Grokit Core AI Documentation PDF

GroKit Core AI Module is a comprehensive platform designed for integrating Generative AI into Unity applications. It supports Text-to-Speech (TTS), Speech-to-Text (STT), and Large Language Models (LLMs) with Vision capabilities. Compatible across multiple platforms, the AI Module offers flexibility and extensibility, allowing developers to mix and match AI providers or incorporate their custom modules. Key features include custom editors, debugging tools, and advanced functionalities like keyword detection and AI response generation, making it an ideal choice for enhancing your application's interactivity and intelligence.

This is a standalone module that is also compatible with GroKit Core.

The AI Module is compatible with all platforms with the exception of Speech-to-Text on WebGL that will require a custom WebGL Microphone.

This module requires third party API keys to function

Online Documentation:

https://3lbxr.notion.site/GroKit-Core-AI-Core-03d04439efb142879dccfbc791e7dd9d

Support Discord

Please join the discord for support or updates regarding AICore

https://discord.gg/nMj2HpgYn7

Learn more about GroKit Core

https://3lbxr.com/grokit-core/

Features:

- Text To Speech
- Speech To Text
- LLM with Vision
- Image Generation
- Custom Editors
- Debugging Tools
- Encryption

Supported 3rd Party Platforms:

Gemini

OpenAI and Assistants

ElevenLabs

Play.HT

Meta Voice SDK

Getting Started

Before getting started, choose what services and make sure you have the API keys available. How to find each of their API keys are in the links below.

⛈️ OpenAI

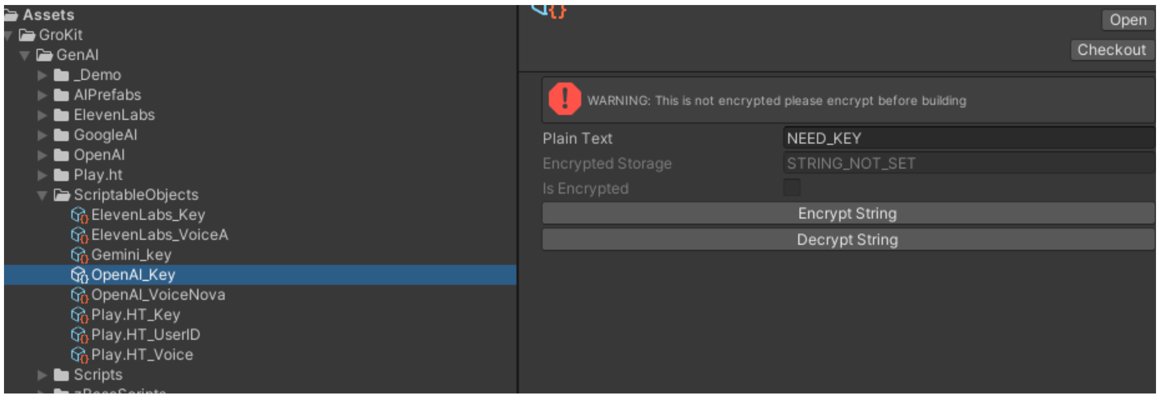
🔊 ElevenLabs

♊ Gemini

🏓 PlayHT

♾️ Meta - Voice SDK

APIs keys can be set via scriptable object or as a simple string. All of the prefabs in AI prefabs are set with the API keys. If you are making a build make sure you Encrypt your Strings



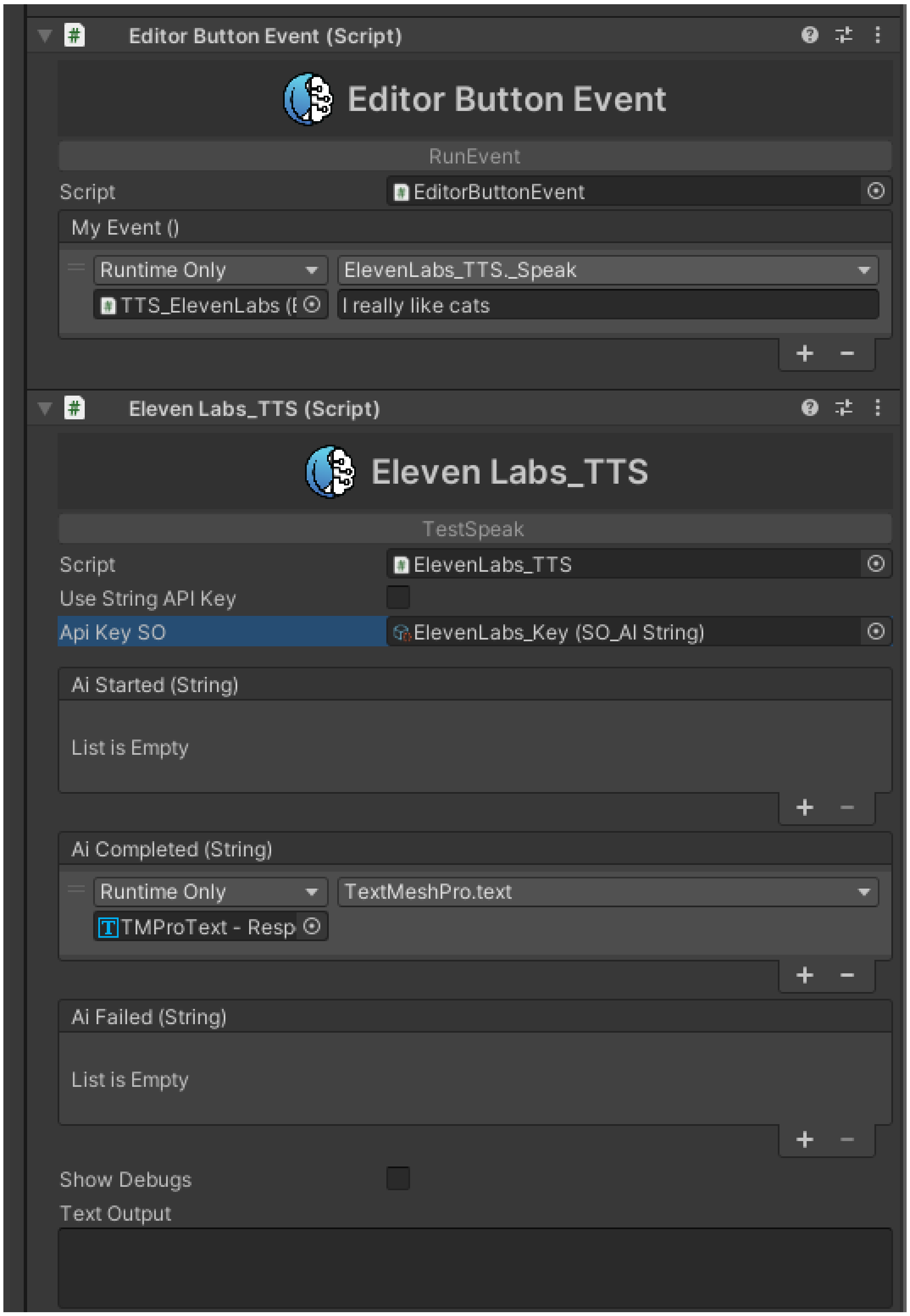
WARNING: This is not encrypted please encrypt before building

Plain Text	NEED_KEY
Encrypted Storage	STRING_NOT_SET
Is Encrypted	

Testing

Each prefab has an Editor Button Event component. This can be used to test quickly. Most components also have test buttons on the top of them become active in playmode.

