OpenAI Integration for Unity

User guide

Setup

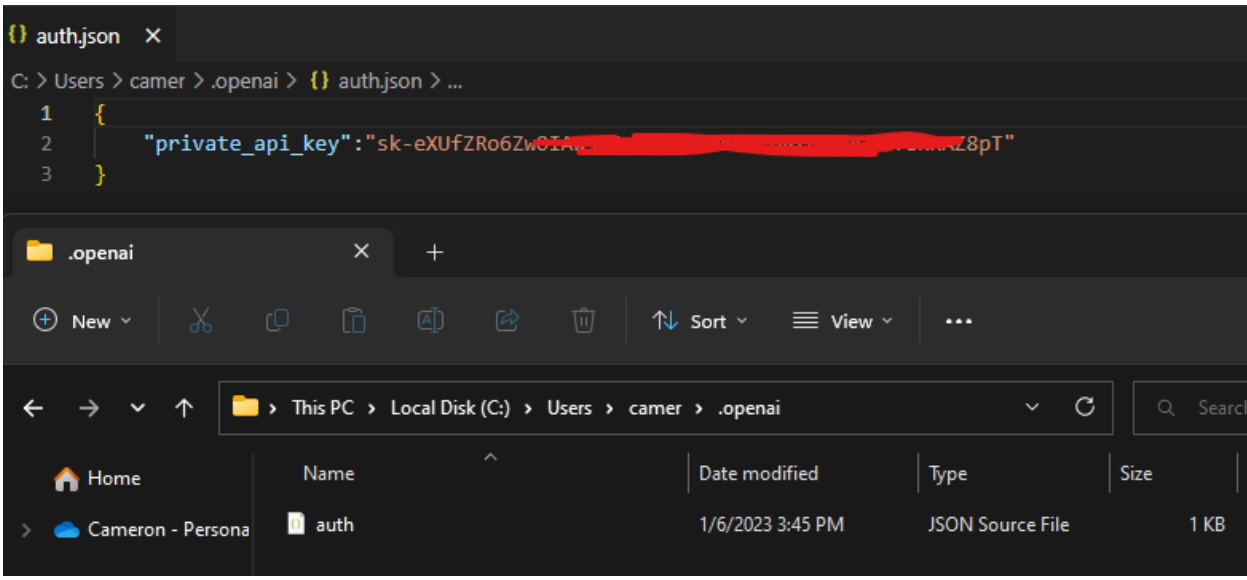
1. Register for OpenAI: openai.com/api
2. Create a new secret key: beta.openai.com/account/api-keys
3. Set your authentication type on the ***DefaultAuthArgsV1*** scriptable object located in ***OpenAi/Runtime/Config/***

   Authentication options:

   a. **Local File - Does not work for in-game API calls.** This authentication method will attempt to find the private key at `***~/.openai/auth.json***` (Linux/Mac) or `***%USERPROFILE%/.openai/auth.json***` (Windows). If this file does not exist or the key is not present, api calls will fail. The JSON file should look like:

   ```
   {
       "private_api_key":"<YOUR API KEY>"
   }
   ```

   Your setup should look like this:



b. **String -** Use this when using API calls in-game. This option allows you to enter your API key directly, but this method isn't safe if your code is exposed publicly (like on GitHub). **Use this option with caution.**

Recommended: Set usage limits in your OpenAI account to prevent accidentally exceeding your budget.
beta.openai.com/account/billing/limits
For more information on OpenAI pricing: openai.com/api/pricing/

Common setup issues:

- **401 Unauthorized** - This means your API key isn't set up correctly. Double check you've entered it correctly, there should be no <> symbols.

- **429 Too Many Requests** - This likely means your OpenAI API account has no billing information setup, or you've sent too many requests. Check your OpenAI API account and verify you have payment information setup on your account.
  This error can also be caused if your 'Max Tokens' setting is set too low in your API request.

