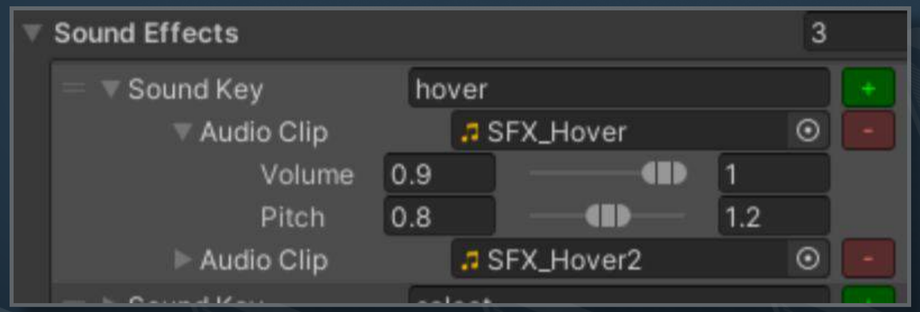
AUDIO

ARSENAL

Create your SFX in Soundbank assets...



...then give variations to any clip or value!



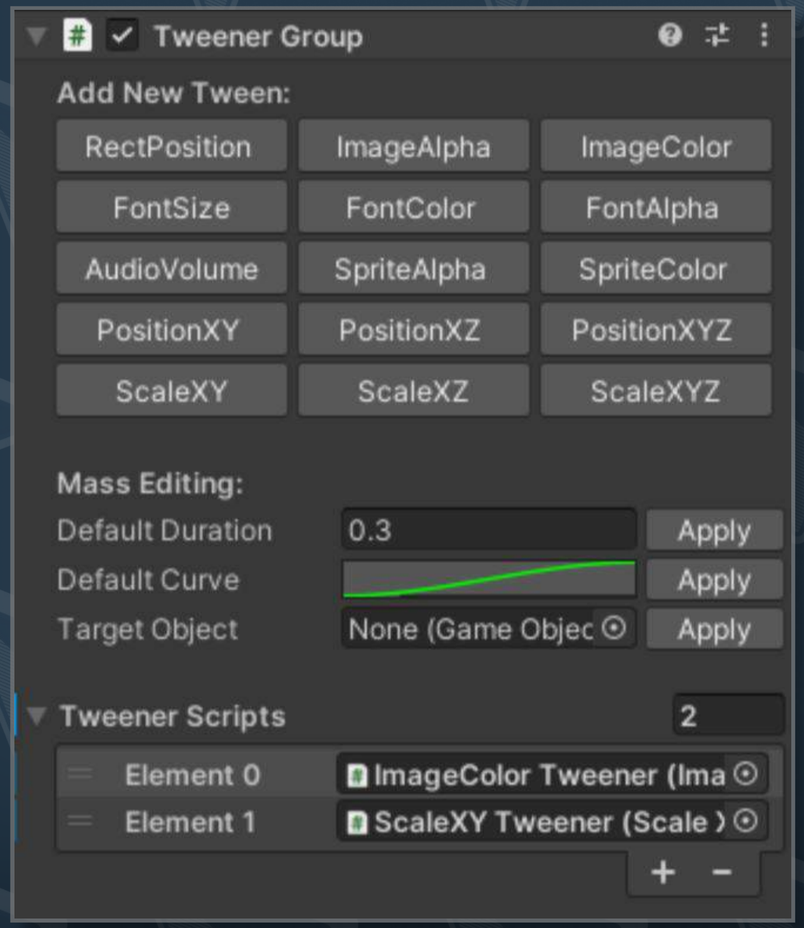
Manage your sounds from a one-line call:

```
SoundManager sounds = SoundManager.instance;

// Get or set volume for a specific group
float masterVolume = sounds.GetVolume("Master");
sounds.SetVolume("Music", 0.9f);

// Play a sound from a soundbank
sounds.PlaySFX("validate");

// Change the global background music
sounds.ChangeBGM(someAudioClip);
// Replay the previously played background music
sounds.RevertBGM();

// Pause/Unpause/Stop the BGM
sounds.PauseBGM();
sounds.UnPauseBGM();
sounds.StopBGM();
sounds.PlayBGM();
```