This example below with ElevenLabs_TTS contains TestSpeak.


Editor Button Event (Script)

Editor Button Event

RunEvent

Script: EditorButtonEvent

My Event ()

Runtime Only | ElevenLabs_TTS._Speak

TTS_ElevenLabs (E | I really like cats

Eleven Labs_TTS (Script)

Eleven Labs_TTS

TestSpeak

Script: ElevenLabs_TTS

Use String API Key: ☐

Api Key SO: ElevenLabs_Key (SO_AI String)

Ai Started (String)

List is Empty

Ai Completed (String)

Runtime Only | TextMeshPro.text

TMProText - Resp

Ai Failed (String)

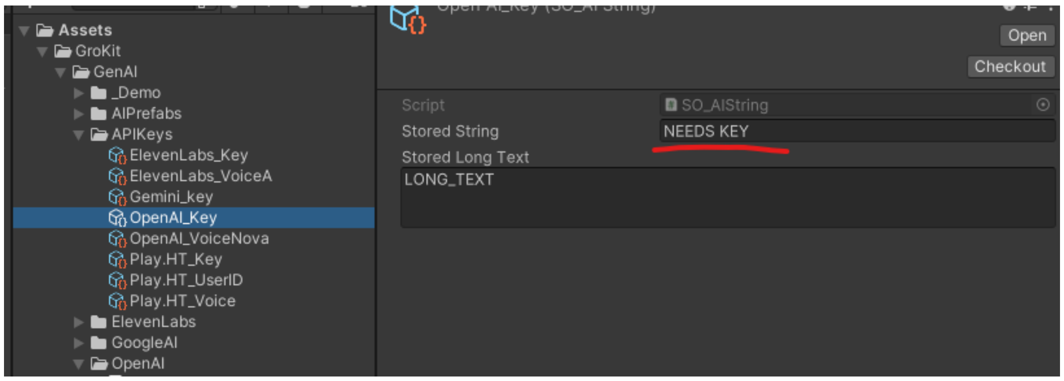
List is Empty

Show Debugs: ☐

Text Output

Running Demo Scenes

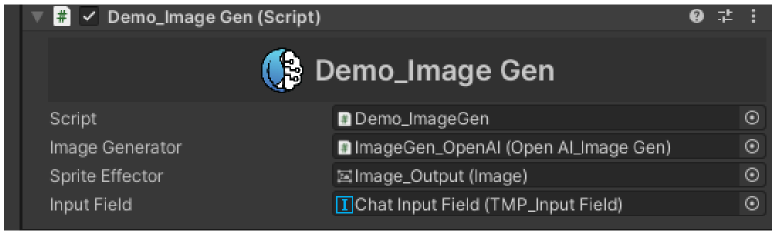
💡 All demo scenes use OpenAI to use them quickly set your API key at this scriptable object.



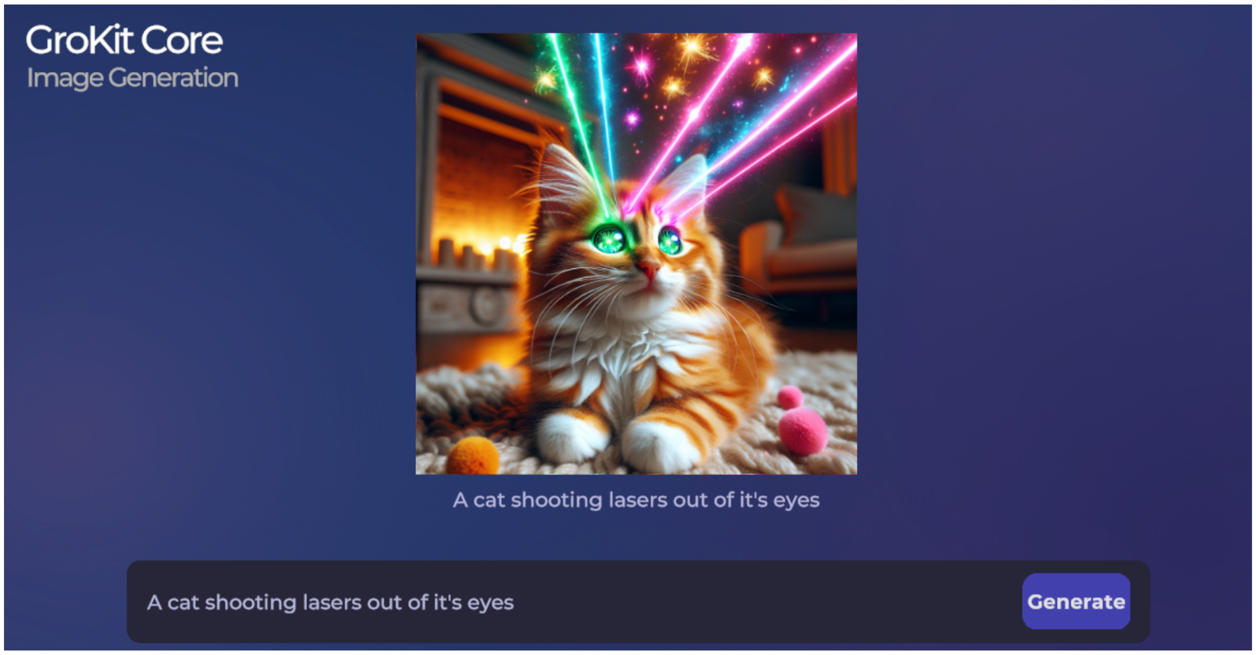
Script SO_AIString
Stored String NEEDS KEY
Stored Long Text
LONG_TEXT

Demo Image Generation

Example uses Dalle for Image generation, Image generation can take upwards of 10 seconds to generate and download the image.

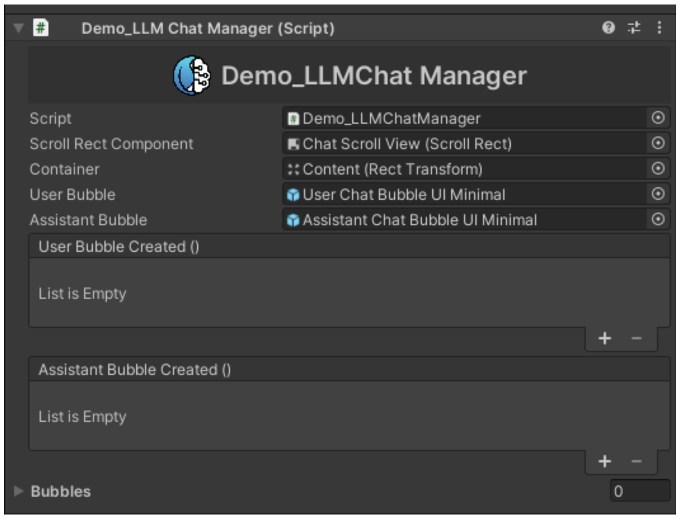


Script	Demo_ImageGen
Image Generator	ImageGen_OpenAI (Open AI_Image Gen)
Sprite Effector	Image_Output (Image)
Input Field	Chat Input Field (TMP_Input Field)


GroKit Core
Image Generation
A cat shooting lasers out of it's eyes
A cat shooting lasers out of it's eyes
Generate

OpenAI Mobile Chat

This demo is for making a mobile chat app like ChatGPT set your game windows to 16.9 to see the UI correctly. The Chat manager manages the bubbles for both the user and the assistant.