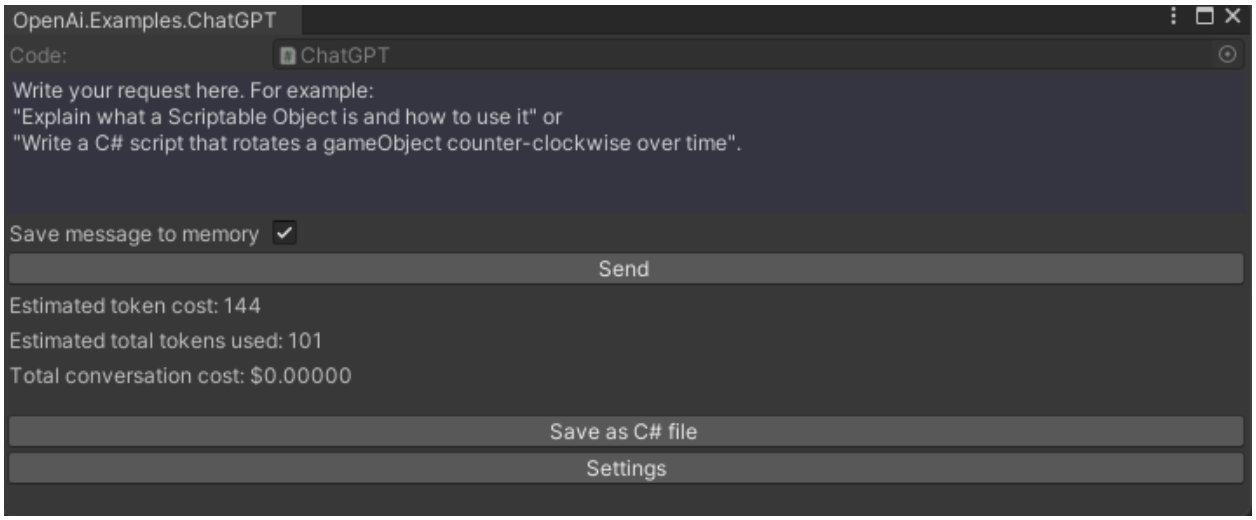
- **Example AI editor tools are blank** - This means your API key isn't setup correctly. Verify the steps above have been completed.

- **Visual Scripting related namespace errors** - If you're using a Unity version older than 2021.x then Visual Scripting isn't included in Unity by default, which will cause these errors. If you wish to use visual scripting, you'll have to import the Bolt package from the asset store and adjust the namespace names. Otherwise, you can safely delete the visual scripting files from this asset, as they aren't required and only exist for demo purposes.

Unity Editor Tools

ChatGTP

This tool functions similarly to OpenAI's own ChatGPT, except you can customize the settings to fine tune the responses you receive. By default, this tool is configured to act as a Senior Unity Game Developer, but you can change this in the settings.



*You can access this tool in Unity by going to **Tools > OpenAI > ChatGPT***

How To Use

Simply enter your request in the text box and then press the send button. If you've chosen for the AI to only return code, you can use the **Save as C# file** button to create a file in your project with the name of the scripts class that was generated in the code. Saving the message to memory will also save the AI's response to memory. **Disable this once you've provided enough context to the AI to prevent older messages from being forgotten and save on token cost.**

Examples:

- *How can I create a custom Unity Editor window that has a settings button, and when clicked it reveals 5 more settings?*

- *Write me a Unity C# script that can be used as a loot table, rewarding the player with different items based on an assigned weight.*
- *Help me come up with a backstory for my game. The main characters name should be Cameron and he travels the world collecting elemental powers.*

Settings

- **Model** - The model to use for completion. See beta.openai.com/docs/models for a list of available models.

- **Max tokens** - The maximum number of tokens to generate. Requests can use up to 4000 tokens shared between prompt and completion. (One token is roughly 4 characters for normal English text)

- **Temperature** - Controls randomness: Lowering results in less random completions. As the temperature approaches zero, the model will become deterministic and repetitive.