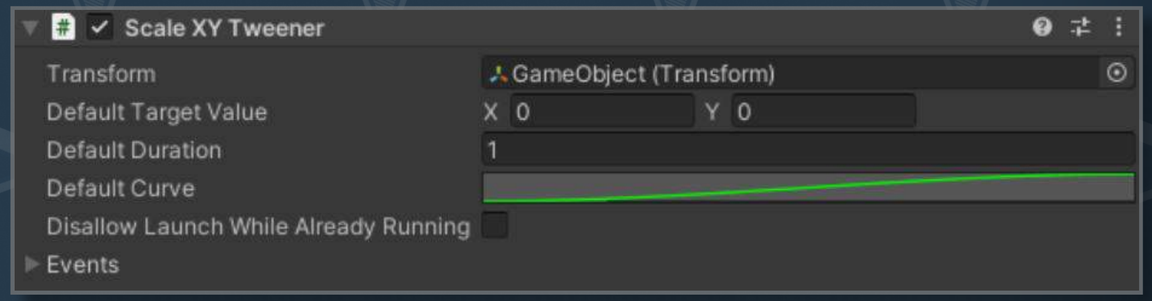
TWEENS

ARSENAL

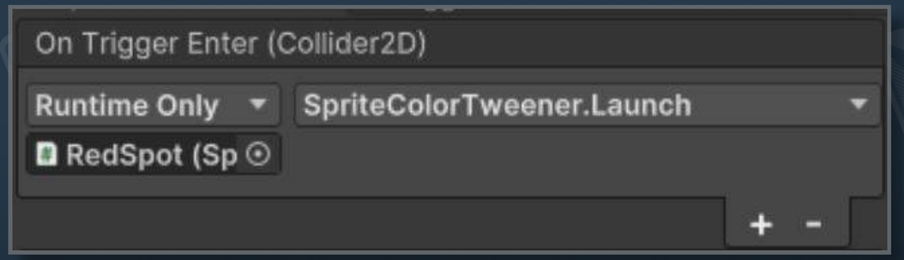
Perform Tweens on a large number of values!



One component per Tween, fully configurable



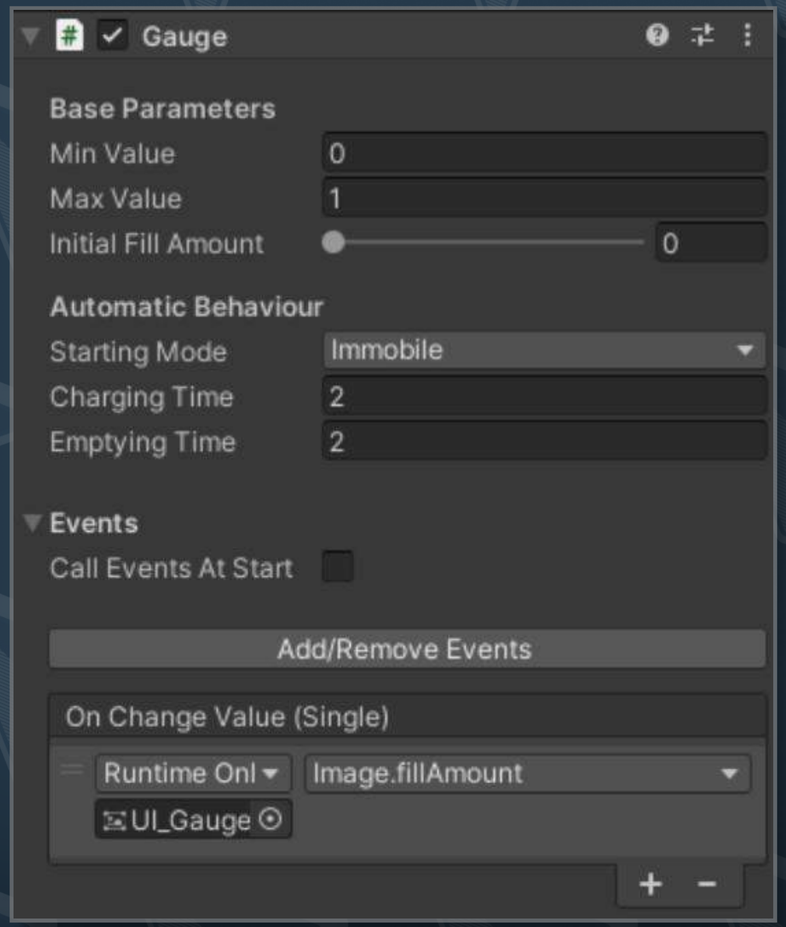
A single Launch() call performs the Tween