Demo_LLM Chat Manager (Script)

Demo_LLMChat Manager

Script	Demo_LLMChatManager
Scroll Rect Component	Chat Scroll View (Scroll Rect)
Container	Content (Rect Transform)
User Bubble	User Chat Bubble UI Minimal
Assistant Bubble	Assistant Chat Bubble UI Minimal

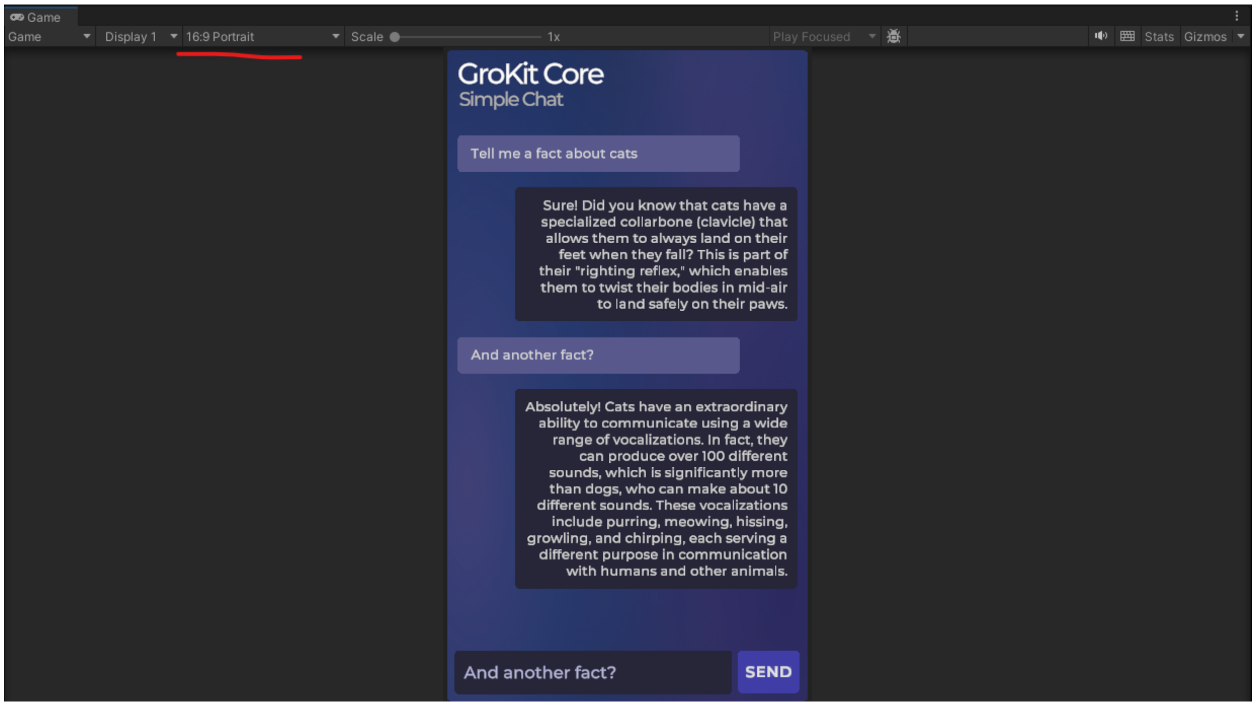
User Bubble Created ()

List is Empty

Assistant Bubble Created ()

List is Empty

Bubbles 0



Tell me a fact about cats

Sure! Did you know that cats have a specialized collarbone (clavicle) that allows them to always land on their feet when they fall? This is part of their "righting reflex," which enables them to twist their bodies in mid-air to land safely on their paws.

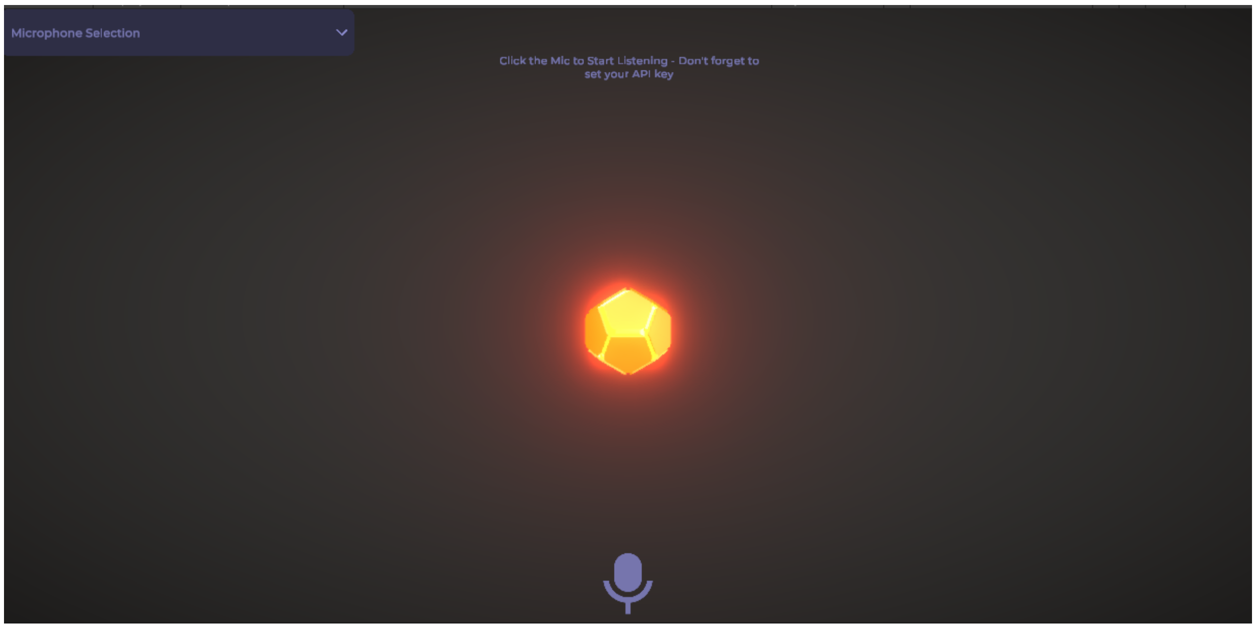
And another fact?

Absolutely! Cats have an extraordinary ability to communicate using a wide range of vocalizations. In fact, they can produce over 100 different sounds, which is significantly more than dogs, who can make about 10 different sounds. These vocalizations include purring, meowing, hissing, growling, and chirping, each serving a different purpose in communication with humans and other animals.

And another fact?