- **Top P** - Controls diversity via nucleus sampling: 0.5 means half of all likelihood-weighted options are considered.

- **Stop** - Where the API will stop generating further tokens. The returned text will not contain the stop sequence.

- **Frequency penalty** - How much to penalize new tokens based on their existing frequency in the text so far. Decreases the model's likelihood to repeat the same line verbatim.

- **Presence penalty** - How much to penalize new tokens based on whether they appear in the text so far. Increases the model's likelihood to talk about new topics.

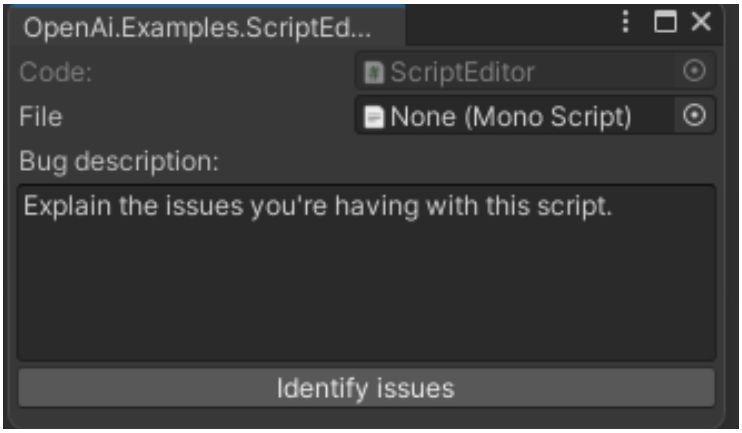
- **Output code only** - Prompts the AI to only respond with code. Use this if you plan on saving the output to a file.
  Note: Selecting this option automatically switches the model to *code-davinci-002* which is currently free in beta and more specialized for writing code.

    - **Add comments to code** - Prompts the AI to add comments to the code it writes to clarify what it's doing.

- **Instruction prompt** - Any text here is added to the beginning of your message to the AI. By default, there is a prompt entered that tells the AI to act as a Senior Unity Game Developer, but you can replace this with anything you'd like.
  Tip: Use the ***Prompt Generator*** (found in ***Tools > OpenAI > Prompt Generator***) to create high quality prompts to try.

Script Editor

This tool allows you to analyze any script in your project using AI, which will take into account whatever message you send it and respond back with suggestions. **You can use this to debug scripts, enhance them, or make general changes.** If you agree with the AI recommendations, you can choose to have it automatically update the file using the "**Apply Recommendations**" button.



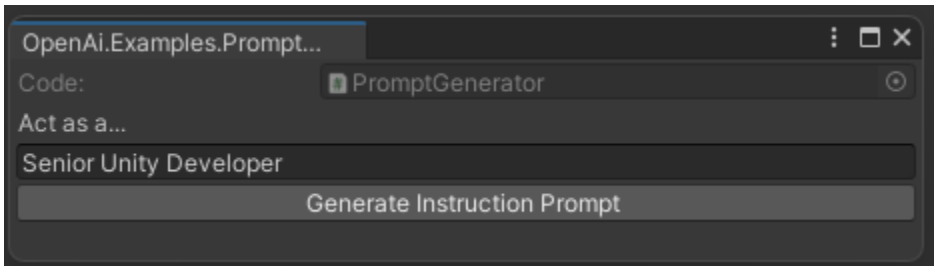
*You can access this tool in Unity by going to **Tools > OpenAI > Script Editor***

How To Use

Simply enter your request in the text box and then press the "**Identify issues**" button. If you'd like to apply the recommendations to the file, press the "**Apply Recommendations**" button to overwrite it.

