DIALOGUE SYSTEM for unity Addon for OpenAI

Table of Contents

Introduction

Thank you for supporting the Dialogue System for Unity and the Addon for OpenAI, ElevenLabs, and Other Generative AI! This addon requires the Dialogue System for Unity. Using the Dialogue System Addon for OpenAI, you can use generative AI to write narrative content faster, get suggestions when you're stuck, fix spelling and grammar, and translate your text to other languages. You can also use the addon for freeform chat similar to ChatGPT, generate portraits using DALL-E, and voiceover using many options including OpenAI, ElevenLabs, DeepVoice, and Overtone.

We want your project to be a success! If you have any questions or feature requests, please contact us any time at support@pixelcrushers.com.

We're Here to Help

- Email: support@pixelcrushers.com
- Forum: https://pixelcrushers.com/phpbb
- Discord: https://discord.gg/FwUaCNt
- Web: Tools → Pixel Crushers → Dialogue System → Help → Report a Bug

Quick Start

- Read the documentation and/or watch the tutorial videos.
- Import the Dialogue System for Unity into your project.
- Import the Dialogue System Addon for OpenAI into your project.
- The addon's Welcome window will appear. Configure the addon to proceed.
- Click the OpenAI Addon Window button in the Welcome window or the AI buttons in the Dialogue System's Dialogue Editor window.

Generative AI and the Dialogue System for Unity

OpenAI is an AI research company best known for technology designed to generate human-like language, in particular using Large Language Models (LLMs) designed for natural language processing tasks. OpenAI has a public API that applications such as the Dialogue System Addon for OpenAI can use to leverage the power of its language models. Modest charges for the use of the API are handled through an API key that you can create on OpenAI's website. (OpenAI Pricing) The Dialogue System Addon's Welcome window will step you through the process if you don't already have an API key.

Ollama is an open-source application that lets users run certain large language models (LLMs) locally on their own computers. In the case of the Dialogue System Addon, this includes Meta's Llama 3.2 and Llama 3.3 models as well as any other models you've installed locally.

The **Dialogue System Addon for OpenAI** is a third party product (i.e., not affiliated with OpenAI or Meta) that allows you to leverage the OpenAI and Llama APIs to enhance your content creation. It's designed to work seamlessly with the Dialogue System for Unity.

Note: Language models are not sentient. They are essentially just very advanced pattern matchers. Results may contain grammatical or factual errors, so review them carefully before using them.

Language Models

The OpenAI API provides access to several language models and parameters. In the Dialogue System Addon, the UIs to specify language models and parameters look like this:

Model: This dropdown allows you to choose a language model for content generation:

Model	Description
GPT_4_5_Preview	Preview version of GPT-4.5.
GPT_4.1	OpenAI’s flagship GPT model for complex tasks
GPT_4o	High-intelligence model for complex, multi-step tasks
GPT_4o_mini	Affordable & intelligent small model for fast, lightweight tasks.
GPT_4_Turbo	GPT-4 with 128k context window.
GPT_4_32K	GPT-4 with larger 32k context window.
GPT_4	Optimized for chat but works well for traditional completions tasks.
GPT_3_5_Turbo_16K	GPT-3.5 Turbo with 16k context window.
GPT_3_5_Turbo	Optimized for chat.
o4-mini	Fast, affordable high-intelligence reasoning model.
O3	One of OpenAI's most powerful reasoning models.
o3-mini	Fast, flexible, intelligent reasoning model.
o1-mini	A faster, more affordable reasoning model than o1.
o1	High-intelligence reasoning model.
o1-pro	A version of o1 with more compute for better responses.
Fine-Tuned	Your own fine-tuned model.
Davinci	Good at complex intent and cause & effect.
Curie	Good at language translation.
Babbage	Good at classification.
Ada	Oldest model. Results may be poor.

Llama 3.2	Medium size LLM suitable for relatively fast responses on desktop PCs.
Llama 3.3	Larger LLM that may require more powerful hardware for responsiveness.
Custom Ollama	Any Ollama model installed on your system, such as Deepseek or Gemma.

Temperature: Specifies the amount of "randomness" to include in the content. A value of zero generates plain, straightforward responses. A value of one generates more unexpected or "creative" responses.

Max Tokens: Specifies the maximum number of tokens to spend on a response. Lower numbers generate shorter responses; higher numbers can generate longer responses. OpenAI pricing is based on the number of tokens used, so you can use this slider to reduce expenses.

You can also use OpenAI to generate images and speech.

For the best balance of speed and price, we recommend GPT-4o-mini (if using OpenAI) or Llama 3.2 (if using Meta's Llama models).

Setting Up Ollama

If you want to use the Llama LLMs locally, you must install Ollama. Instructions are here: https://ollama.com/

If you've installed the Llama 3.2 or 3.3 models, you can select them from the model dropdowns in the addon. If you've installed any other models, you can specify the model name in the main addon window's **Fine-Tuned & Custom Models** section.

ElevenLabs Speech Synthesis

You can use the API provided by ElevenLabs, a separate company from OpenAI that specializes in speech synthesis, to generate voiceover audio for your dialogue at design time or runtime.

ElevenLabs uses a separate API key. Any charges for using the API are applied to your API key account. (See Pricing.)

Overtone Speech Synthesis

You can also use Least Squares' Overtone for runtime text to speech. See Overtone Integration.

DeepVoice Speech Synthesis

You can also use AiKodex's DeepVoice for editor-time text to speech. See DeepVoice Integration.

General AI Assistant

The Dialogue System Addon for OpenAI also has a general AI Assistant window that's useful to ask questions in a ChatGPT-like interface.

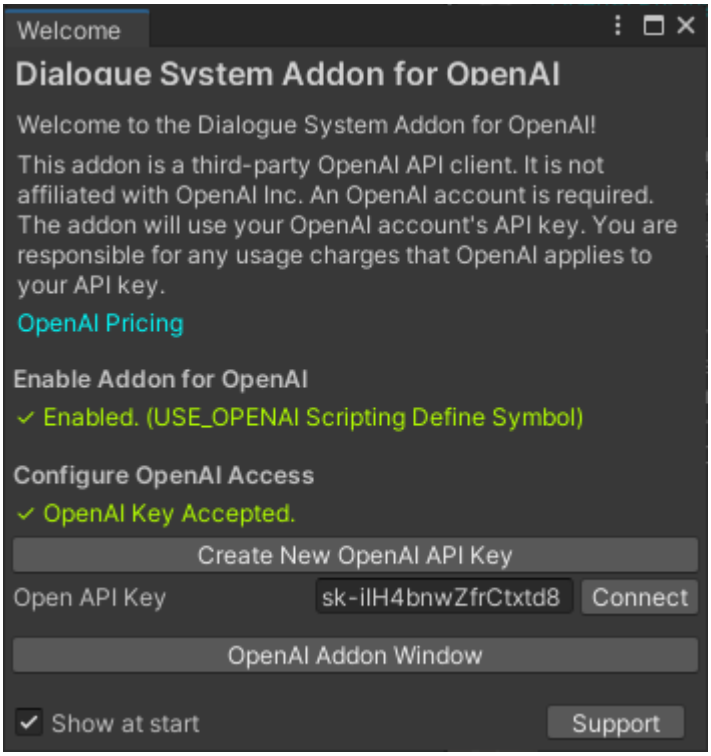
OpenAI API Base URL & Other Services

The Dialogue System Addon for OpenAI connects to the OpenAI API at `https://api.openai.com/v1/` by default. If you want to use a different service, such as Microsoft Azure, that complies to the same API, you can set `OpenAI.BaseURL`. The URL should end with a forward slash (`/`).

Assembly Definition Files

Optional: The Dialogue System and this Addon are in Unity's special **Plugins** folder, which is compiled separately from your project's main code, so there's no need to use assembly definition files (asmdefs) to improve compilation times. However, if your own code is in an asmdef, you may need to reference the Dialogue System through asmdefs. To do this, import the asmdef unitypackages located in the Dialogue System's `Scripts` folder and the Addon's folder. Then you can add references to the DialogueSystem and PixelCrushers asmdefs in your own asmdef.

Welcome Window

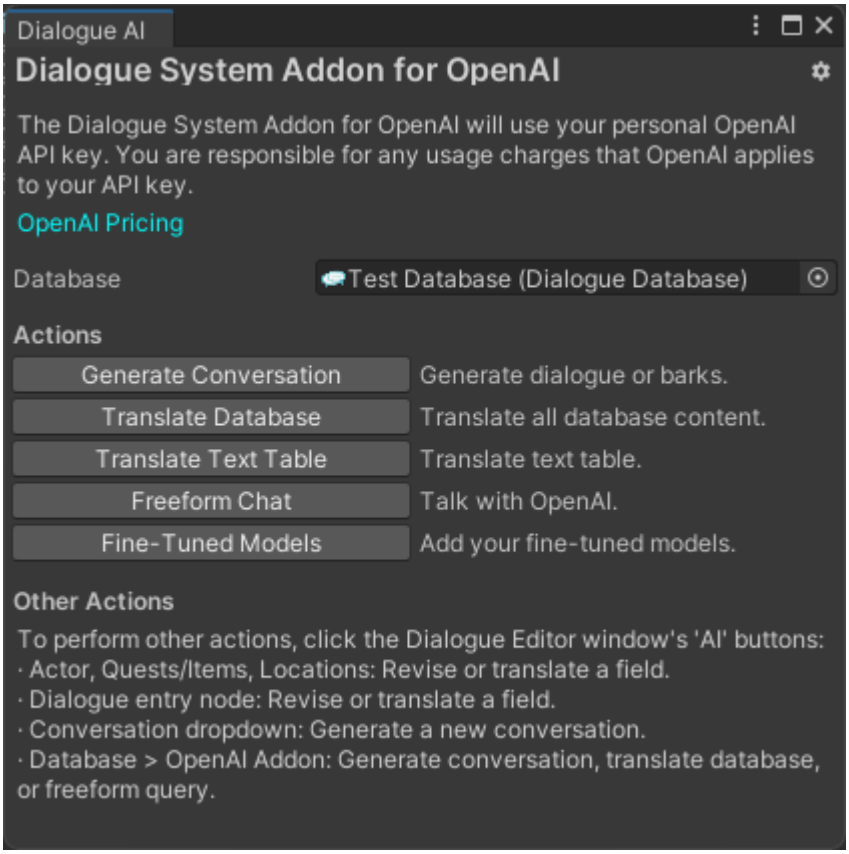


Use menu item **Tools → Pixel Crushers →Dialogue System → Addon for OpenAI → Welcome Window** to open the Welcome window.

The Addon for OpenAI will use your OpenAI API key, which you can create on OpenAI's website. You are responsible for any usage charges that OpenAI applies to your API key.

You can use the Welcome window to create and enter an OpenAI API key, and to open the Main Window.