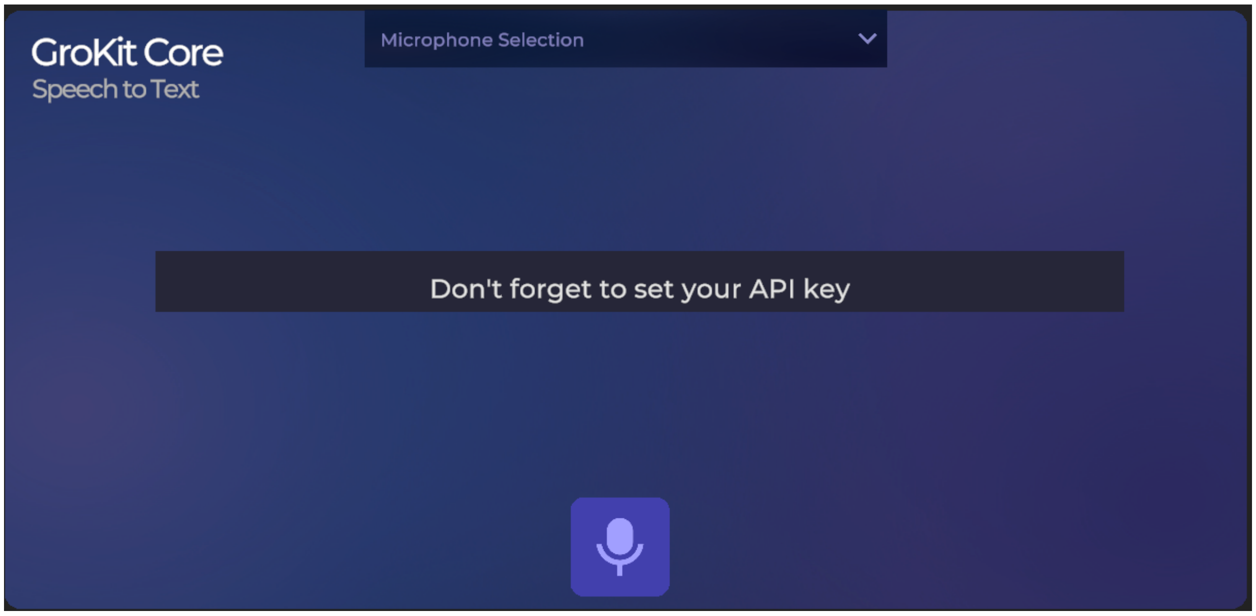
OpenAI Responder

This creates a Siri like experience where a user speaks and an LLM and then speaks back to them. Input devices can be set in the upper right hand corner. Click the mic button to start speaking


Microphone Selection
Click the Mic to Start Listening - Don't forget to set your API key

STT

Am example of Speech to text using OpenAI's whisper API. Click the mic and start speaking



TTS

This is an example of Text to Speech and the ability to save text as a file.

Folder:

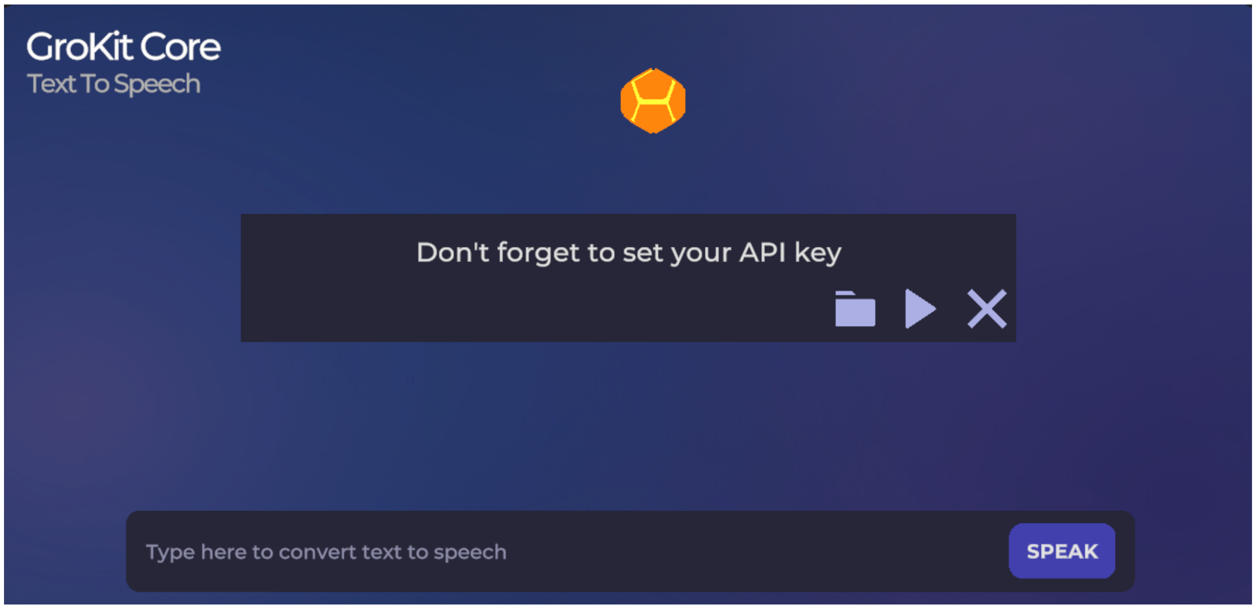
This saves the text in the bubble as a .wav file

Play:

Replays the the text in the bubble.

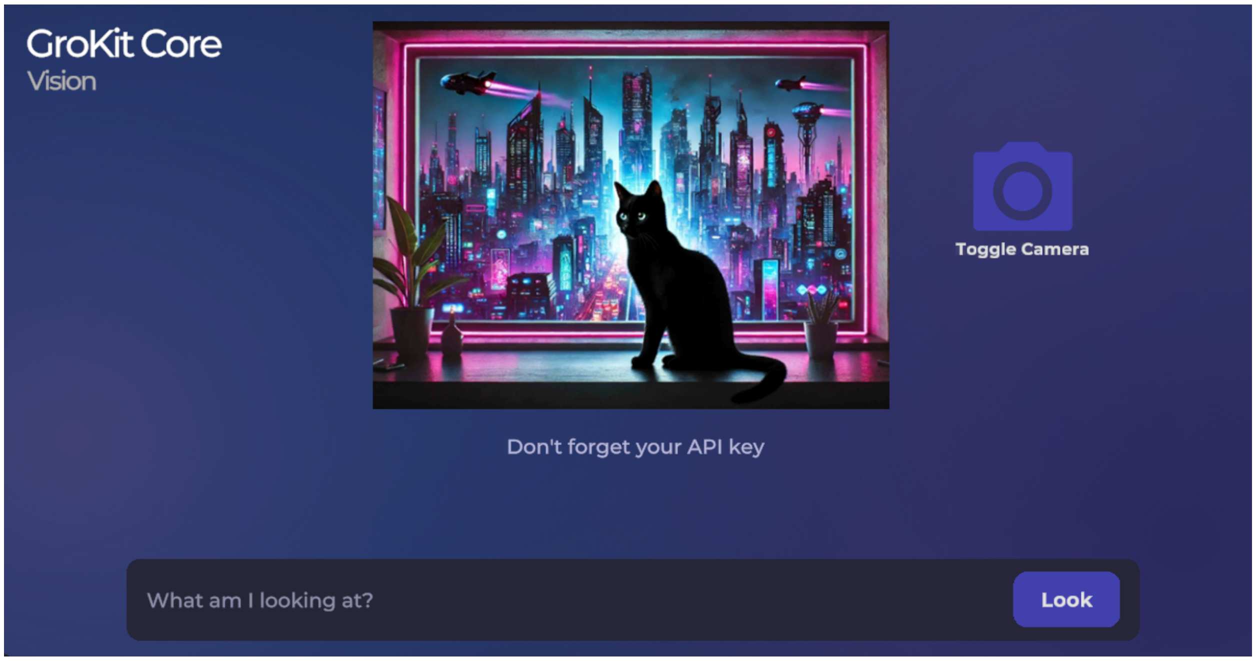
X:

Stops the audio source from speaking


GroKit Core
Text To Speech
Don't forget to set your API key
Type here to convert text to speech
SPEAK

Vision

Vision demo uses gpt-4o multimodal capabilities to look at an image. It will default to this cat, clicking toggle camera will activate your webcam and take a picture submitting it to the LLM when look is clicked



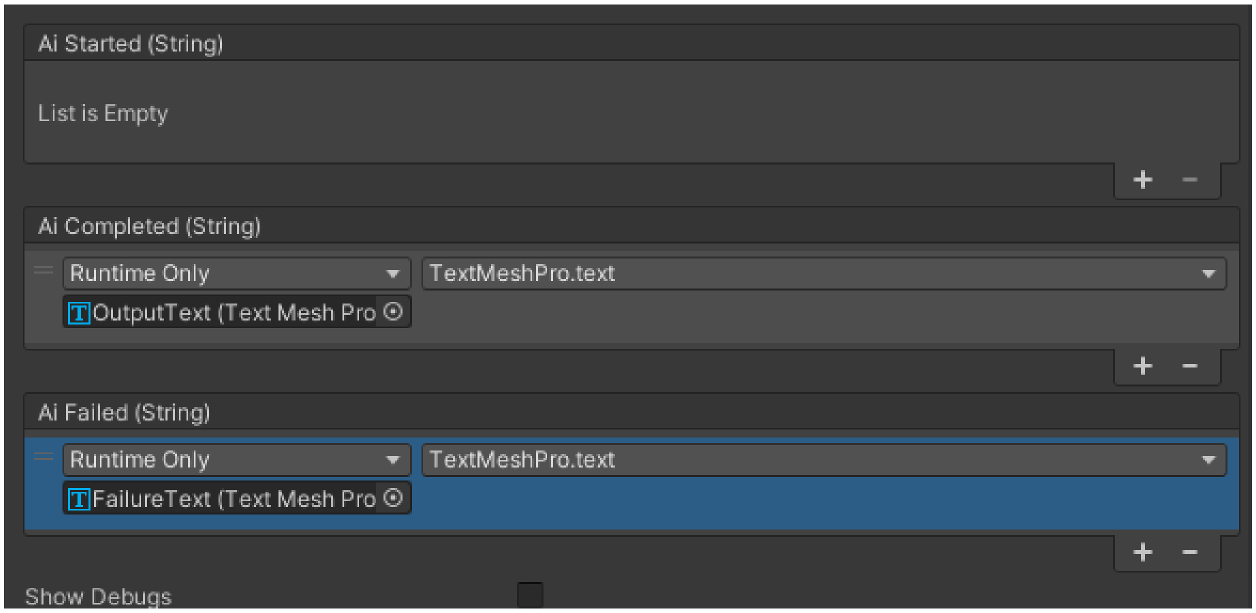
AI Components

AICore uses Inheritance in order to allow for easy plug and play components. Using Unity Events everything can be connected easily.

Base AI Events:

BaseAI is an abstract class which LLM / TTS / SST And Image Gen. This allows access to universal events and functions.

These are all UnityEvent<string> so they can use the dynamic string allowing instant updates to text fields.


Ai Started (String)

List is Empty

Ai Completed (String)

Runtime Only | TextMeshPro.text

OutputText (Text Mesh Pro

Ai Failed (String)

Runtime Only | TextMeshPro.text

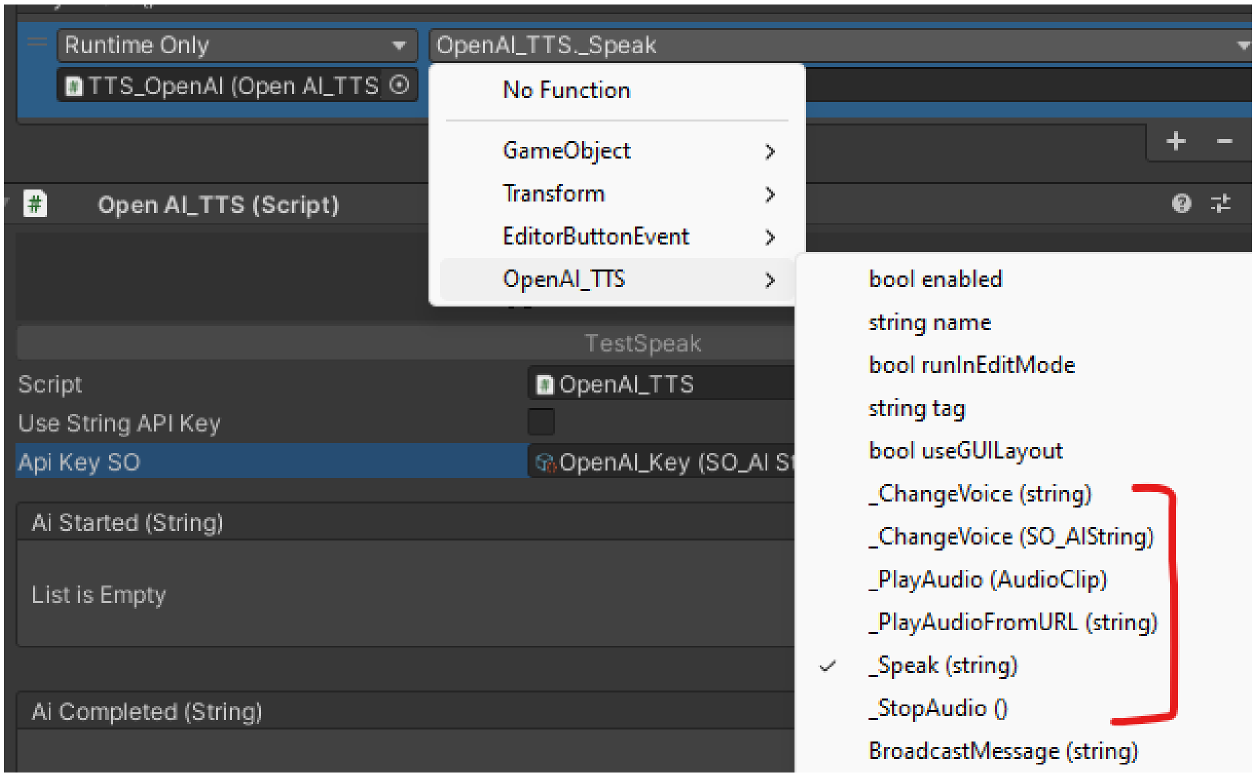
FailureText (Text Mesh Pro

Show Debugs

- AIStarted
    - Run when the AI starts and the string is the input to the AI (when in string format)
- AICompleted
    - Generates the output of the AI
- AIFailed
    - When the AI fails it will throw a `Debug Error` and it's string will be the input it received. More information on failures can be found in 😵‍💫 FAQ / Troubleshooting