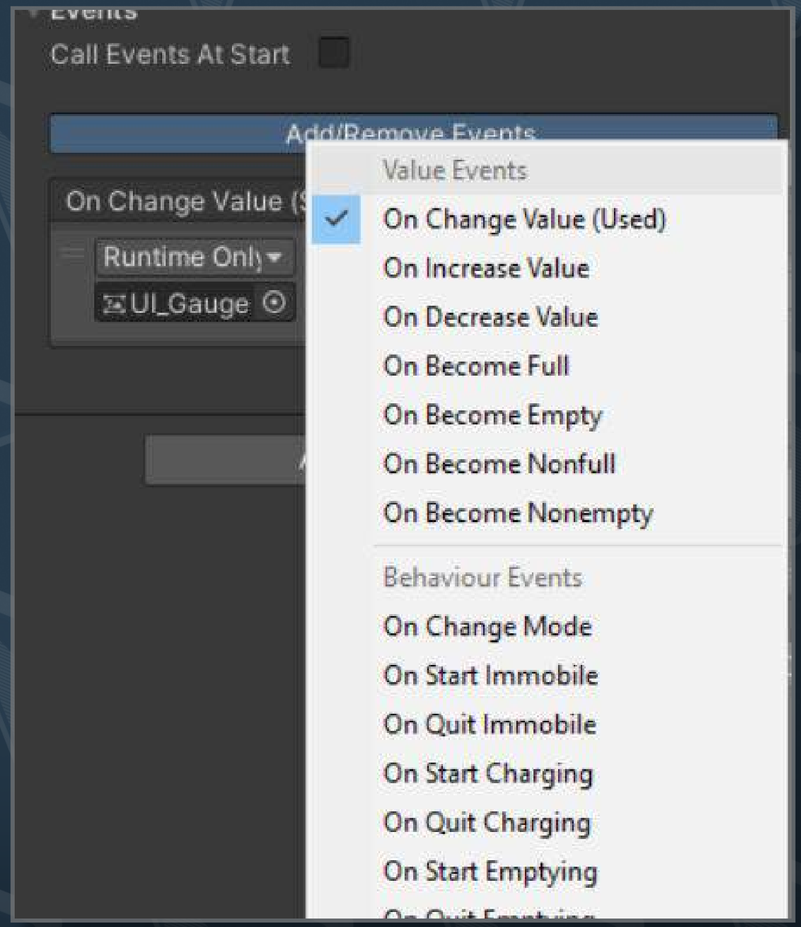
GAUGES

ARSENAL

Any element can implement a Gauge logic:



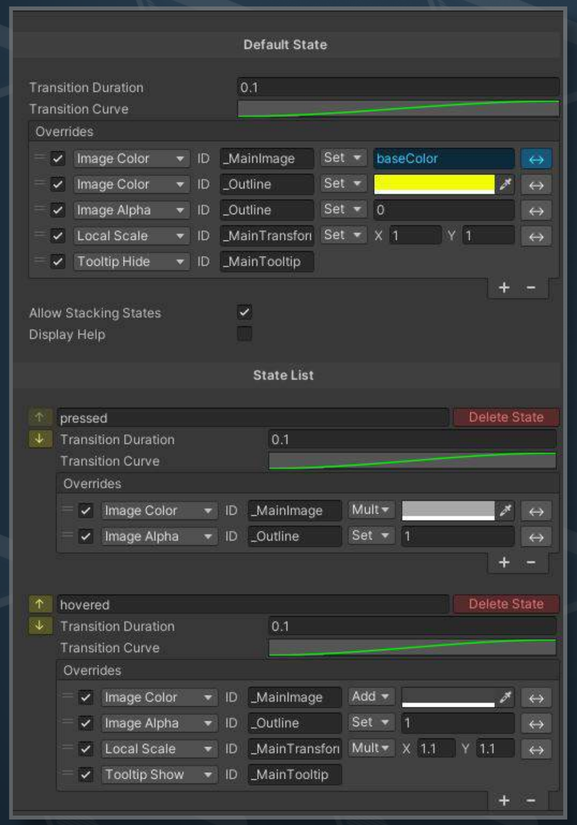
A Gauge can access several UnityEvents for flexibility.



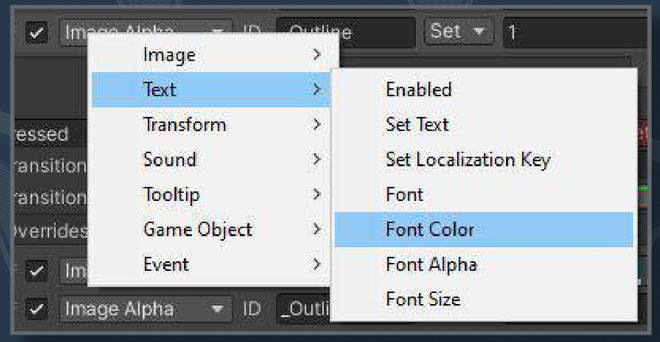
THEMES

ARSENAL

All UI styles and behaviours, in one single Theme asset.