Main Window



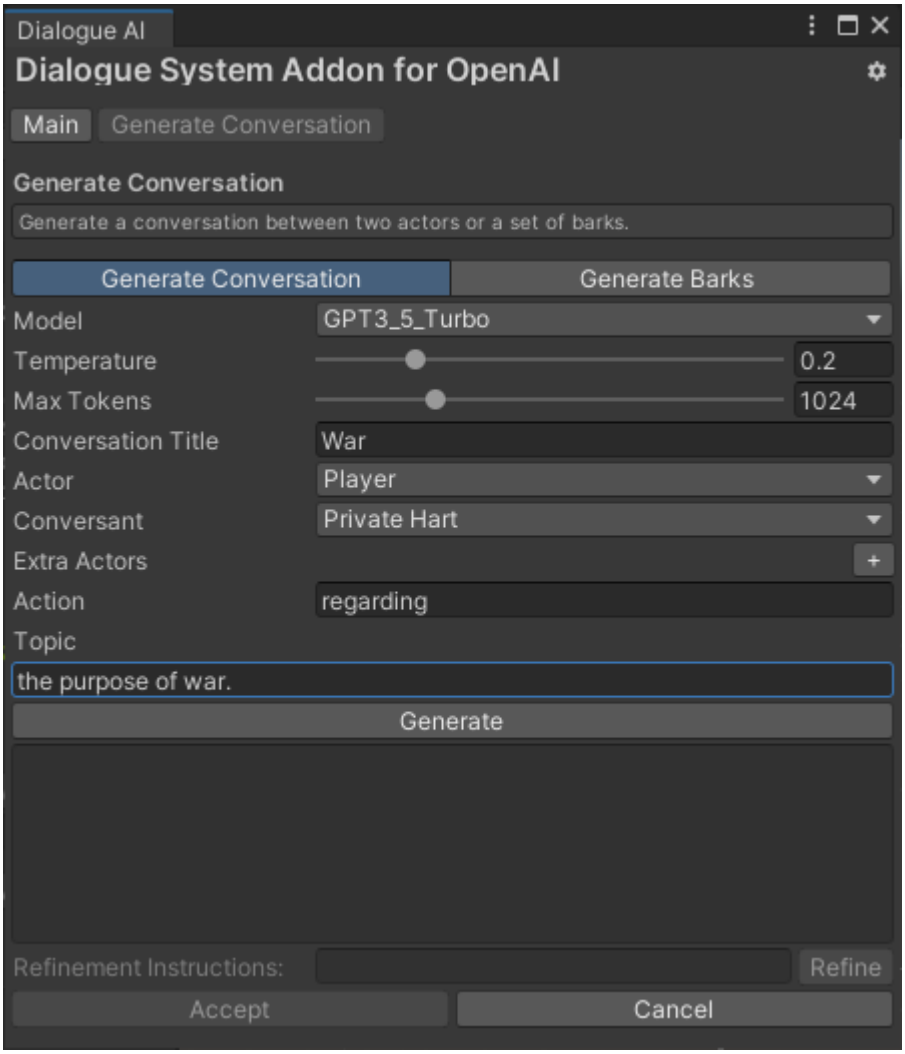
You can access the Main window from the addon's Welcome window or menu item **Tools → Pixel Crushers →Dialogue System → Addon for OpenAI → Main Window**.

On the Main Window, you can access these windows:

- **Generate Conversation**: Generate a new conversation or set of barks.
- **Translate Database**: Translate your database to one or more languages.
- **Translate Text Table**: Translate a text table asset.
- **Freeform Chat**: Talk with OpenAI in a format similar to ChatGPT.
- **Fine-Tuned Models**: Add your own fine-tuned models to the addon.

The Addon for OpenAI also adds **AI** buttons to several locations in the Dialogue System's Dialogue Editor window. You can click these **AI** buttons to access additional features.

Generate Conversation



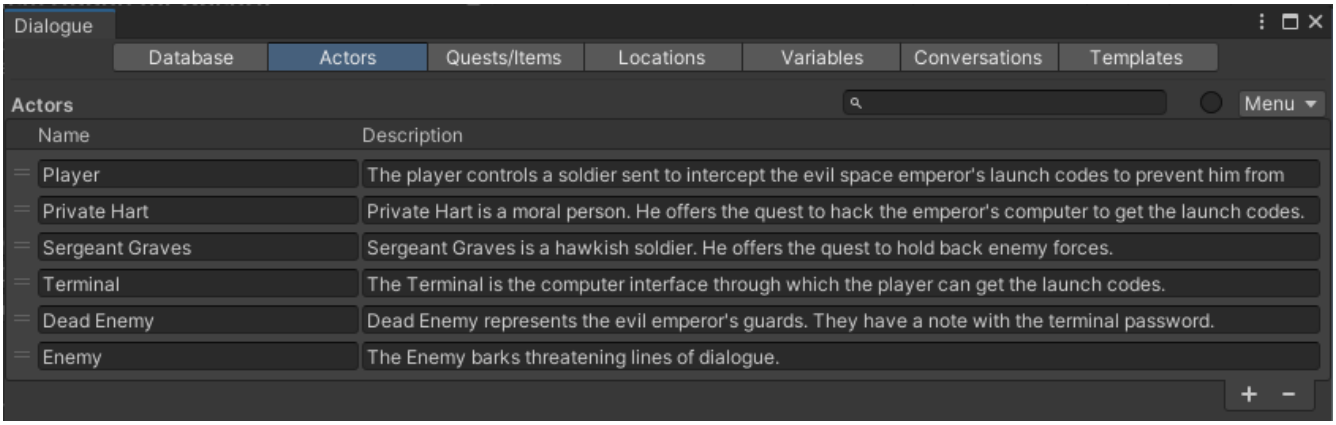
Access the Generate Conversation window from the Main Window. Here, you can create a new linear conversation between two actors or create a bark conversation.

Generate Conversation

To create a conversation, specify a title, participants, action, and topic. For example, the screenshot above will prompt the GPT_3_5_Turbo language model to generate a conversation titled “War” between Private Hart and Sergeant Graves regarding the purpose of war. Then click **Generate**.

When the Addon receives a response from OpenAI, it will show the generated conversation. If you want to adjust the conversation, enter Refinement Instructions and click **Refine**. To add the conversation to your dialogue database, click **Accept**.

When creating a conversation, the prompt will also include the participants' **Description** fields and any actors mentioned in the topic. Include the actor's name in the Description field, such as:



Knowledge & Goals Fields

You can optionally add fields named **Knowledge** and **Goals** to actors to contain additional information to provide in the prompt. To add these fields to all actors, open the Dialogue Editor's **Templates** section > **Actors Template** and click **Add Knowledge and Goals Fields**.

Location Descriptions

If you tick the Generate Conversation window's **Use Location Descriptions** checkbox (not shown in screenshot above) the prompt will also include the **Description** fields of all Locations in your dialogue database.

Branching Dialogue

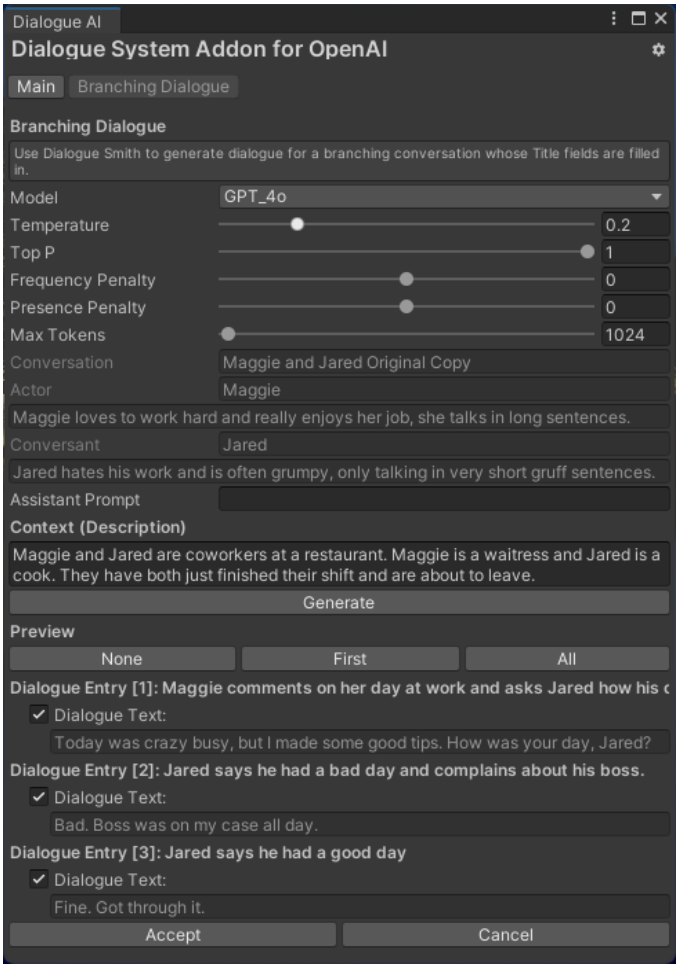
To generate branching dialogue, see the Branching Dialogue section.

Generate Barks

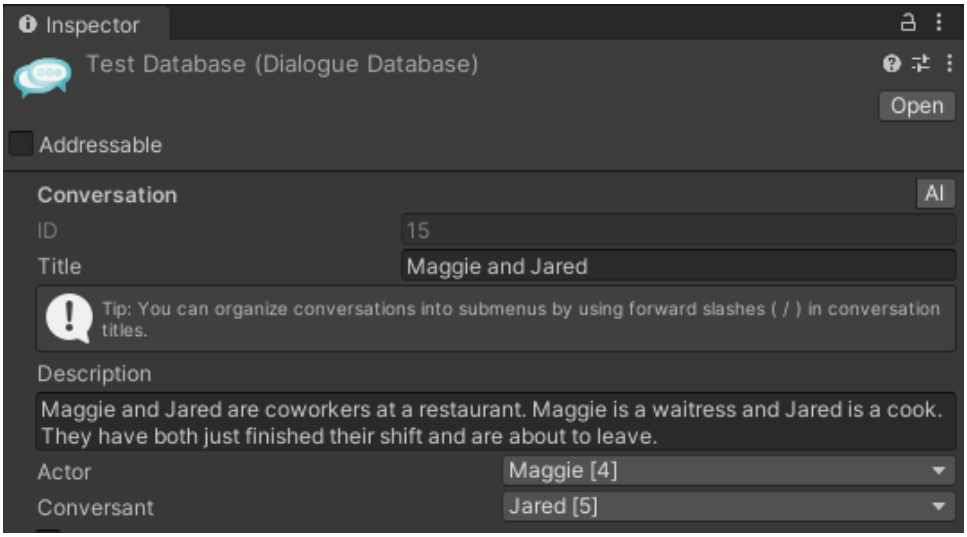
To generate barks, click the Generate Barks toolbar. The UI will appear the same as Generate Conversation with the addition of a Num Barks slider. Set the slider to the number of barks you'd like to generate, and click **Generate Barks**.

When you're satisfied with the results, click **Accept** to add a conversation containing the barks.