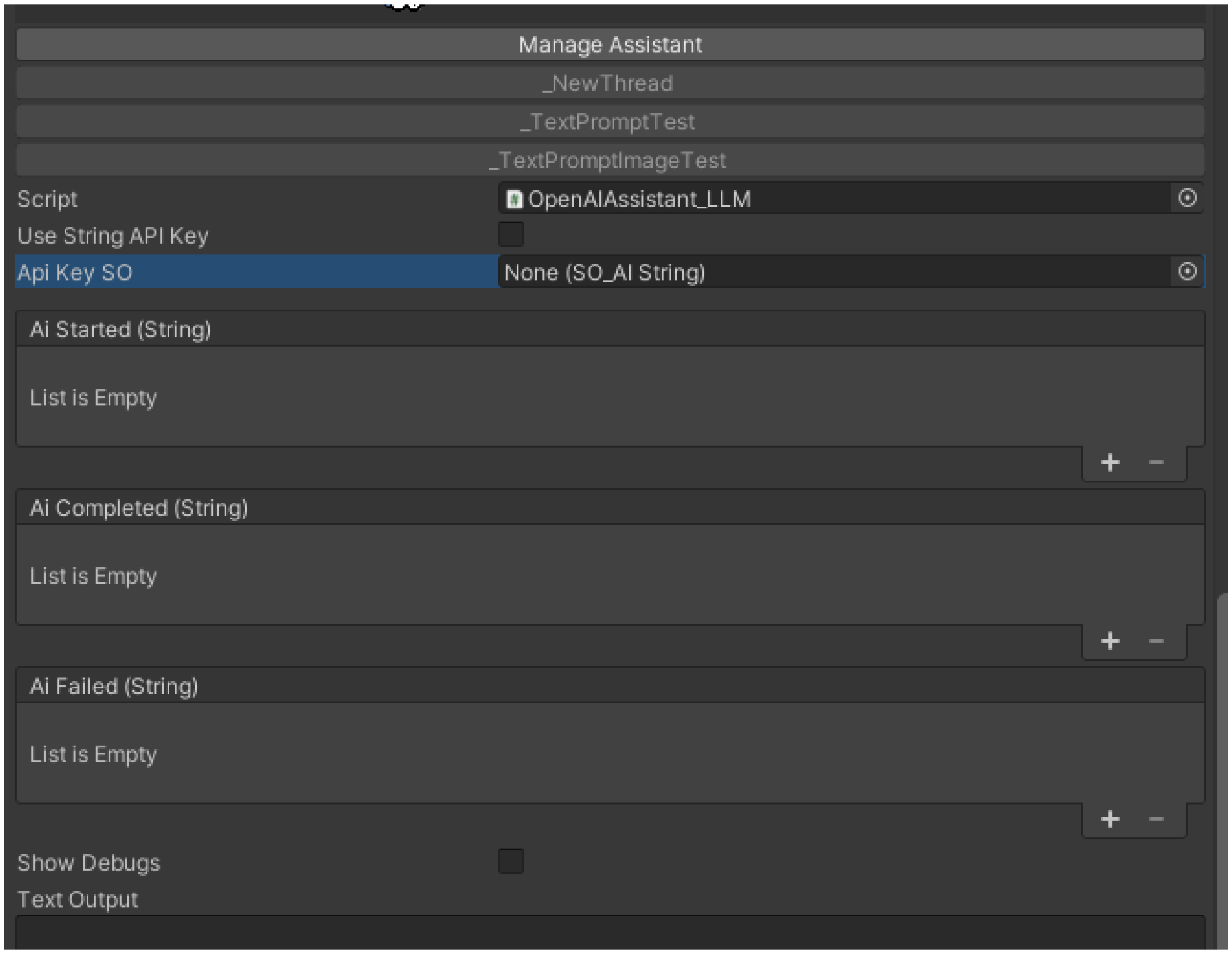
Event Functions

Any function designed to be used by a Unity Event starts with an _ Underscore for example:



Editor Buttons

At the top of some components are editor buttons. These can be clicked in the editor if they are greyed out it means they are only design to be used in playmode. These are designed to help test your application before deploying.



Script: OpenAIAssistant_LLM
Use String API Key
Api Key SO: None (SO_AI String)

Ai Started (String)
List is Empty

Ai Completed (String)
List is Empty

Ai Failed (String)
List is Empty

Show Debugs
Text Output

Show Debugs

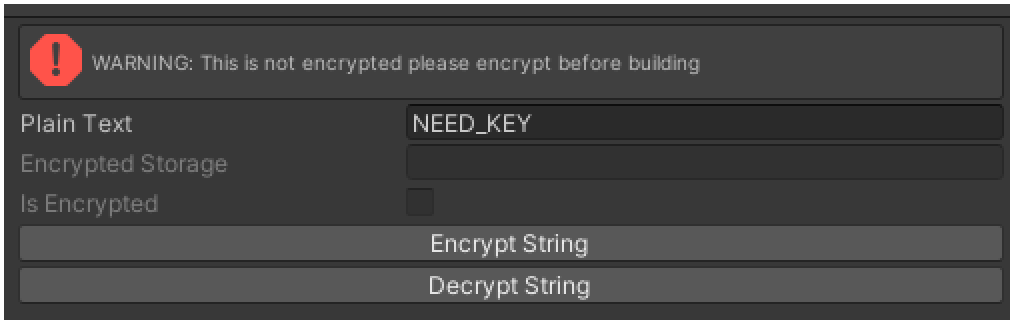
Most components have show debugs this will give you additional information.

Security - SO_StringEncrypted

AICore offers a few ways to store an API key securely first is via a `SO_StringEncrypted` which is a scriptable object. That allows you to encrypt your API keys protecting them in builds.

Make sure to

Encrypt your Strings Before Builds


WARNING: This is not encrypted please encrypt before building

Plain Text	NEED_KEY
Encrypted Storage	
Is Encrypted	☐

🛑 This encryption is considered medium security measure.

We recommend you do the following

- Rotate your API Keys
- Rate Limit on your third party account
- Monitor your Usage

The best security method is authentication getting the key via a server call and setting rate limits with your provider. you can set that using `_SetAPIKey()` and set the API keys at runtime.

BaseAI

🗒️ This abstract component serves as a base class for AI implementation. It provides functionalities to handle AI operations and events. AI Keys' are setups up using the AI_String to speed up development.

Inspector Variables

- `SO_StringEncrypted apiKeySO` The ScriptableObject containing the API key.
- `bool showDebugs` A flag to enable/disable debug information.
- `string textOutput` A text field to display output information.

Unity Events

- `public UnityEvent<string> aiStarted` Event triggered when the AI operation starts.
- `public UnityEvent<string> aiCompleted` Event triggered when the AI operation is successfully completed.
- `public UnityEvent<string> aiFailed` Event triggered when the AI operation fails.