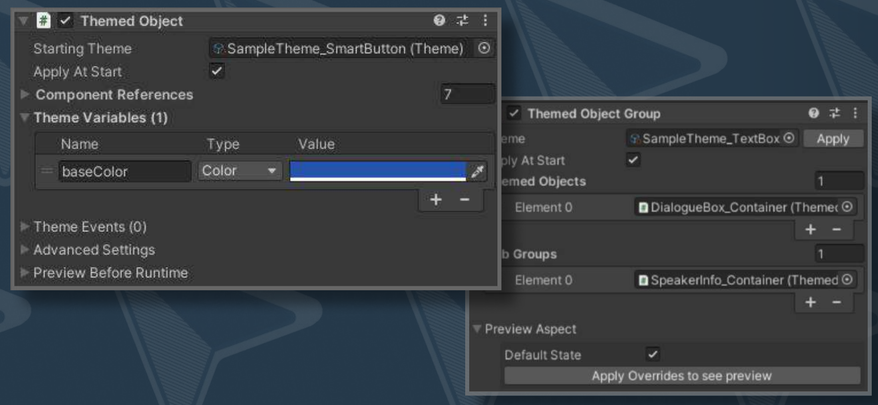


Lots of parameters available!



Play with logic operators:
“and, or, xor, not”

Binding to an object (or group!)
is done with a single component.

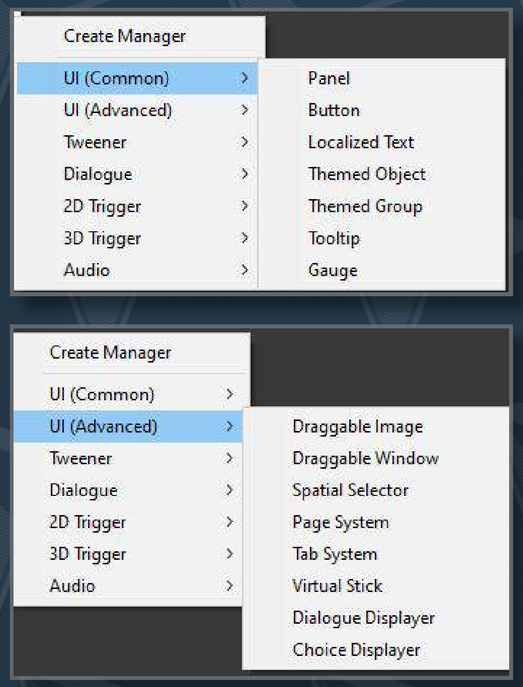


Easily enable/disable states at runtime,
either via code or event:

```
ThemedObject obj = GetComponent<ThemedObject>();

obj.AddState("hovered");
obj.RemoveState("hovered");
```

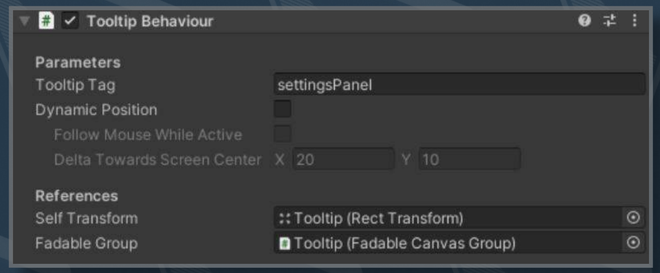
UI PRESETS

ARSENAL



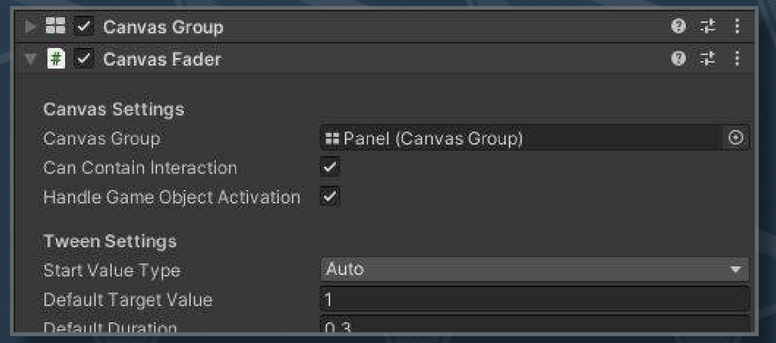
Tooltip:

Can follow cursor, and/or position itself dynamically.