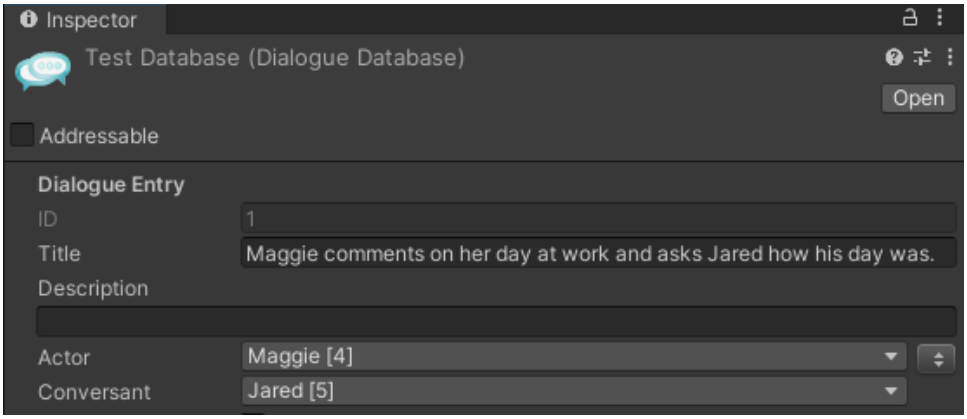
Branching Dialogue



The Addon can also fill in text for pre-existing conversations. Set the conversation's **Description** to the subject matter of the conversation:

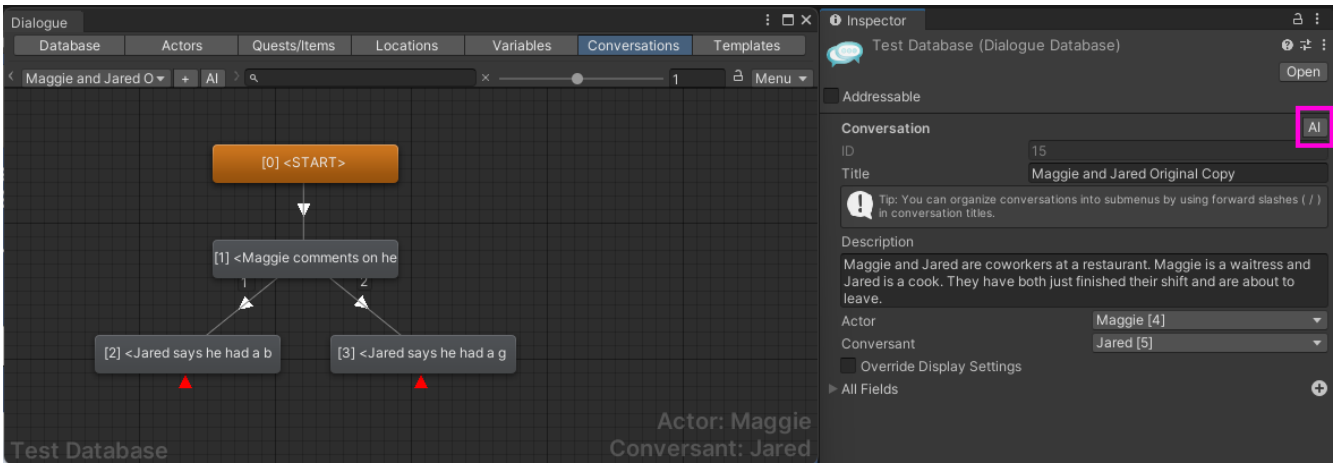


Set each dialogue entry's **Title** to the general idea of what you want the actor to say:



TIP: If you plan to have more than one conversation that uses a similar structure, you can use conversation templates (video tutorial).

Then inspect your conversation properties and click the AI button next to the Conversation heading:

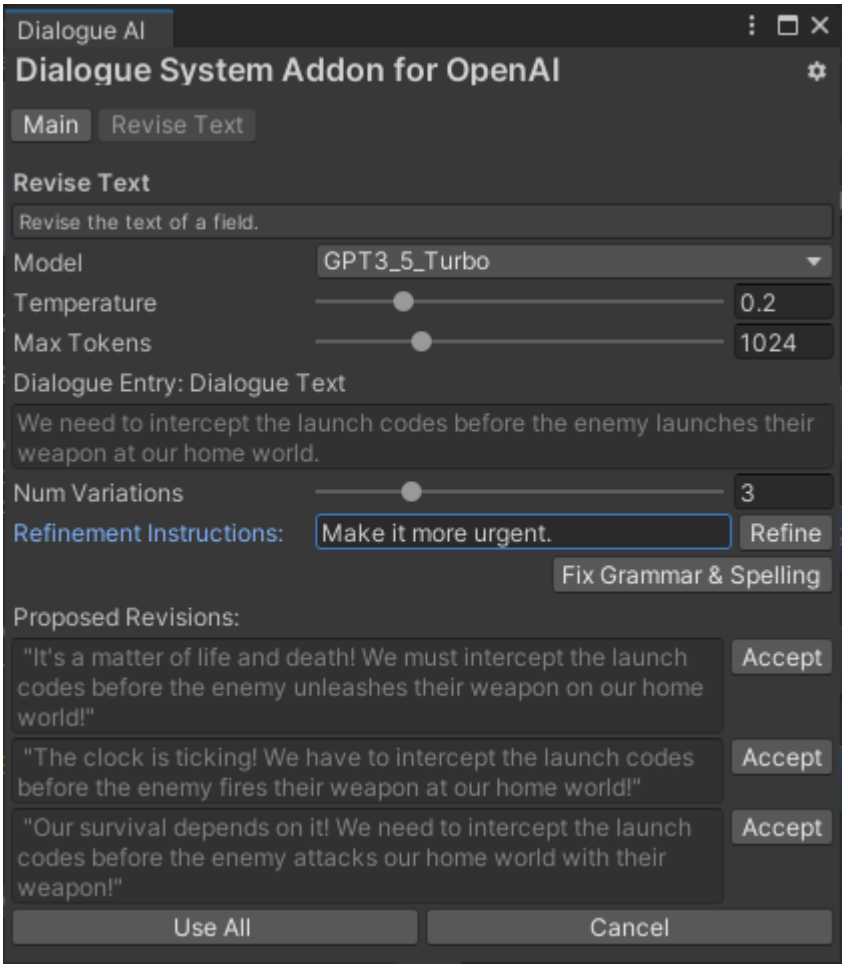


Set the OpenAI model settings you want to use, and click **Generate**. The Addon will generate text for each dialogue entry that doesn't already have dialogue text. When complete, you can tick the checkboxes next to the texts that you want to accept. Then click Accept to add them to the conversation.

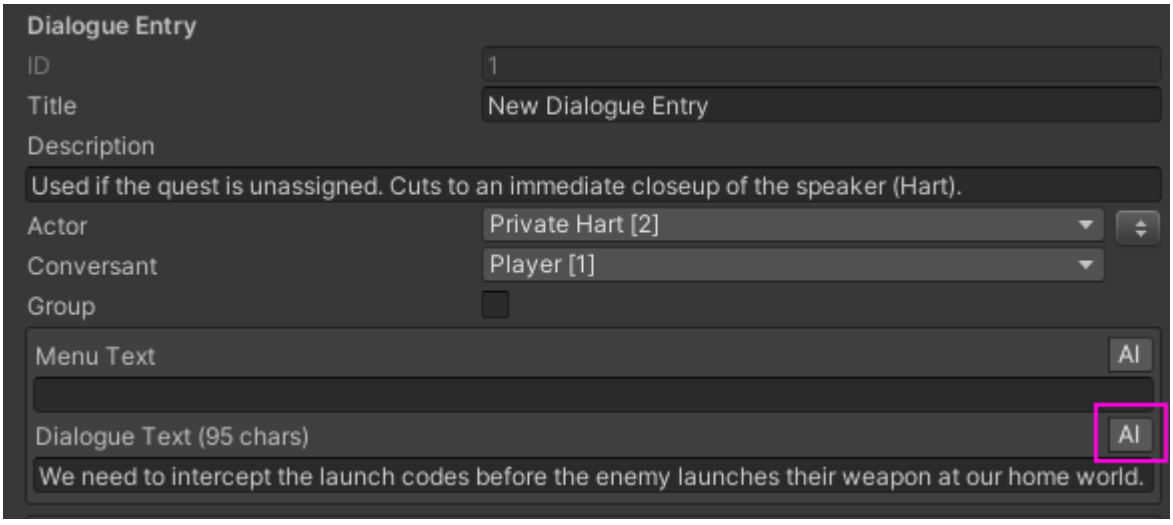
Variations

Not shown: **# Variations** slider. Use this slider to generate several different variants for each dialogue entry. You can choose to accept any number of the options presented. If you accept two or more variants, the addon will add a randomizer entry that links to all of the accepted variants so that conversations will randomly choose among them.

Revise Text



Access the Revise Text window by clicking the **AI** button next to a field in the Dialogue Editor such as the Dialogue Text field:

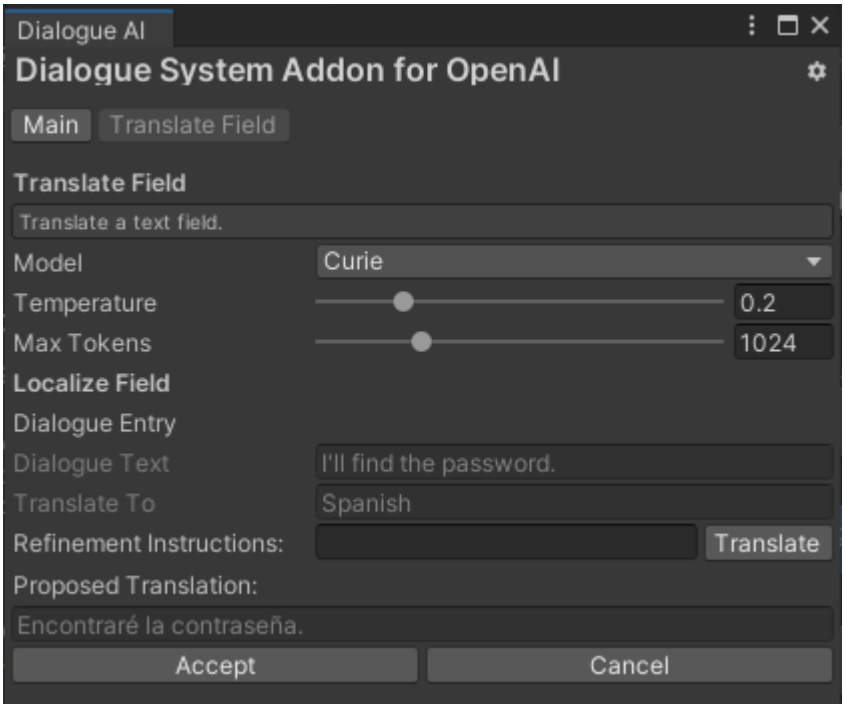


You can specify how you want to refine the text by specifying refinement instructions and clicking **Refine**, or click **Fix Grammar & Spelling** to fix grammar and spelling.

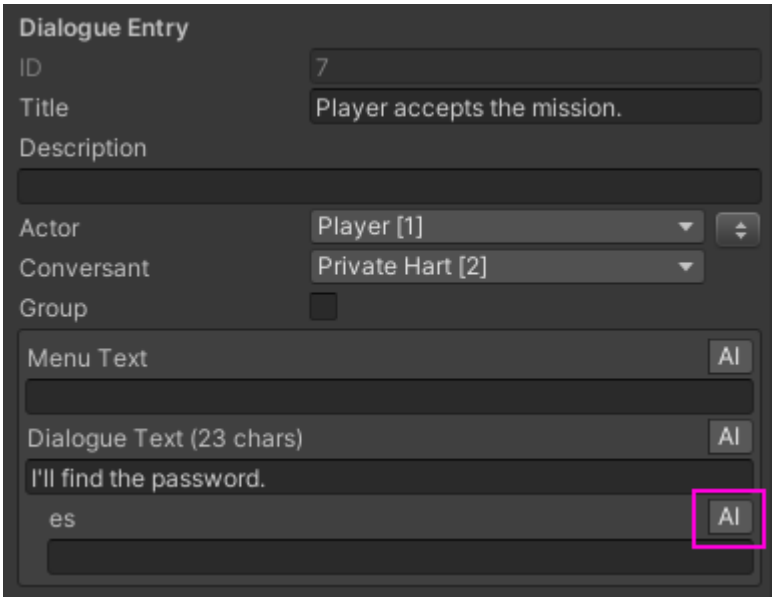
If the **Num Variations** slider is set to 1, you will receive one proposed revision. To accept it, click **Accept**.