Event Functions

- `_SetAPIKey(string key)` Forces the API key to use a string this only works at runtime

BaseAI_LLM

🗒️ This component extends BaseAI and is responsible for handling AI text prompts and image prompts as well as conversations.

Inspector Variables

- `string addToPromptStart` Text to add at the beginning of each prompt. for example (Be Brief)
- `bool debugEcho` A flag to control if the prompts should be echoed without processing them through the AI.
- `string systemPrompt` System prompt for this LLM
- `BaseAI_Conversation AIConversation` A reference to a conversation manager to store and update the conversation history.

Debug

- `string testPrompt` The test prompt string to use for testing.
- `string testImagePrompt` The test prompt string for image prompts.

- `Texture2D testImage` The test image to use for image prompts.

Event Functions

- `_TextPrompt(string chg)` Prompts the AI with the specified text.
- `_TextPromptWithImage(string chg, Texture2D image = null)` Prompts the AI with the specified text and image.
- `_TextPromptFromInput(TMP_InputField myText)` Prompts the AI with the text from a TMP_InputField.
- `_TextPromptTest()` Executes a test prompt.
- `_TextPromptImageTest()` Executes a test prompt with an image.
- `_ClearHistory()` Clears the conversation if there is a base conversation

BaseAI_Conversation

This sub component is responsible for managing conversations between the user and the assistant AI. It provides functionality to add new conversation pieces, clear conversation history, and store system prompts and conversation data. A system prompt can also be overwritten here.

Inspector Variables

- `bool overrideSystemPrompt` Flag to override the system prompt
- `string systemPromptOverride` Overwritten system prompt for BaseAI_LLM
- `int maxHistory` The maximum number of conversation pieces to store in the history.
- `List<conversationData> systemConversationList` A list of predefined conversation data that will not be cleared on _ClearConversationHistory.

💡 The system conversation is where you can pre train the model for how you want it to respond. System conversation is alway

- `List<conversationData> conversationList` The list of conversation data representing user and assistant messages.

Event Functions

- `_ClearConversationHistory()` - Clears the conversation history by removing all conversations from the `conversationList`.

BaseAI_ImageGen

🗒️ Extends BaseAI class and includes functionality for generating images based on prompts. It includes inspector variables for defining the width and height of the generated image, as well as a template material to override the default material. It also provides events for requesting a material and handles the process of downloading and applying textures.

Inspector Variables

- `int width` The width of the generated image.
- `int height` The height of the generated image.
- `Material templateMaterial` Overrides the default material if provided.
- `string addToImagePrompt` A prompt to add to the image generation.
- `UnityEvent<Material> requestMaterial` Event triggered to request a material.

Event Functions

- `_ImagePrompt(string chg)` Initiates an image prompt request based on the input `chg`.
- `_ImagePromptFromInput(TMP_InputField myText)` Initiates an image prompt request based on the text input from a TMP_InputField.

- `_ImagePromptTest()` Initiates an image prompt request with a predefined test prompt.

BaseAI_SpeechToText

For the Speech-to-text AI system. It includes settings for recording audio, detecting silence, and processing audio clips. It also includes event functions and static utility functions.

⚠ Mic Input issues are one of the tops problem with this component so ensure the correct microphone is setup.

The mic input canvas MicInputCanvas prefab can help with selecting the correct mic for testing or for users.

Inspector Variables

- `bool forceInputDevice` Determines if a specific microphone input device should be used.
- `string defaultMicrophone` The default microphone device name.
- `string inputDevice` The selected input microphone device when `forceInputDevice` is enabled.
- `List<string> inputSources` A list of available microphone input sources.
- `int sampleRate` Specifies the audio sampling rate (default: 16000).
- `float maxRecordingTime` Maximum duration for recording (default: 5 seconds).
- `bool startListeningToggles` Toggles recording to be finished if start listening is called while actively listening.

Silence Detection

💡 These are the controls for silence detection system, which will stop recording when it hears silence `requiredSilenceDuration` Tweaking values on this may be required.

- `float silenceThreshold` Threshold for detecting silence in the audio (default: 0.02).
- `float requiredSilenceDuration` Required duration of silence before stopping recording (default: 1.5 seconds).
- `float stopAfterSilence` Time to wait after silence detection before stopping (default: 1 second).
- `AudioSource audioSource` Reference to the AudioSource component used during recording.

- `AudioClip recording` Stores the recording clip.
- `bool isListening` Indicates if the system is currently listening.
- `bool echoTestForMic` does not process just echos back what the mic heard

Unity Events

- `public UnityEvent startedListening` Event invoked when recording starts.
- `public UnityEvent stopListening` Event invoked when recording stops.

Event Functions

- `_StartListening()` Starts or stops the recording process based on current state.
- `_StopRecordingAndProcess()` Stops recording and processes the captured audio.
- `_CancelListening()` Cancels the listening and ends the microphone recording.
- `__ChangeMic(string chg)` Changes input mic to this device
- `_ProcessAudioClip(AudioClip myClip)` Processes the given audio clip asynchronously.

BaseAI_TextToSpeech

> Extends BaseAI class and contains functionality for text-to-speech operations. It provides the ability to speak text prompts using a designated voice and supports customization through a scriptable object.

Inspector Variables

- `public AudioSource speakerSource` The audio source used for speech playback.
- `public string testPrompt` A prompt used for testing text-to-speech functionality.
- `public bool useScriptable` A flag to control if a scriptable object should be used for voice selection.
- `public string voiceID` The ID of the voice to be used for speech synthesis.
- `public SO_StringEncrypted voiceSO` A scriptable object that contains the voice ID.

Unity Events

- `public UnityEvent audioPlayStarted` Event triggered when audio playback starts.

Event Functions

- `_StopAudio()` Stops the currently playing audio.
- `_Speak(string chg)` Converts the input string into speech and plays it using the specified voice.
- `_ChangeVoice(string voiceChange)` Changes the voice to be used for speech synthesis.
- `_ChangeVoice(SO_StringEncrypted goString)` Changes the voice to be used for speech synthesis based on a scriptable object.
- `_PlayAudio(AudioClip whatClip)` Plays the specified audio clip.
- `_PlayAudioFromURL(string url)` Plays an audio clip from the specified URL.

Using Meta Voice SDK

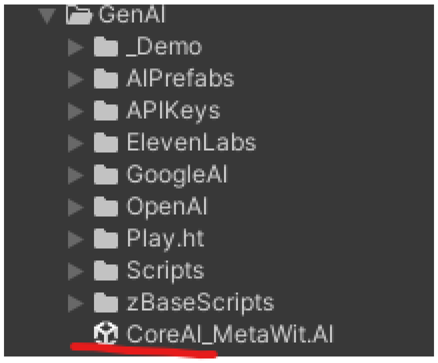
Prep Work:

Step 1:

- Download https://assetstore.unity.com/packages/tools/integration/meta-voice-sdk-immersive-voice-commands-264555 into your project

Step 2:

- Open CoreA_MetaWit.AI Package



Information on this can be found here

https://developer.oculus.com/documentation/unity/voice-sdk-overview/

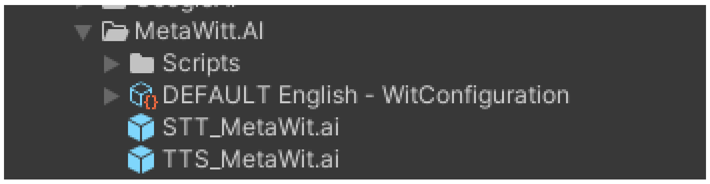
Getting your Keys

Wit.AI keys are acquired from https://wit.ai/ however we have provided `DEFAULT English - WitConfiguration` Which will speak and transcribe in english. If you need additional languages or configurations please refer to the documentation or create a new configuration by accessing the menu Meta > VoiceSDK > Getting Started

Witt.AI Handles things very differently as it is heavily integrated into Meta's SDK as such API keys are not required and some additional BaseAI features are not used.