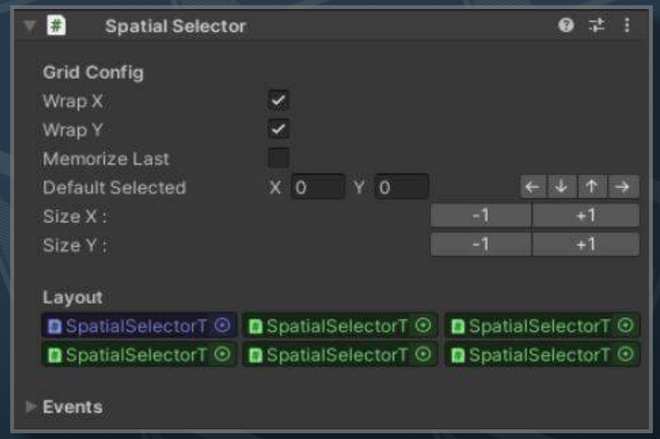
Panel:

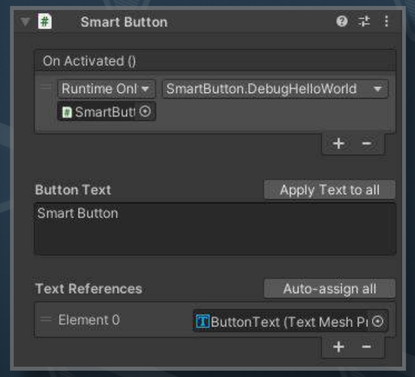
A simple panel with a CanvasFader, that can FadeIn() or FadeOut().



Spatial Selector:

A layout of buttons. Can accept navigation in four directions.





Smart Button:

A button with a fully, quickly customizable behaviour.
Highly reliant on UnityEvents.

LOCALIZATION

ARSENAL

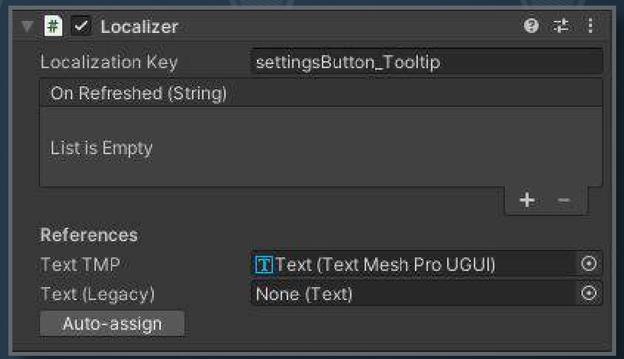
Copy the Google Sheet template (click here), fill it and export it as a TSV file.

	A	B	C
1	Key	FR	EN
2		Home screen	
3	settingsButton	Paramètres	Settings
4	settingsButton_Tooltip	Accéder à l'écran des paramètres.	Open the Settings panel.
5	dialogueButton	Lancer un dialogue	Launch Dialogue
6	dialogueButton_Tooltip	Pour tester les fonctionnalités de dialogue.	For testing dialogue-related features.
7		Dialogue displayer	
8	dialogueSkipInstruction	Maintenir Retour pour sauter le dialogue.	Hold the Back button to skip dialogue.
9		Settings panel	
10	settingsResolution	Résolution	Resolution
11	settingsFullscreen	Plein écran	Fullscreen
12	settingsLanguage	Langue	Language

Give this file to the LocalizationManager.

Give a Localizer Component to any element carrying text.

You can process dynamic text with the UnityEvent.



Protip: Chunks split your work into smaller files!