If **Num Variations** is greater than 1, you will receive multiple suggestions. To accept one of the suggestions, click the **Accept** button next to it. Or you can click **Use All** to add them all to the conversation, replacing the original dialogue entry node. If the original node was a non-player node, it will be replaced by a group node that calls RandomizeNextEntry() to randomly choose from the variations.

Translate Field



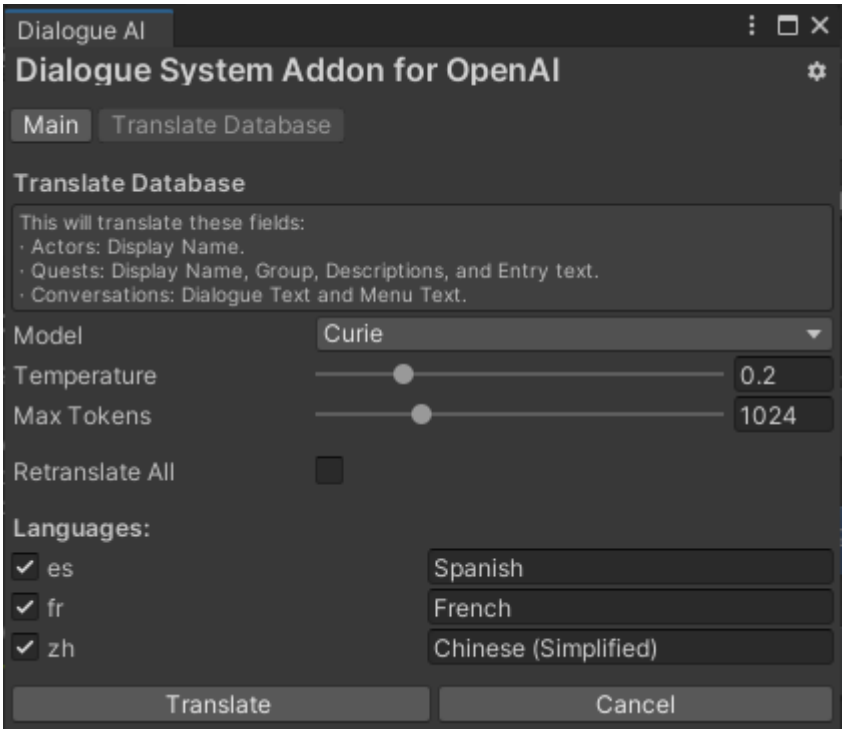
Access the Translate Field window by clicking the **AI** button next to any localization field in the Dialogue Editor, such as this field in a dialogue entry node: (See Localization)



Enter optional refinement instructions, then click **Translate**. To accept the translation, click **Accept**.

Note: This AI button opens a menu to select Translate or Voiceover to generate voiceover audio.

Translate Database



Access the Translate Database window from the Main Window. This window allows you to translate the entire database. Define the languages that you want to translate to in the Dialogue Editor first. (See Localization)

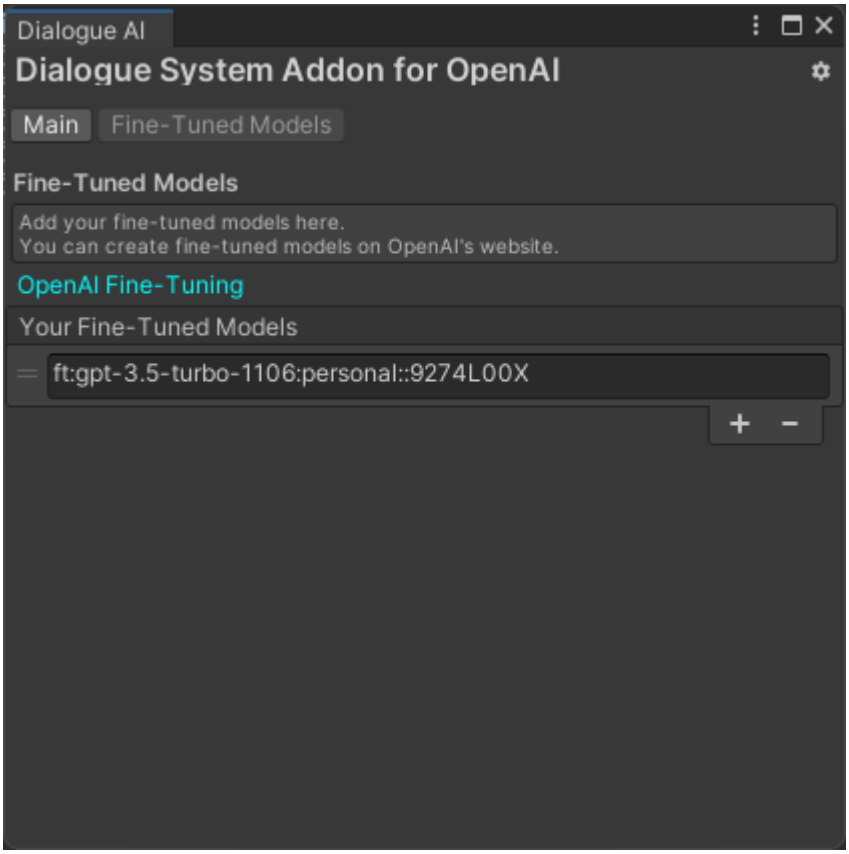
By default, to minimize costs, this window will only translate localization fields that are blank. To retranslate fields that already have content, tick **Retranslate All**.

Then tick the languages that you want to translate to, and click **Translate**.

If you have a medium or large database, this operation will take some time. OpenAI limits API calls to 60 per minute, so the operation will batch several requests and then wait a little before continuing.

Note: Translate Field defaults to the Curie language model, but you can change it to GPT3_5_Turbo if you prefer.

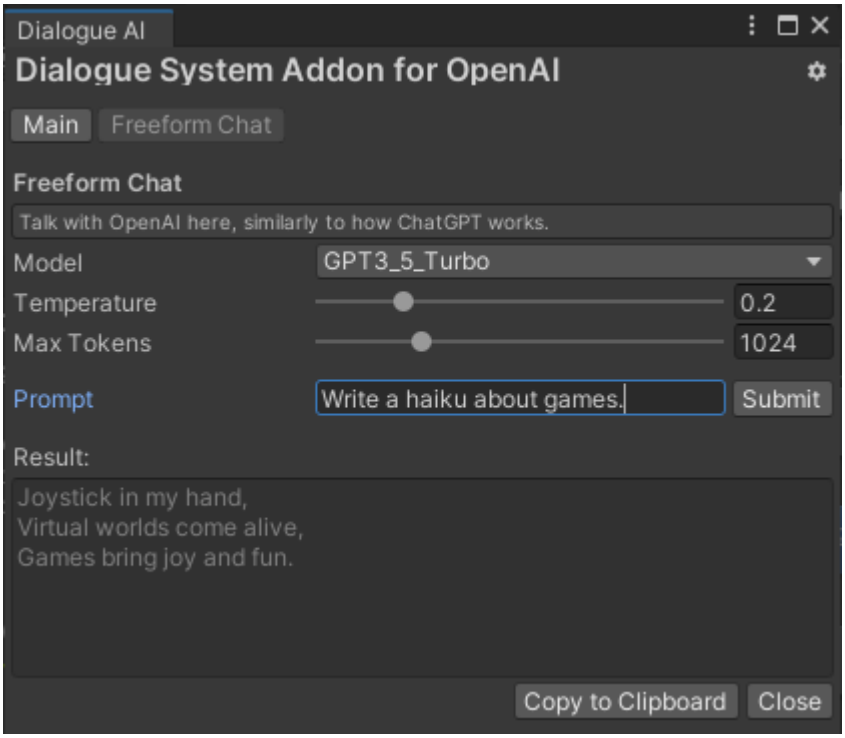
Fine-Tuned Models



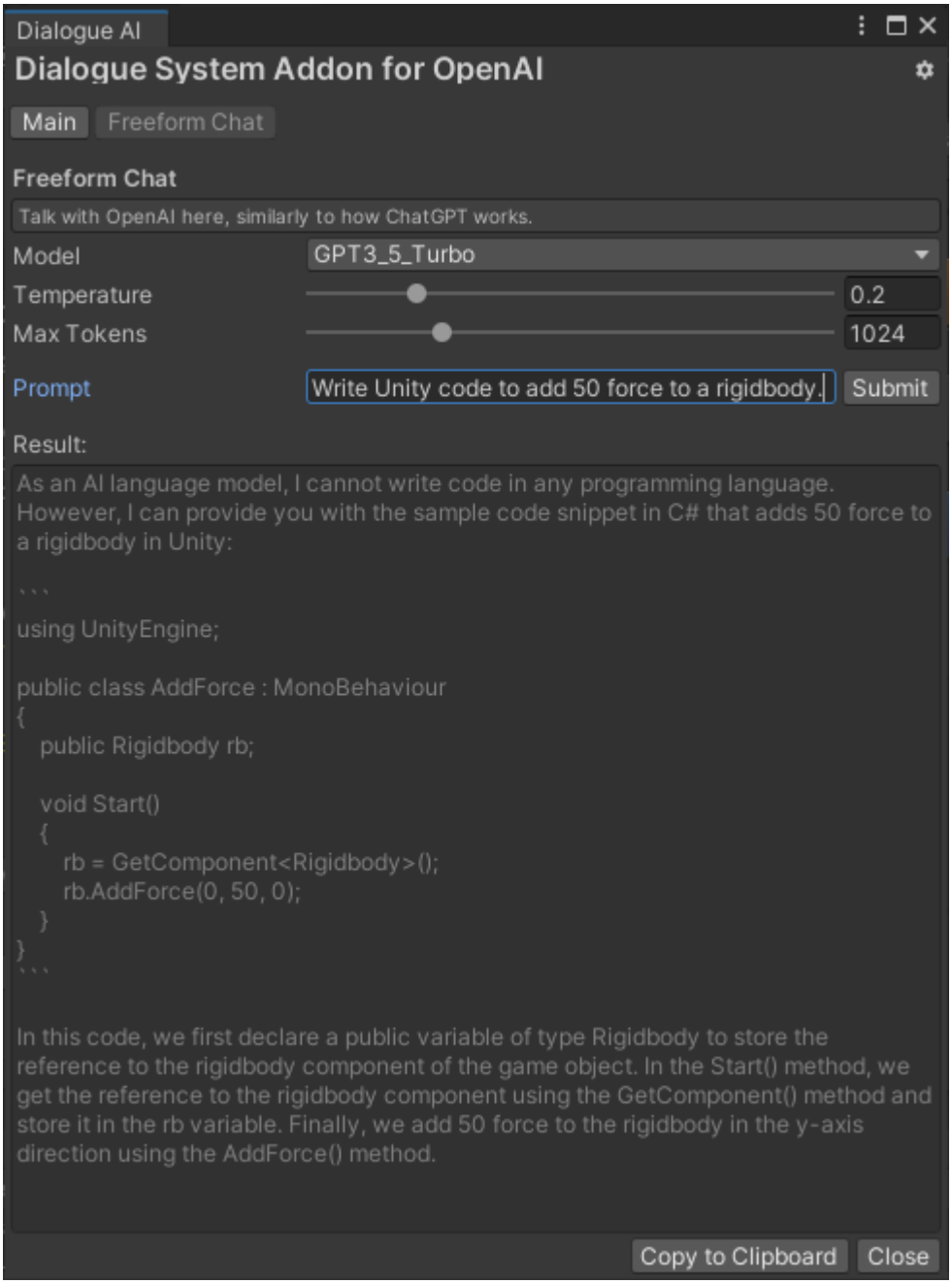
Add your custom fine-tuned models to the Fine-Tuned Models panel to make them available in the Model dropdowns of the addon's text generation panels. To create a fine-tuned model, upload a JSONL file on OpenAI's website:

https://platform.openai.com/finetune

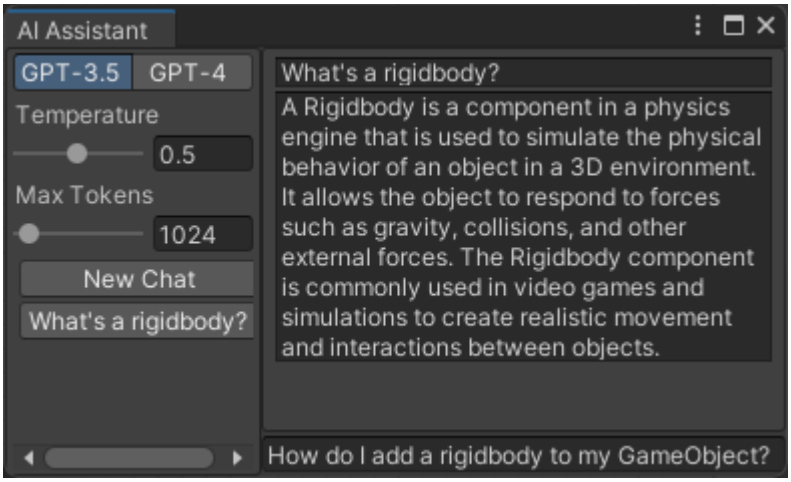
Freeform Chat



Access the Freeform Chat window from the Main Window. This window operates similarly to ChatGPT. You can repeatedly enter text in the **Prompt** field to continue a dialogue with the OpenAI language model. You can then copy this to the clipboard for use elsewhere, such as pasting generated code into a script, such as: (continued on next page)

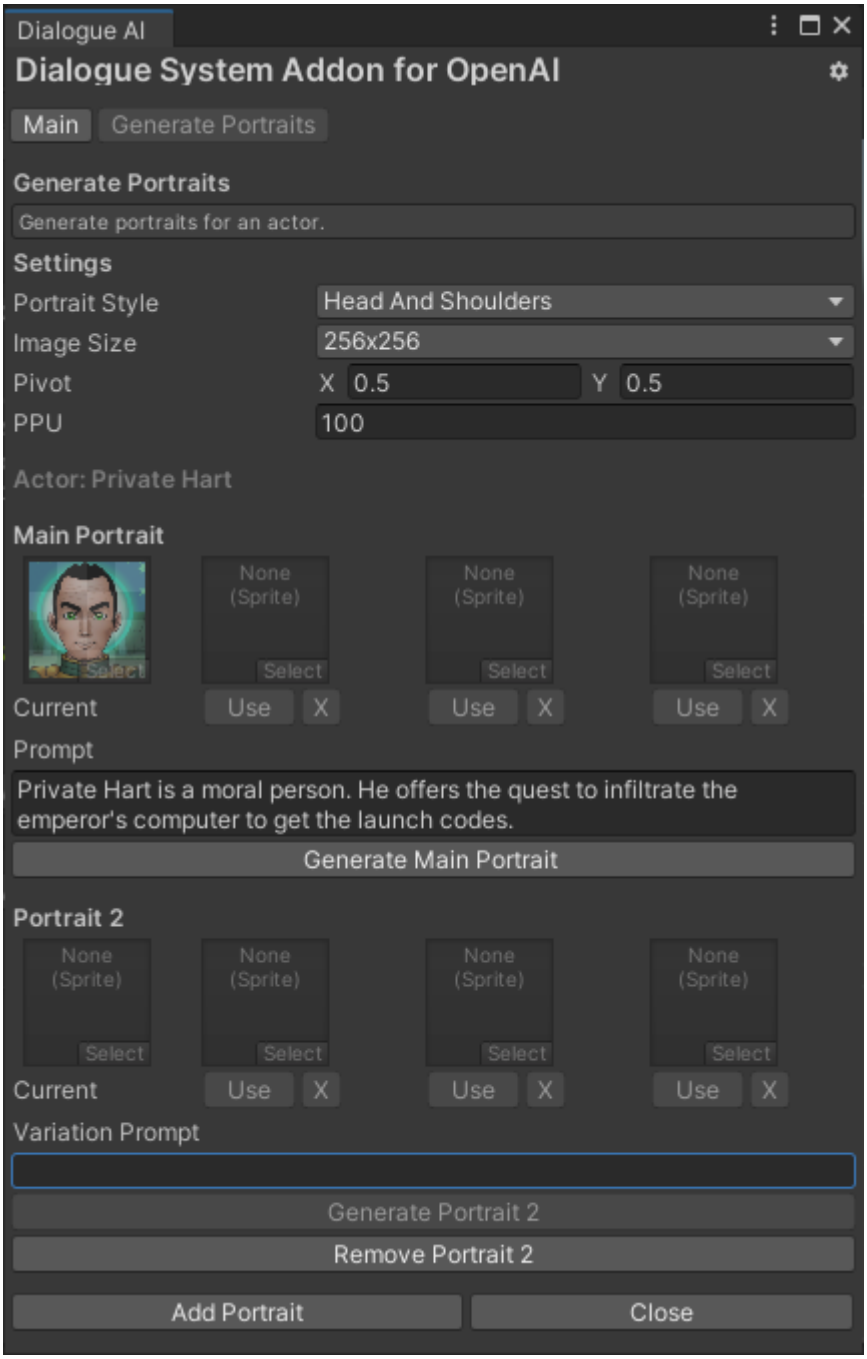

Dialogue AI
Dialogue System Addon for OpenAI
Main
Freeform Chat
Freeform Chat
Talk with OpenAI here, similarly to how ChatGPT works.
Model
GPT3_5_Turbo
Temperature
0.2
Max Tokens
1024
Prompt
Write Unity code to add 50 force to a rigidbody.
Submit
Result:
As an AI language model, I cannot write code in any programming language. However, I can provide you with the sample code snippet in C# that adds 50 force to a rigidbody in Unity:
```
using UnityEngine;
public class AddForce : MonoBehaviour
{
public Rigidbody rb;
void Start()
{
rb = GetComponent<Rigidbody>();
rb.AddForce(0, 50, 0);
}
}
```
In this code, we first declare a public variable of type Rigidbody to store the reference to the rigidbody component of the game object. In the Start() method, we get the reference to the rigidbody component using the GetComponent() method and store it in the rb variable. Finally, we add 50 force to the rigidbody in the y-axis direction using the AddForce() method.
Copy to Clipboard
Close

AI Assistant Window

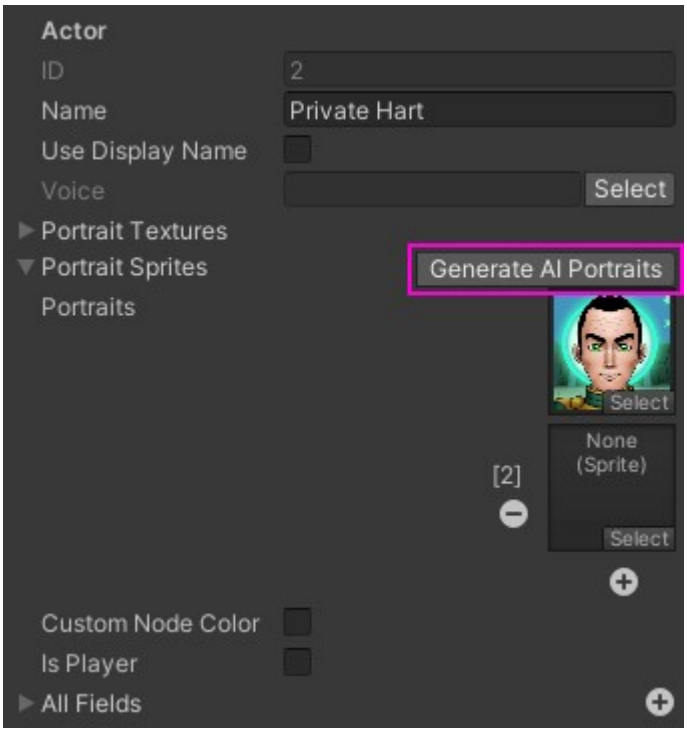


Access the AI Assistant window from the Welcome Window or the Tools menu. This window also operates similarly to ChatGPT and may be handy to dock in your editor alongside other windows, including the Main window.

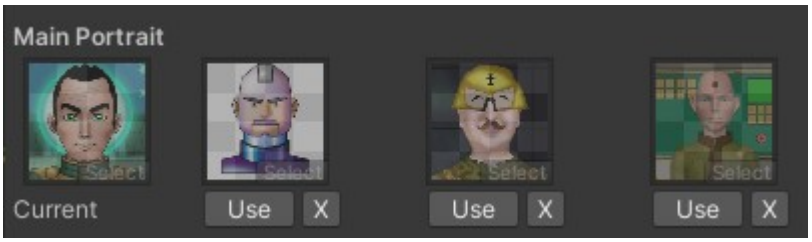
Generate Portraits



To access the Generate Portraits window, inspect an actor in the Dialogue Editor and click **Generate AI Portraits**:



Under **Main Portrait**, the actor's current portrait appears on the left. If you have generated new potential images, they will be shown to the right of the Current image:



To use one of these potential images as the main portrait, click the **Use** button. You can then save the image as an asset in your project, and it will be assigned to the actor.

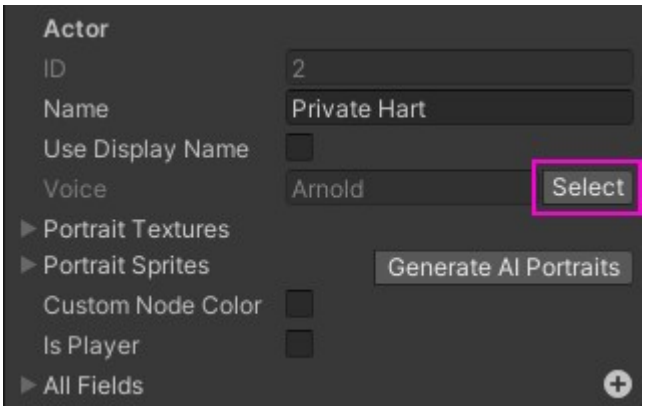
To add more portraits to the actor's Portrait Sprites list, click **Add Portrait**.