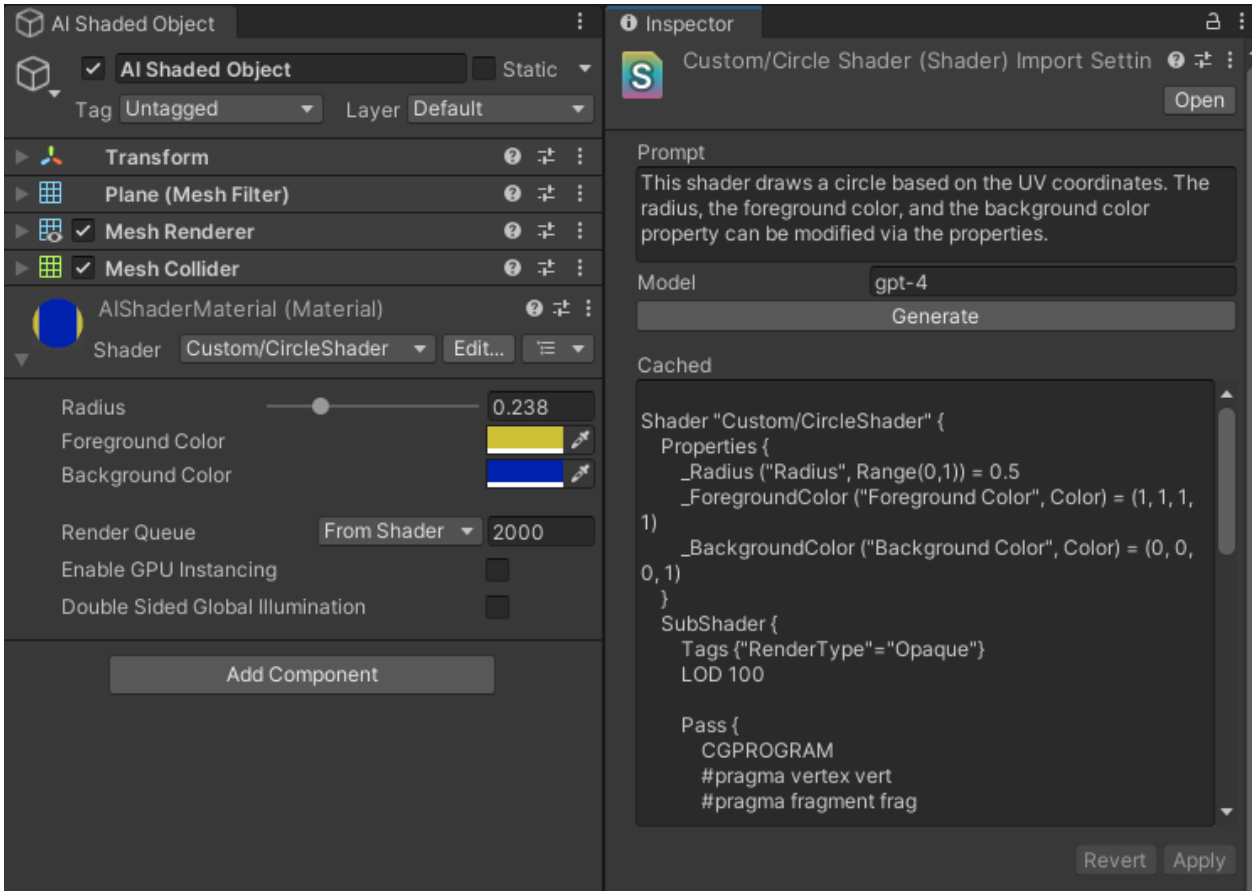
Prompt Generator

The key to getting great results from AI tools is to craft the perfect prompt. This tool will automatically generate a high quality prompt for you, just give it a brief description of how you want the AI to act and it'll give you a prompt template you can edit and use in any other AI tool.



*You can access this tool in Unity by going to **Tools > OpenAI > Prompt Generator***

AI Shaders



This tool uses AI to generate shaders for you! Simply right click in your project files and choose *Create > AI Shader*. Then open the shader in your inspector and describe the kind of shader you want to create and press the **generate** button.

If you like the code and want to see it in action, press the **apply** button to save. Now you can apply this shader to any material and see it in action!