	A	B
1	Chunk_1	FR
2		
3	settingsButton	

```
LocalizationManager loca = LocalizationManager.instance;

loca.LoadChunk("Chunk_1"); // also unloads previous chunks
loca.LoadChunkAdditive("Chunk_1"); // doesn't unload previous chunks
loca.UnloadAllChunks();
```

Change language in a single method call:

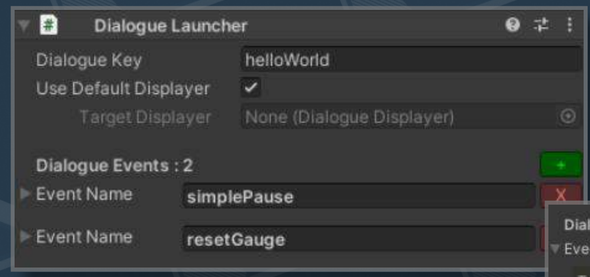
```
LocalizationManager loca = LocalizationManager.instance;

loca.SetLanguage("EN");
```

DIALOGUES

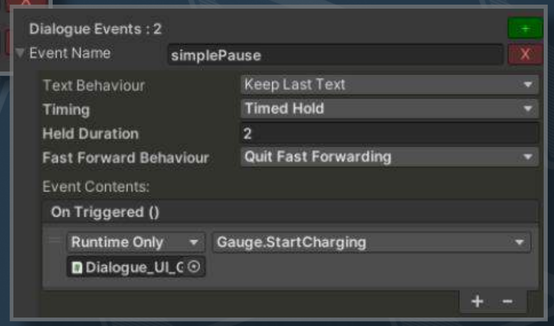
ARSENAL

Work with localized TSV files: [template here]

Chunk_1	Type	Portrait name (can be NONE), Event name, Option Key, or Redirection Key	FR	EN
helloWorld	Line	speaker_Alice	Bonjour ! Je me prénomme Alice.	Hi! My name is Alice.
	Line	speaker_Bob	Enchanté ! Je suis flob.	Nice to meet you, I'm flob!
	Line	speaker_Alice	Je peux parler en utilisant les balises de TextMeshPro.	I can talk using rich text from TextMeshPro.
	Line		...et avoir des répliques de taille variable.	Our dialogues are fully localized, and each language can use a different number of sentences.
	Line		Nos dialogues sont localisés, et peuvent s'étaler sur un nombre différent de phrases selon la langue.	
	Line	speaker_Bob	Incroyable ! Mais, qu'en est-il des autres fonctionnalités ?	Fantastic! But, a good dialogue isn't just text. What about the rest?
	Line		A part afficher du texte, dans un dialogue, il peut se passer plein de choses.	
	Line	speaker_Alice	C'est une très bonne question ! Nos dialogues peuvent invoquer des UnityEvents.	That's a very good question! Our dialogues can be linked to UnityEvents.
	Line		Ces Events sont définis dans le composant DialogueLauncher.	These events are defined in the DialogueLauncher component.
	Line		Exemple simple : marquons une pause de deux secondes.	As a simple example, let's pause for two seconds.
	Event	simplePause		
	Line	speaker_Bob	...J'ai vu une jauge se remplir !	...I just saw a gauge fill up!
	Line	speaker_Alice	Exactement ! Et ça n'est pas tout : les textes peuvent incorporer des informations dynamiques.	Exactly! And wait, there's more: texts can contain dynamic information.
	Line		Par exemple, cette scène de test a été lancée il y a déjà 5time secondes.	For instance, this test scene has been launched exactly 5time seconds ag
	Line		Oh, et il y a autre chose d'important, que l'on appelle les Player Choices.	Oh, and we have something else named the Player Choices.
	Line		Vous allez comprendre, jouons à un jeu simple. Promis, je ne triche pas ! Que choisissez-vous?	You will understand - let's play a short game. I won't cheat, I swear! Now, what do you choose?
	Option	goToRock	Pierre	Rock
	Option	goToPaper	Papier	Paper
	Option	goToScissors	Ciseaux	Scissors
goToRock	Line	none	Vous fermez le poing pour imiter la forme d'un rocher.	You show your fist in order to mimic a rock's shape.
	Line		La main d'Alice, à l'inverse, est tendue.	Alice chose to lay her fingers flat.
	Line	speaker_Alice	Aha, j'ai choisi Papier, je gagne !	Yay, I picked Paper, so I win!
	Redirection	afterGame		
goToPaper	Line	none	Vous présentez votre main tendue pour signifier une feuille de papier.	You present your hand horizontally, like a sheet of paper.
	Line		Alice semble avoir fait la même chose.	Alice seems to have done the same.