For each additional portrait, you can enter a **Variation Prompt**. Then click Generate Portrait to generate a variation of the actor's main portrait image based on the prompt.

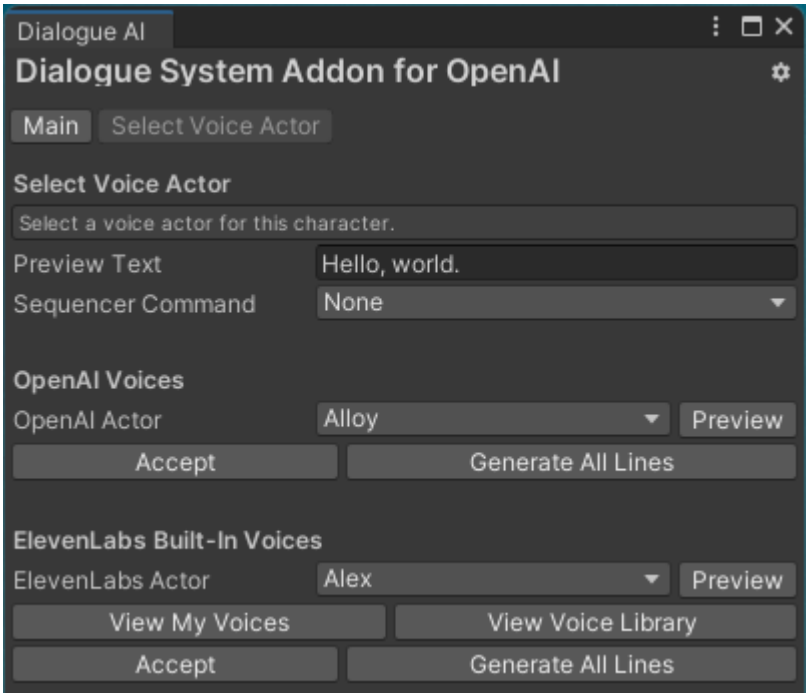
Voiceover Audio Generation

You can use OpenAI, ElevenLabs, Overtone, or DeepVoice speech synthesis API to generate voiceover audio for your dialogue. If you want to use ElevenLabs, first create a profile on https://elevenlabs.io/ and create an API key. Then enter this API key in the addon's Welcome window. Then you can assign voices to actors and generate audio for dialogue entries.

Assign Voices To Actors



In the Dialogue Editor, inspect an actor and click **Select** next to the Voice field.



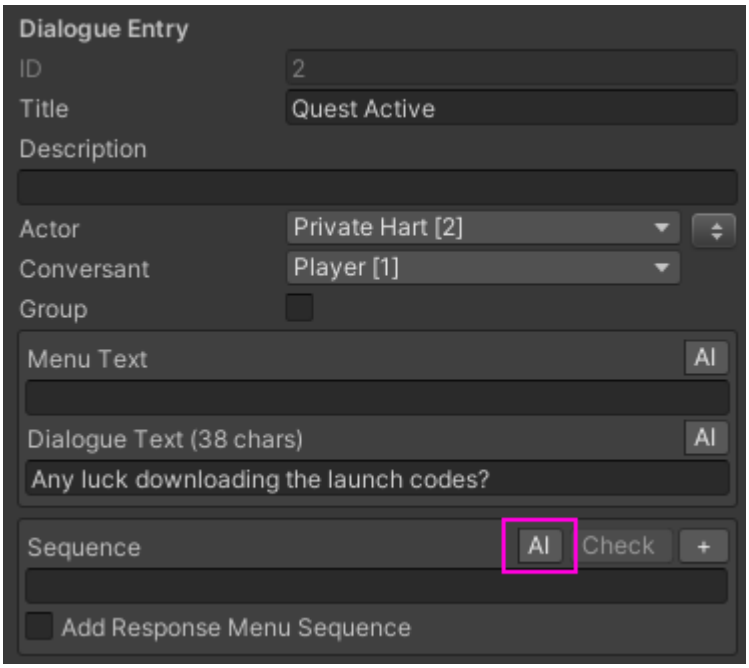
Then select an AI voice actor from the any of the dropdowns. To preview what the voice actor sounds like, enter some preview text and click **Preview**. To assign the selection to your actor, click **Accept**.

For ElevenLabs, you can also click **View My Voices** or **View Voice Library** to add more voices.

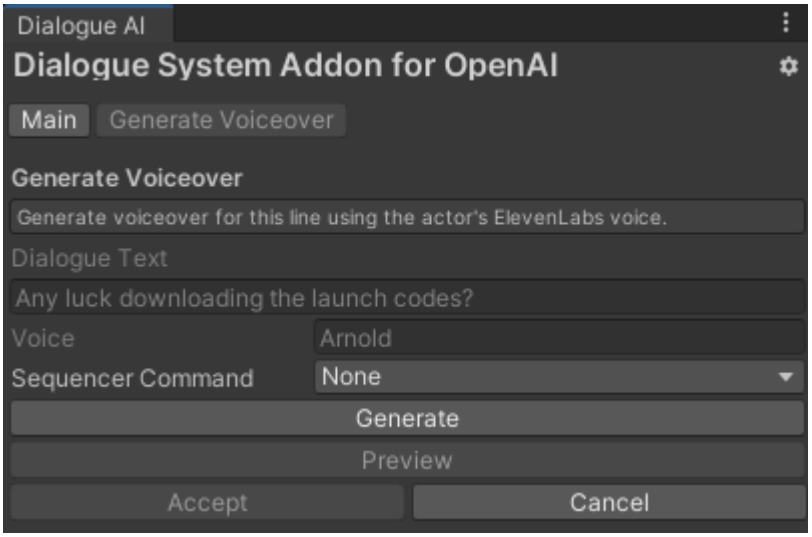
To generate voiceover audio for all of this actor's lines, click **Generate All Lines**. You can also generate voiceover for individual dialogue entries as shown below.

Generate Voiceover Audio

To generate voiceover audio for a dialogue entry, click the Sequence field's **AI** button:



This will open the Generate Voiceover window:



If you want to add a sequencer command such as AudioWait() or SALSA() to the dialogue entry, set the **Sequencer Command** dropdown. Note: It's more common to leave the Sequence field blank and use the Dialgoue Manager's Default Sequence.

Click **Generate** to use the actor's selected voice and API to generate voiceover audio. When the operation is done, it will play the audio. If you want to hear it again, click **Preview**. To accept the audio, click **Accept**. Save the audio file as an asset in your project. Some notes on saving audio: (see next page)

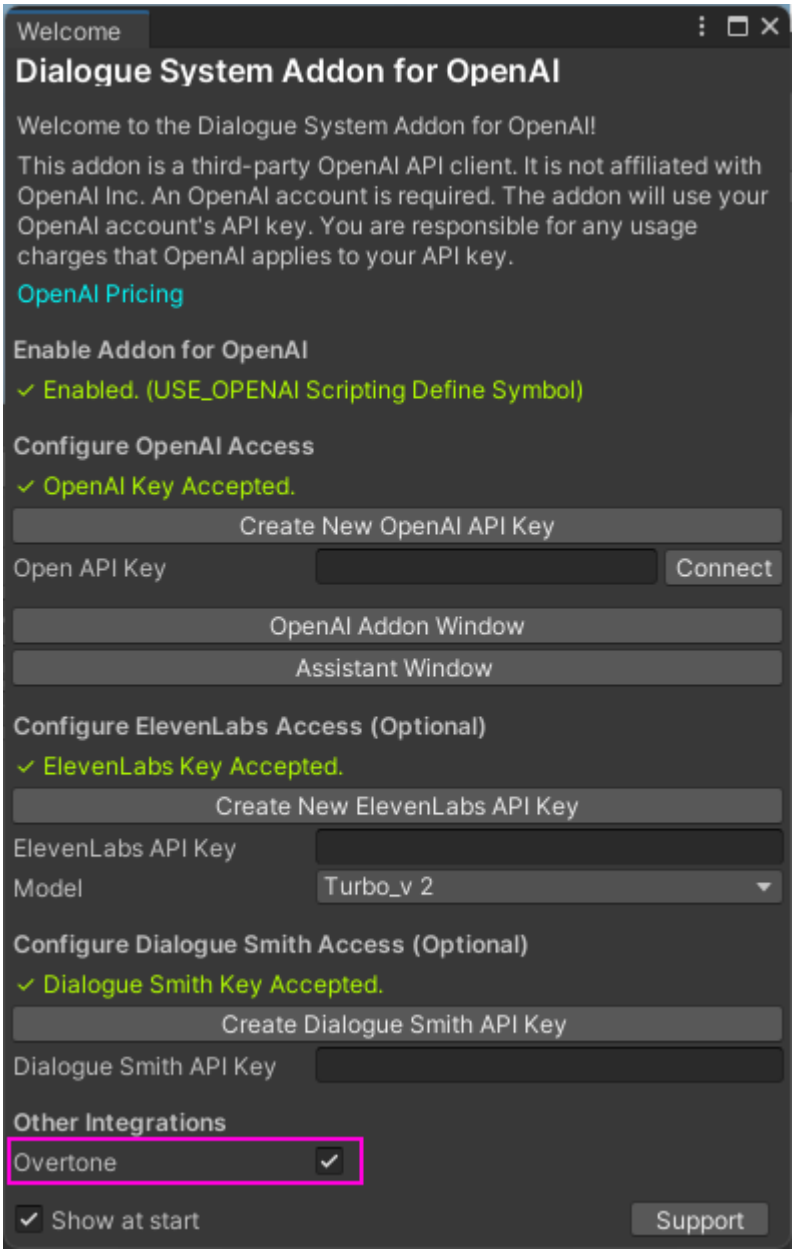
Notes on Saving Audio Files

- If the open scene has a Dialogue Manager, the default filename will be entrytag name for the dialogue entry as specified by the Dialogue Manager's Entrytag Format selection. Otherwise it will use the default entrytag format: ActorName_ConversationID_EntryID.
- If you've enabled Addressables support by clicking the Dialogue System's Welcome Window > USE_ADDRESSABLES checkbox, the audio file will be marked Addressable, and the addressable key will be set to the entry's entrytag.
- Otherwise save the file in a Resources folder.
- Clicking **Accept** will *not* add a sequencer command to the entry's Sequence field. To handle it manually for a single entry, you can drag the audio file into the Sequence field. However, a common way to set up voiceover is to set the Dialogue Manager's Default Sequence to something like: `AudioWait(entrytag)` and leave the Sequence field blank.