OpenAI + ElevenLabs Voice Chat Demo

Set your ElevenLabs API key in the ELAuthArgs file located in GPT AI Integration > ElevenLabs > Runtime > Config. Your API key can be located in your account settings on the ElevenLabs website.

Be careful to make sure this file is not exposed publicly or else your API key will be at risk.

This demo works by transcribing your recorded audio with the [OpenAI Whipser API] and feeding that text into GPT 3.5. You can then take the response you receive back and generate a voice from ElevenLabs.

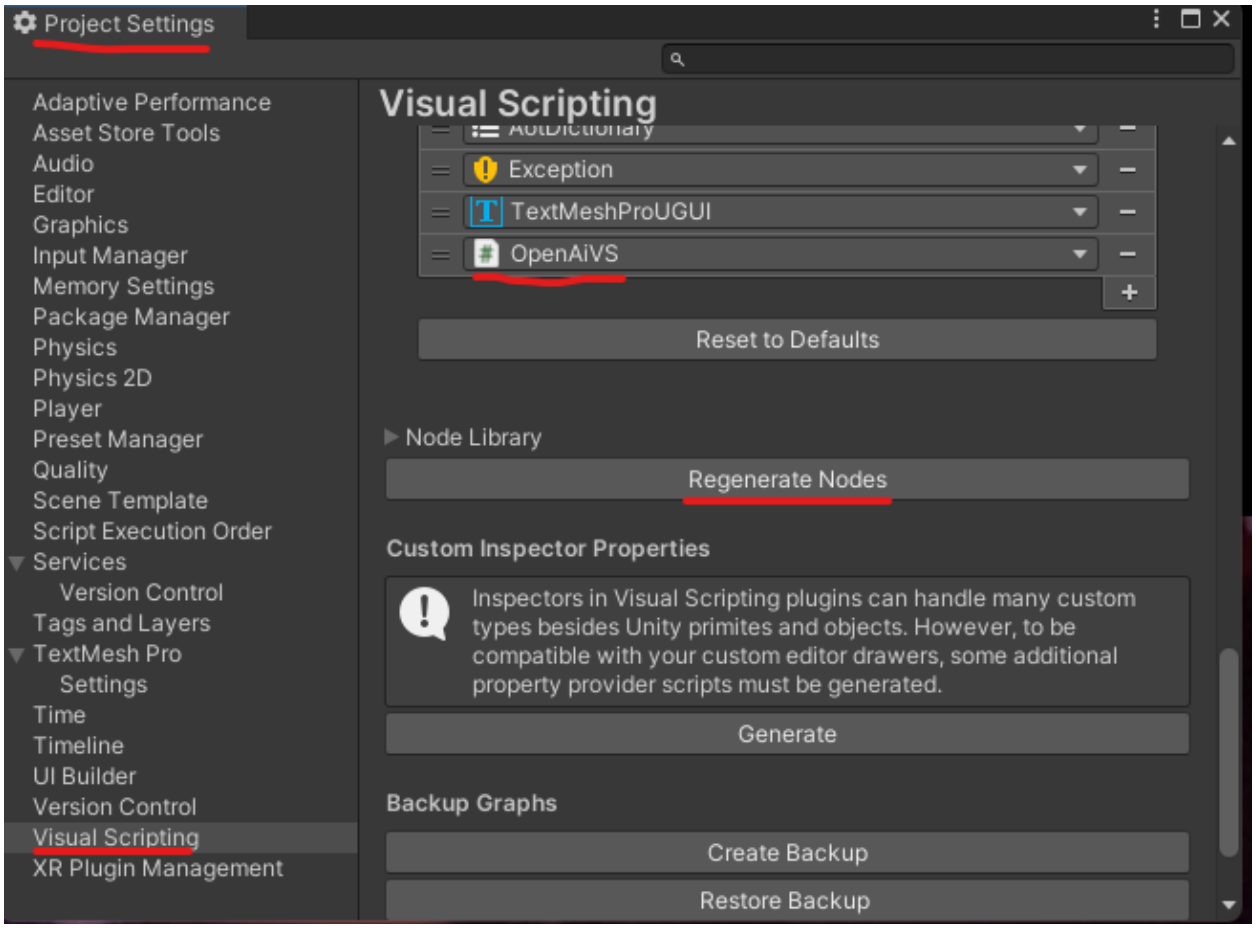
To change the voice returned, edit the ELSpeaker.cs file *(located in ElevenLabs > Runtime > Config)*:

```
private const string url =
"https://api.elevenlabs.io/v1/text-to-speech/21m00Tcm4TlvDq8ikWAM";
```