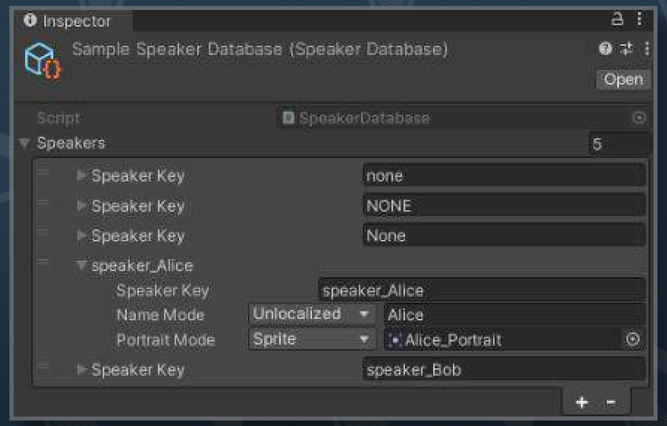
For starting a dialogue, use the DialogueLauncher component.



Dialogues can contain events, and you control when/how to play them.

Dialogues are shown in a full, preconfigured, highly customizable canvas.

Includes player choice UI, for dialogue branching.



SpeakerDatabase:

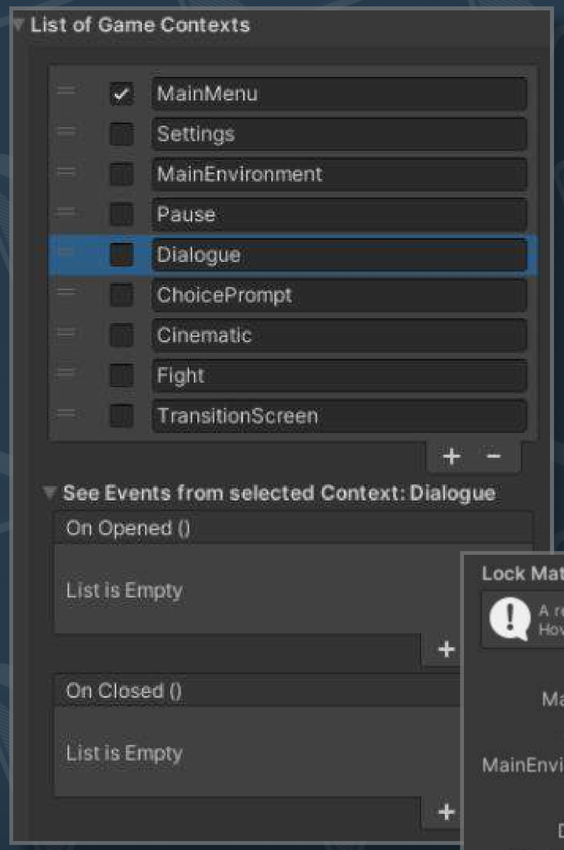
This asset handles names and portraits.

Supports any format for portrait displaying, including animations.

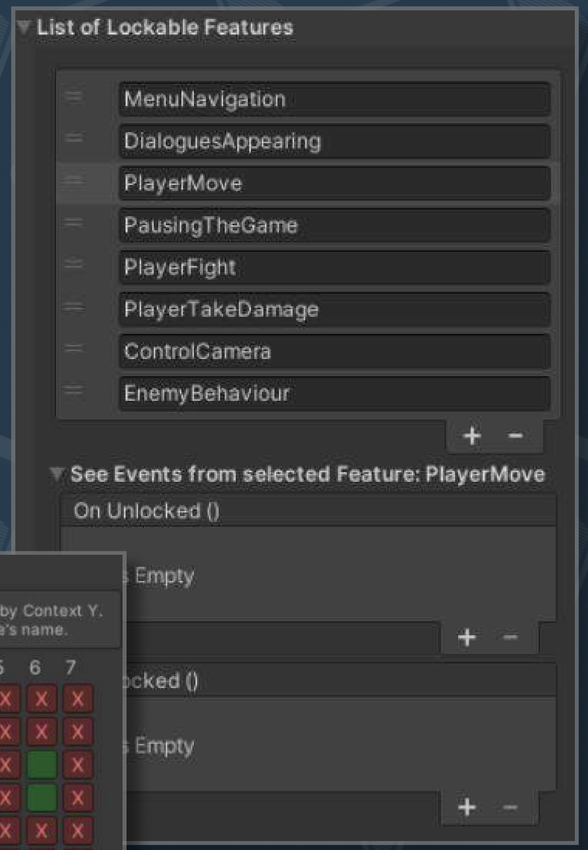
CONTEXTS

ARSENAL

At any moment in your game, any number of Game Contexts can be active at once.



This is the Context Manager.