Overtone Integration

To enable integration with Least Squares' Overtone:

1. In the Welcome Window, tick the **Overtone** checkbox:



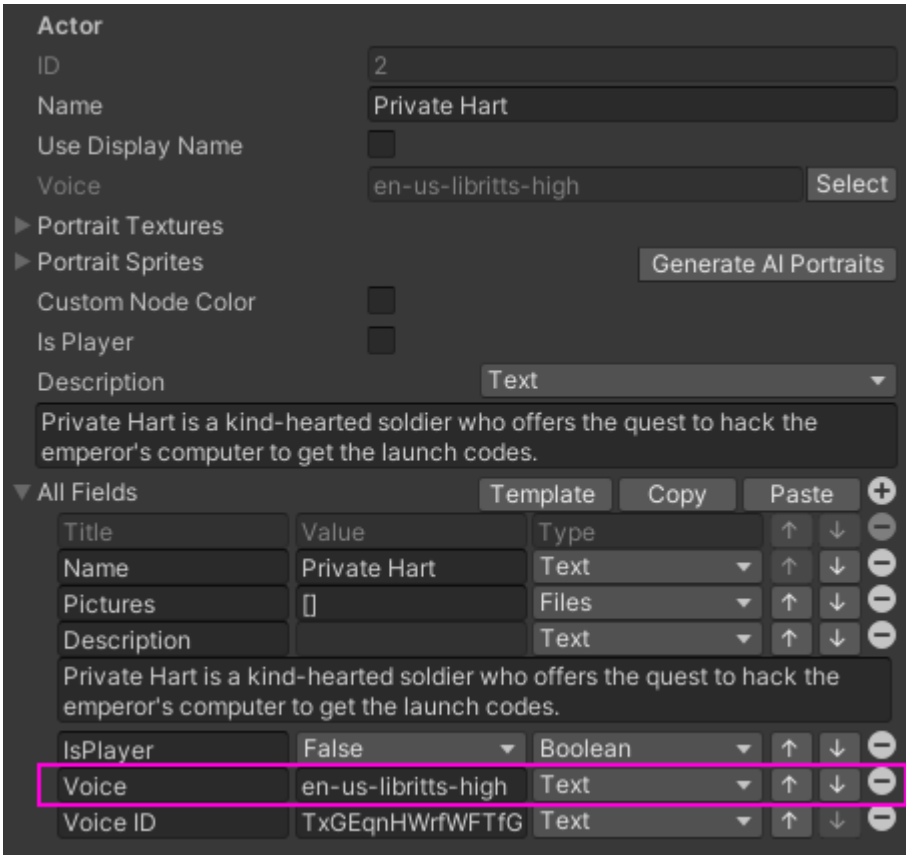
2. From the Third Party Support folder, import the Overtone Integration.unitypackage file. This will add Overtone Integration files to Assets / Pixel Crushers / Dialogue System Addon for OpenAI / Overtone Integration.

Overtone Integration Demo

The Runtime Demo Overtone scene demonstrates a Text Input Conversation and Pirate Adventure CYOA using Overtone. To play it, you must first add your OpenAI API key to the **Runtime AI Conversation Settings** component located on the Dialogue Manager's Canvas > Basic Standard Dialogue UI.

Assign Overtone Voices To Actors

Overtone voices are located in Overtone's Resources folder. To assign an Overtone voice to an actor, assuming the Welcome Window's **Overtone** checkbox is ticked, use the same Voice button in the Dialogue Editor's Actors inspector that you would for ElevenLabs. Alternatively, expand the actor's All Fields foldout and manually specify the voice name in the actor's **Voice** field. Create the Voice field if necessary.



Note: Overtone does *not* use the Voice ID field. It may be present if you were previously using ElevenLabs, but it will not be used by Overtone.

Overtone Scene Setup

As required for any scene that uses Overtone, your scene must have a GameObject with Overtone **TTSEngine** and **TTSPlayer** components.

Overtone Runtime Text To Speech

To play Overtone text to speech in runtime conversations, add a **Runtime Overtone** component to the same GameObject as your **Runtime AI Conversation Settings** component (typically the dialogue UI). Dialogue entries will uses the voices assigned to actors in *Assign Overtone Voices To Actors* above.

Overtone Barks

To play Overtone text to speech with barks, add an **Overtone Barker** component to your barking character. Barks will uses the voices assigned to actors in *Assign Overtone Voices To Actors* above.

Overtone Sequencer Command

To play Overtone text to speech in dialogue entries, use the sequencer command `Overtone()`. This sequencer command has no parameters.

Overtone Lua Function

To use Overtone from a Lua function, add an **Overtone Lua** component to your Dialogue Manager. Then use:

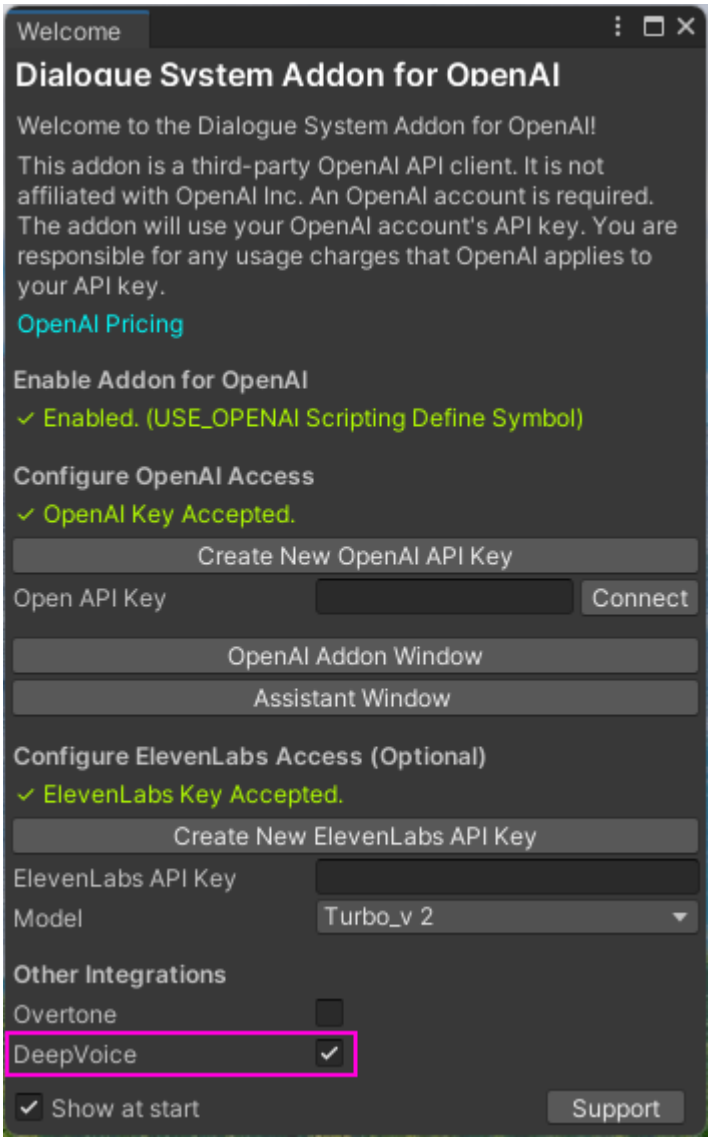
```
Overtone("actor", "text")
```

where *actor* is the name of an actor in your dialogue database and *text* is the text to speak.

DeepVoice Integration

To enable integration with AiKodex's DeepVoice:

1. In the Welcome Window, tick the **DeepVoice** checkbox:



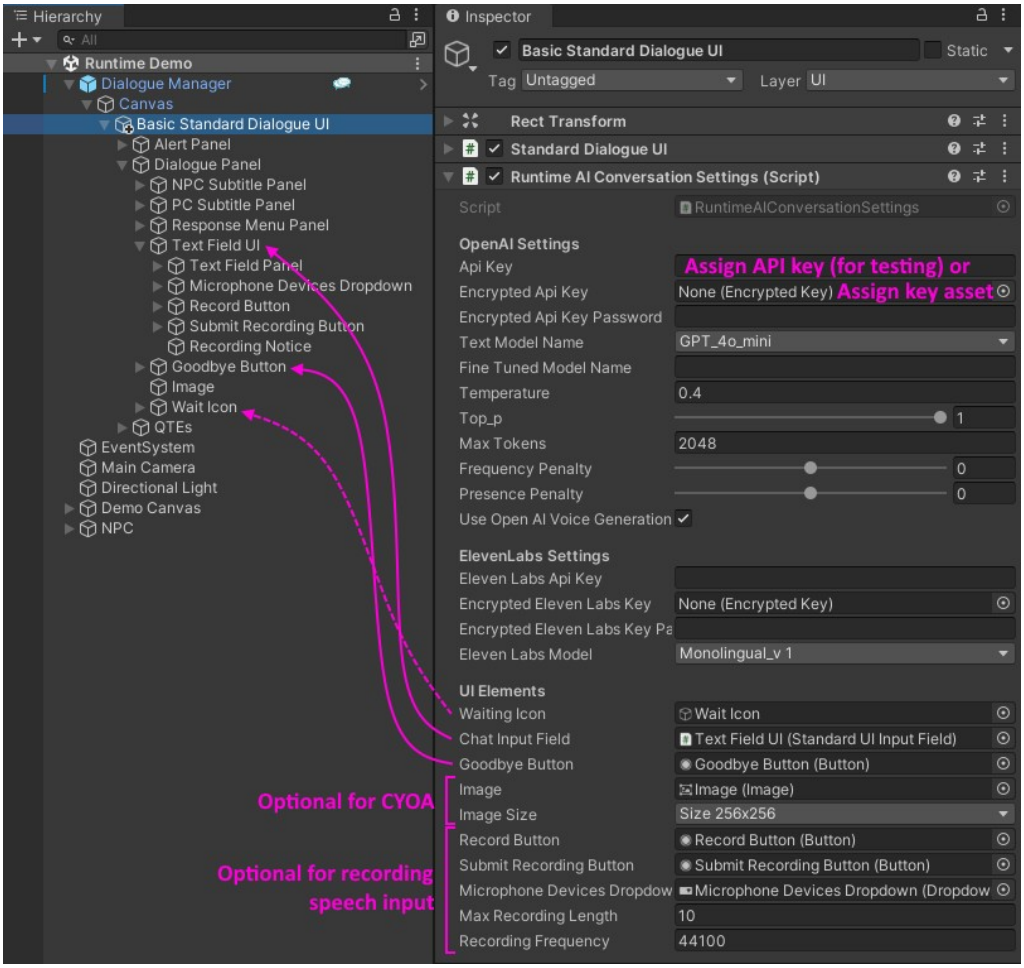
2. In the Dialogue Editor window, inspect an actor. Next to the **Voice** field, click **Select**. The OpenAI Addon window will show a section to select a DeepVoice model and voice. To generate voiceover audio for all lines assigned to this actor, click **Generate All Lines**.

3. Inspect a dialogue entry. Next to the **Sequence** field, click **AI**. Adjust the **Variability** and **Clarity** if you want. This will open the OpenAI Addon window. Click **Generate** to send a request to DeepVoice to generate an audio clip.

Runtime Dialogue

The Dialogue System Addon for OpenAI can also use the OpenAI API at runtime to generate conversations and barks. Runtime operation is entirely optional. If you decide to use runtime generation, consider how you will handle API keys. To avoid excessive charges, it's probably best to require end users (e.g., players) to provide their own API keys. See the discussion

To configure runtime generation, add a **Runtime AI Conversation Settings** component to your dialogue UI:



Add three UI elements to your dialogue UI, and assign them to the component:

- **Waiting Icon**: A GameObject that will be shown while waiting for a response from OpenAI. For example, this could be an animated hourglass or a spinning ball.
- **Chat Input Field**: Input field where the player will enter freeform text responses if running a freeform text input conversation. You can usually assign the existing Text Field UI here.
- **Goodbye Button**: Button that the player can click to end the conversation instead of entering text into the Chat Input Field. You do not need to assign anything to its OnClick() event.