The last part of this URL represents the voice (`21m00Tcm4TlvDq8ikWAM`) which you can change to get a different voice response. Here are the list of voice ID's:
https://api.elevenlabs.io/v1/voices

Visual Scripting Demos (Unity 2021.x+)

1. To use the visual scripting features, you'll first have to goto ***Project Settings > Visual Scripting*** and initialize if you haven't already.
2. **Add OpenAiVS to your node types** and then press the "**Regenerate Nodes**" button.



In-Game AI Chat

Scenes > Visual Scripting - Chat Demo

The **Controller gameObject** has the script graph containing all of the logic. It also contains the **OpenAiVS script** which is **required** on any gameObject using OpenAI with visual scripting.

Simply start play mode and enter in any message to the AI. You can control the settings of the AI from within the **OpenAiDemo script graph** that's attached to the controller.

In-Game Language Translation

Scenes > Visual Scripting - Translate Demo

The **Controller gameObject** has the script graph containing all of the logic. It also contains the **OpenAiVS script** which is **required** on any gameObject using OpenAI with visual scripting.

Simply start play mode and enter in the language you'd like to translate your messages to, then write a message to send. You can control the settings of the AI from within the **OpenAiDemo script graph** that's attached to the controller.

Using the API with C#

Checkout the Voice Demo scene for an all C# example of how to use this assets AI integrations. You can also see OpenAIDemo.cs and OpenAIVS.cs for examples of how to use C# to call the API in-game.

To call the API in the Unity editor, check the ChatGPT.cs, PromptGenerator.cs, and ScriptEditor.cs files for examples.

Keep in mind that the new GPT 3.5 and GPT4 models have a different format of request than all of the other AI models, so they use different methods.

GPT Example method:

```
// Use this method when calling a GPT model
            public async Task<ApiResult<ChatCompletionV1>>
SendChatGPTRequest(string message)
        {
            SOAuthArgsV1 auth = completer.Auth;
            OpenAiApiV1 api = new OpenAiApiV1(auth.ResolveAuth());
            ApiResult<ChatCompletionV1> chatComp = await
api.ChatCompletions
                .CreateChatCompletionAsync(
                    new ChatCompletionRequestV1()
                    {
                        model = "gpt-3.5-turbo",
                        messages = new[]
                        {
                            new ChatMessageV1()
                            {
                                role = "user",
                                content = message
                            }
                        }
                    });

            return chatComp;
        }
```

Other models example method:

```
//Use this method when calling a model other than gpt
                    comp = await
completer._gateway.Api.Engines.Engine(model).Completions.CreateCompletionA
sync(
                        new CompletionRequestV1()
                            {
                                prompt = prompt.text,
                                max_tokens = completer.Args.max_tokens,
                                temperature = completer.Args.temperature,
                                top_p = completer.Args.top_p,
                                stop = completer.Args.stop,
                                frequency_penalty =
completer.Args.frequency_penalty,
                                presence_penalty =
completer.Args.presence_penalty
                            });

                        if (comp.IsSuccess){
                        response.text =
comp.Result.choices[0].text.Replace("\\n", "").Replace("\\\"",
"\"").Replace("\\t", "\t");
                        } else {
                        response.text = $"ERROR: StatusCode:
{comp.HttpResponse.responseCode} - {comp.HttpResponse.error}";
                    };
```

Need more help or have additional questions?

[Join the discord!]