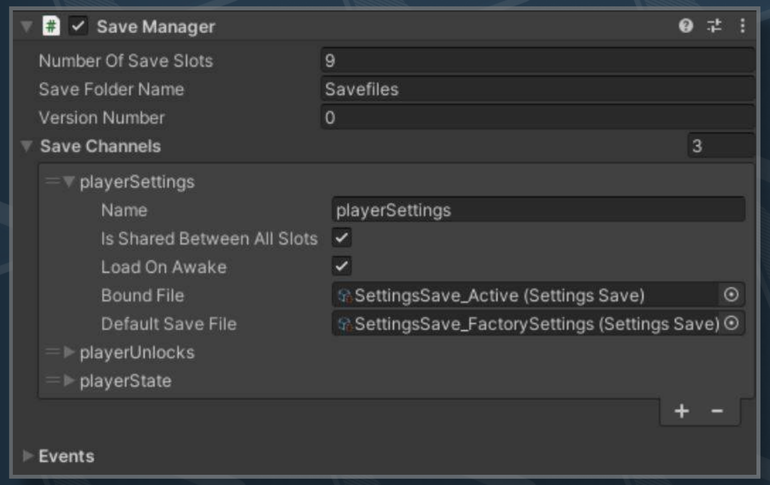


A Feature is “locked” if at least one active Context prohibits its use.

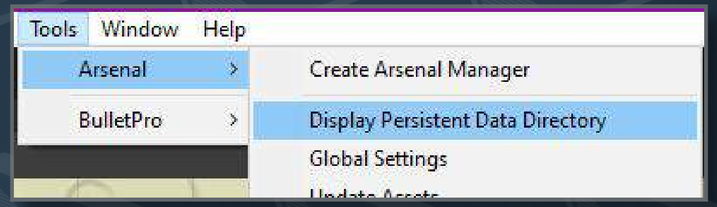
The Lock Matrix centralizes this logic, thus making transitions easier.

SAVE SYSTEM

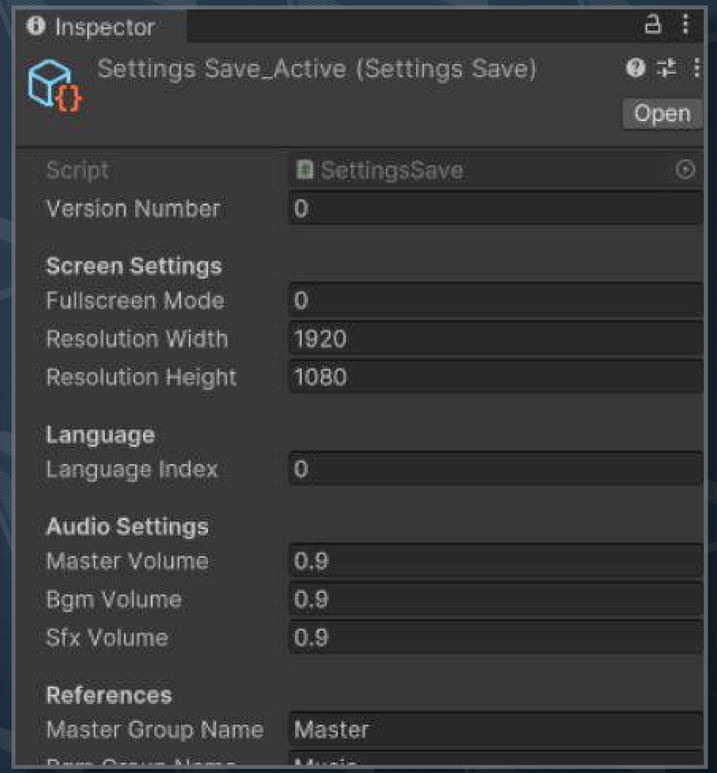
ARSENAL



Split saved data into “channels”, for an easier management.



Your save files exist not only as JSON files, but also as editable assets!



Save/load the game, with simple one-liners.

Also works across multiple save slots.

```
SaveManager save = SaveManager.instance;

// Slot management:
save.DuplicateSlot(3, 5); // copies contents of slot 3 into slot 5
save.DeleteSlot(4); // erases contents of slot 4 (all channels)
save.SwitchSlot(2); // sets active slot as slot 2
// subsequent save/load operations will affect slot 2, except for "shared" channels

// Save/load routine:
string channelName = "playerProgress";
save.LoadGame(channelName); // loads/resets to last save performed in current slot
save.SaveGame(channelName); // saves game (for this channel) in current slot
save.ResetToDefault(channelName); // reset to factory settings
save.EraseSavefile(channelName); // destroys json file for channelName at current slot

// Fake gamestate for debug purposes (doesn't interact with json)
save.LoadDebugFile(channelName, debugFile);
```