Getting Started

After importing MetaVoiceSDK open the MetaWitAI.package



For best results use the provided prefabs which are already configured with Default configuration and ready to use

Meta_SpeechToText : BaseAI_SpeechToText

This component facilitates speech-to-text functionality using the Meta Wit.AI API. It listens for voice input, processes transcriptions, and supports switching microphones.

Silence detection is handled on the `AppVoiceExperience` component

The Prefab MetaSpeechToText is already setup and ready to go with default configuration

Inspector Variables

- `AppVoiceExperience metaAppVoiceExperience` Reference to the Meta App Voice Experience component.

Unity Events

- `public UnityEvent<string> partialTranscribe` Event triggered with partial transcriptions. as Wit.AI supports streaming transcripts

Meta_SpeechToText : BaseAI_SpeechToText

This component is responsible for text-to-speech functionality using the Meta TTS system. It manages the initialization of the TTS speaker and processes voice input with optional SSML formatting.

The Prefab MetaTextToSpeech is already setup and ready to go with default configuration

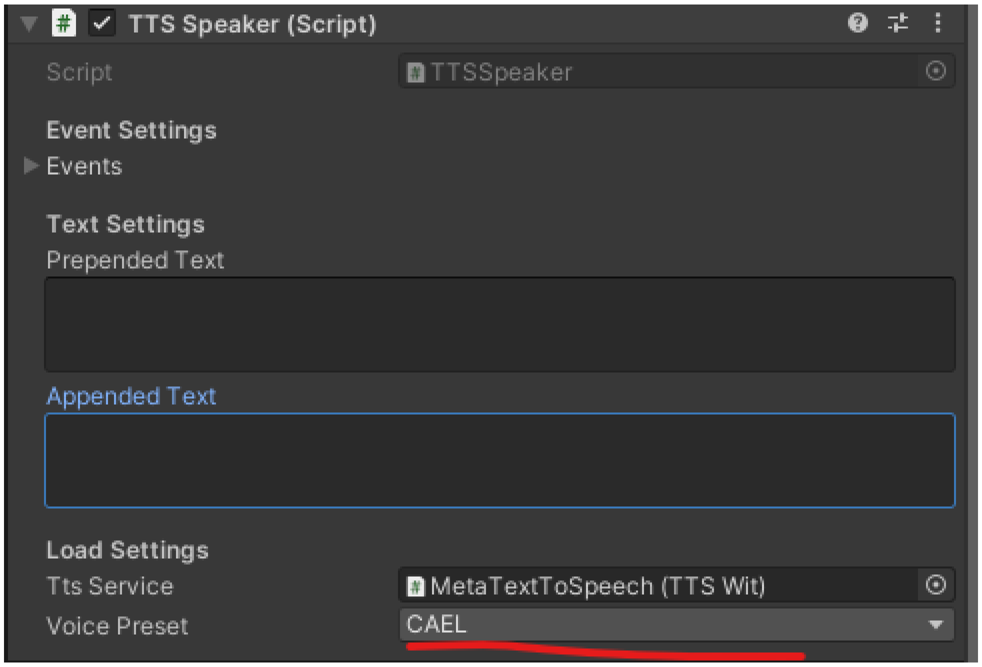
This component does not work with `TTS_SaveToFile`

Inspector Variables

- `TTSSpeaker MetaTTS` The TTS speaker component used for text-to-speech.
- `bool useSSML` A flag to indicate whether to use SSML formatting for the speech.

Changing Voices:

Changing voices is not handled in the Meta_SpeechToTex. Instead it's handled in the TTS-Speak component.



TTS Speaker (Script)

Script: TTSSpeaker

Event Settings

Events

Text Settings

Prepended Text

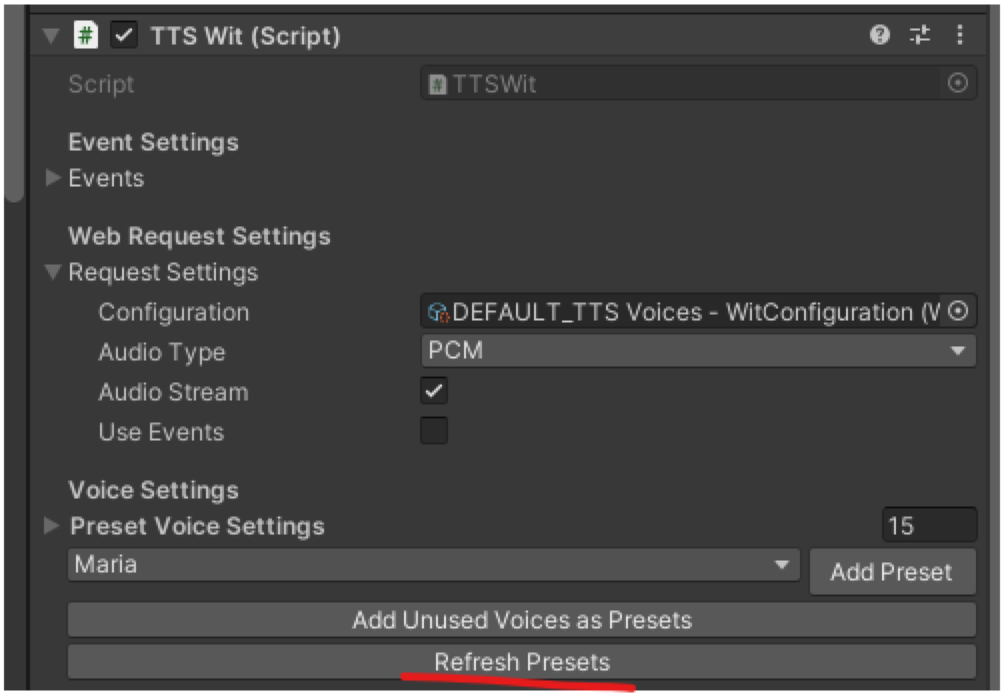
Appended Text

Load Settings

Tts Service: MetaTextToSpeech (TTS Wit)

Voice Preset: CAEL

If you are missing voices you may need to refresh your presets at TTS_Wit. For additional information refer to the documentation.



TTS Wit (Script)

Script: TTSWit

Event Settings

Events

Web Request Settings

Request Settings

Configuration: DEFAULT_TTS Voices - WitConfiguration (W

Audio Type: PCM

Audio Stream: ✓

Use Events: ☐

Voice Settings

Preset Voice Settings: 15

Maria

Add Preset

Add Unused Voices as Presets

Refresh Presets