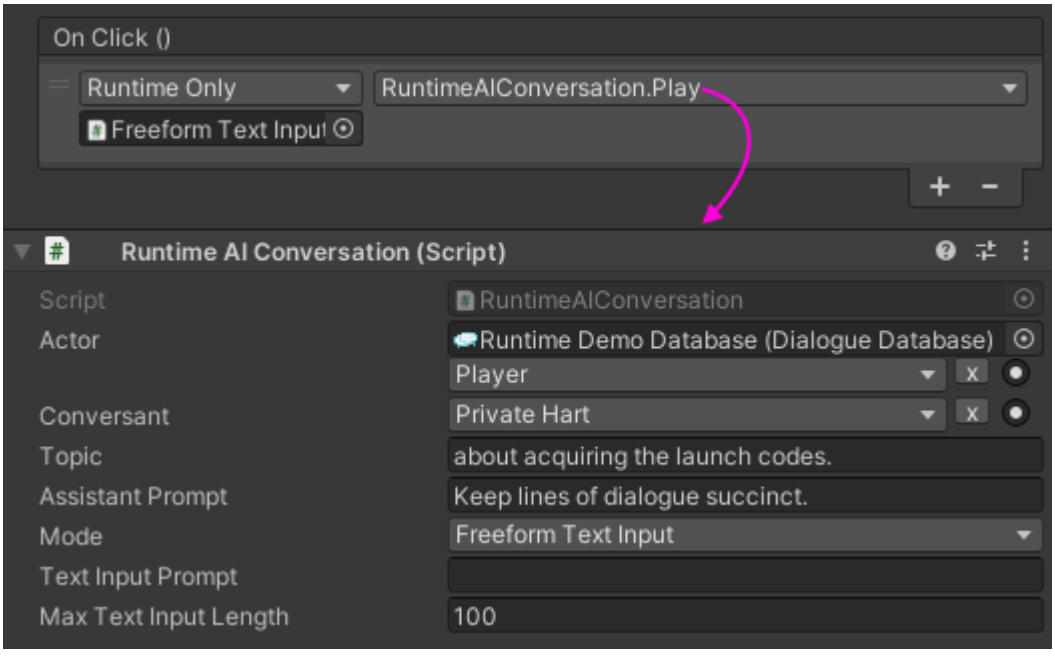
You can also add these optional elements:

- **Image & Image Size**: Used to show generated images in CYOA (Choose Your Own Adventure) mode.
- **Record Button**: Starts recording speech input.
- **Submit Recording Button**: Stops recording speech input and submits the recording to OpenAI for text transcription.
- **Microphone Devices Dropdown**: Dropdown that lets user select which microphone to record.
- **Max Recording Length**: Limits recorded audio clip size to this length in seconds.
- **Recording Frequency**: Records from microphone at this frequency.

While you can enter your API keys directly in the settings component for quick internal testing, embedding your API key in a public release like this is not secure. Prior to public release you should:

1. Remove the API key(s) from the settings component.
2. Retrieve API keys from your own secure server at runtime or create Encrypted Key assets and assign them to the settings component. To create Encrypted Key assets, see Securing API Keys.

Then add a **Runtime AI Conversation** component to your scene. This component holds the configuration for a single conversation or bark. To activate it at runtime, call its Play() method. For example, you can add it to an NPC with a Usable component, and configure the Usable's OnUse() event to call RuntimeAIConversation.Play. In the screenshot below, the Runtime AI Conversation component's Play() method is invoked by a UI Button's OnClick() event:



Set these fields:

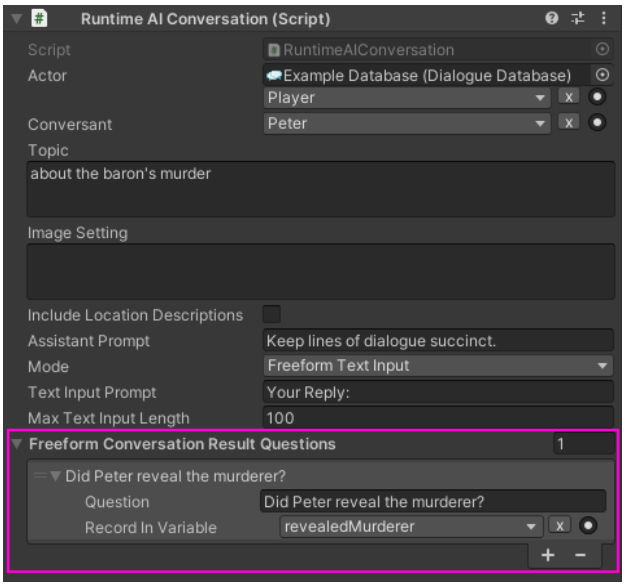
- **Actor**: For conversations, typically the player. For barks, the barking character.
- **Conversant**: The other participant, typically an NPC.
- **Topic**: What the conversation or bark is about.
- **Assistant Prompt**: Optional guiding instructions for OpenAI.
- **Mode**: Can be:
  - *Freeform Text Input*: Plays a conversation line by line, generating responses to each line of player input. The player can end the conversation by clicking the Goodbye button. The NPC will end the conversation if the conversation history approaches Max Tokens.
  - *Response Menu*: Generates a complete conversation up-front and runs it like a prewritten conversation using the regular player response menu and/or player subtitle panel.
  - *Bark*: Generates and plays a single bark through the Actor.
  - *CYOA*: Plays a Choose-Your-Own-Adventure style story complete with illustrations.
- **Text Input Prompt**: Only used in Freeform Text Input mode. The label shown in front of the Chat Input Field. Tip: To keep conversations on topic, you can include instructions similar to: "If the player goes off topic, reply with a reminder to keep the discussion on the original topic."
- **Max Text Input Length**: Only used in Freeform Text Input mode.

Adding Runtime Context

Note: You can use the markup tags `[var=`*variable*`]` and `[lua(`*code*`)]` in your Topic. This allows you to change information in the topic text at runtime based on Dialogue System variable values and/or the values of Lua expressions.

Recording Conversation Results Back Into Dialogue System

When playing Freeform Text Input conversation, you can provide questions in the **Freeform Conversation Result Questions** section:



At the end of the conversation, the addon will pose these questions to OpenAI and record the answers in the variables you specify.

Runtime Demo

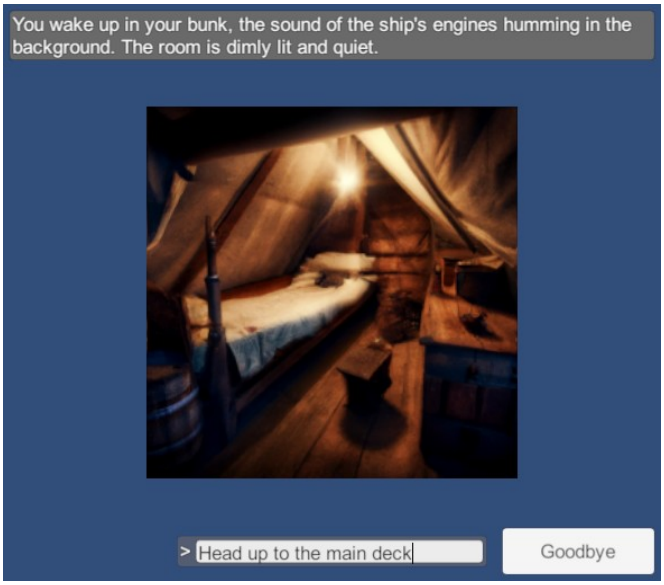
The Demo folder contains a bare bones scene that demonstrates all five types of runtime dialogue generation:

- **Freeform text input conversing with an NPC**. This conversation is generated line by line. If you've specified an ElevenLabs API key in your dialogue UI's Runtime AI Conversation Settings component, it will also generate voiceover audio immediately after generating each line of dialogue. You can use text input or microphone speech input.
- **Response menu-based conversation**. Demonstrates a conversation between an NPC and the player using the dialogue UI's response menu. The conversation is generated first and then played. Voiceover audio is generated as each line is played.
- **NPC-NPC conversation**. Demonstrates a conversation involving two NPCs, without player interaction. The conversation is generated first and then played. Voiceover audio is generated as each line is played.
- **Bark**. Demonstrates a generated bark.
- **CYOA**. Choose-your-own-adventure story with freeform text input or speech input.

To play the Runtime Demo, you *must* first enter your OpenAI API key in the Runtime AI Conversation Settings component located on Dialogue Manager > Canvas > Basic Standard Dialogue UI.

You may also enter an ElevenLabs API key if you want the demo to generate voiceover audio. If you've entered an ElevenLabs API key, make sure you've assigned voices to the actors in your dialogue database.

CYOA mode can optionally use DALL-E to generate images for each stage of the adventure. To enabled this, add an Image to the Dialogue Panel and assign it to Runtime AI Conversation Settings.



Securing API Keys

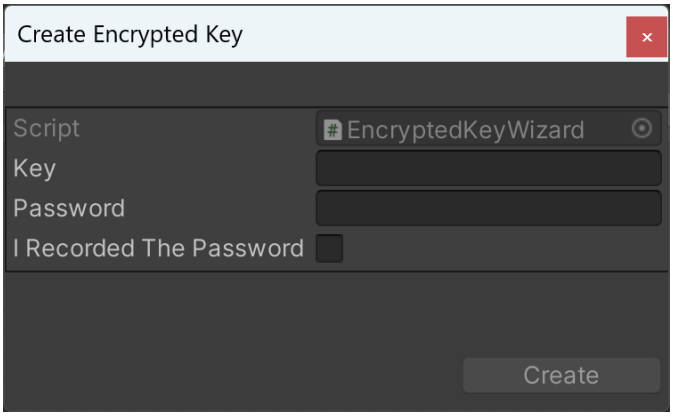
It's convenient to assign API keys directly to the **Runtime AI Conversation Settings** component's **API Key** and **ElevenLabs API Key** fields during development and for in-house builds. But for public builds, such as game builds that you make available to players, you should secure your API keys.

Secure API Keys On Server

The safest way to secure API keys is to store them on your own server, not in the Unity build. At runtime, retrieve the API key from your server and assign it to the property `RuntimeAIConversationSettings.Instance.APIKey` for OpenAI and ElevenLabsApiKey for ElevenLabs.

Encrypted API Key Asset

The Addon also provides an alternative that isn't as secure as storing API keys on your own server but is significantly better than storing plain text API keys in your build. Use menu item **Tools → Pixel Crushers →Dialogue System → Addon for OpenAI → Create Encrypted Key Asset...**.



Enter the API key and a password of your choosing. *Record this password somewhere.* You can't recover it from the encrypted key asset. Then tick the checkbox and click Create.

Finally, assign the key and password to your **Runtime AI Conversation Settings** component's API Key or ElevenLabs API Key fields. While this isn't as secure as retrieving the key from a secure server at runtime, it's significantly better than embedding plain text keys in your build.

Questions? Feature Suggestions?

If you're stuck or have feature suggestions, please contact us. We're here to help!