AI Engine

Support

Contact

More Tools

SUMMARY

INSTALLATION

Requirements

- Minimum Unity Version 2021.3 +
- ChatGPT API key

Installation

If you have other tools from Pampel Games installed in your project, make sure to update them to the latest version before installing AI Engine. This will ensure that all tools work together seamlessly.

To install AI Engine, simply use Unity's Package Manager like you would with any other Unity package.

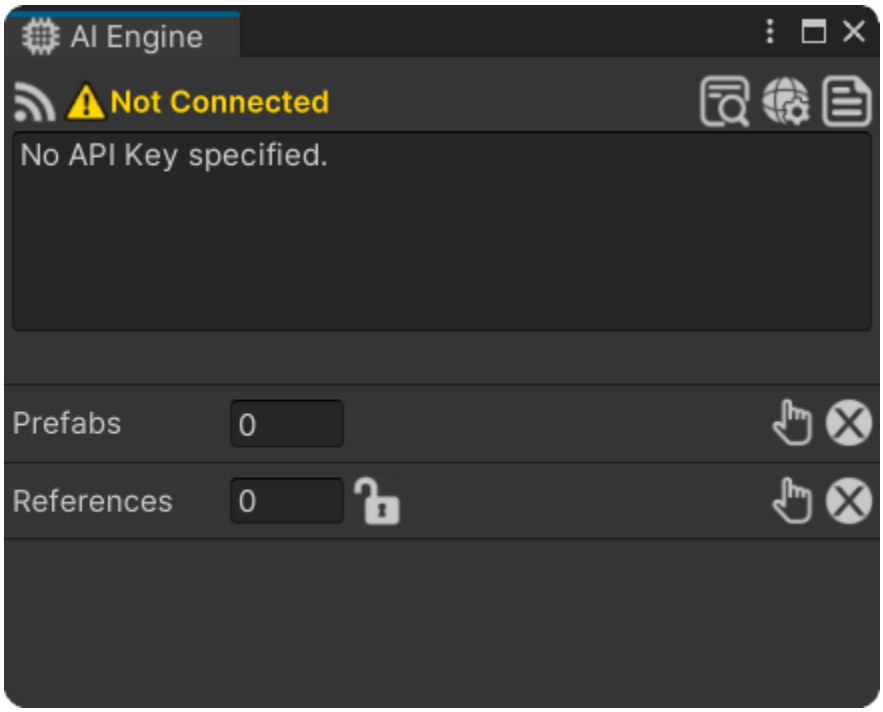
By default, AI Engine will be installed in the 'Assets/PampelGames/AI Engine' directory. However, you can modify the folder names or move the 'PampelGames' folder to a different location within the 'Assets/' folder.

QUICK START

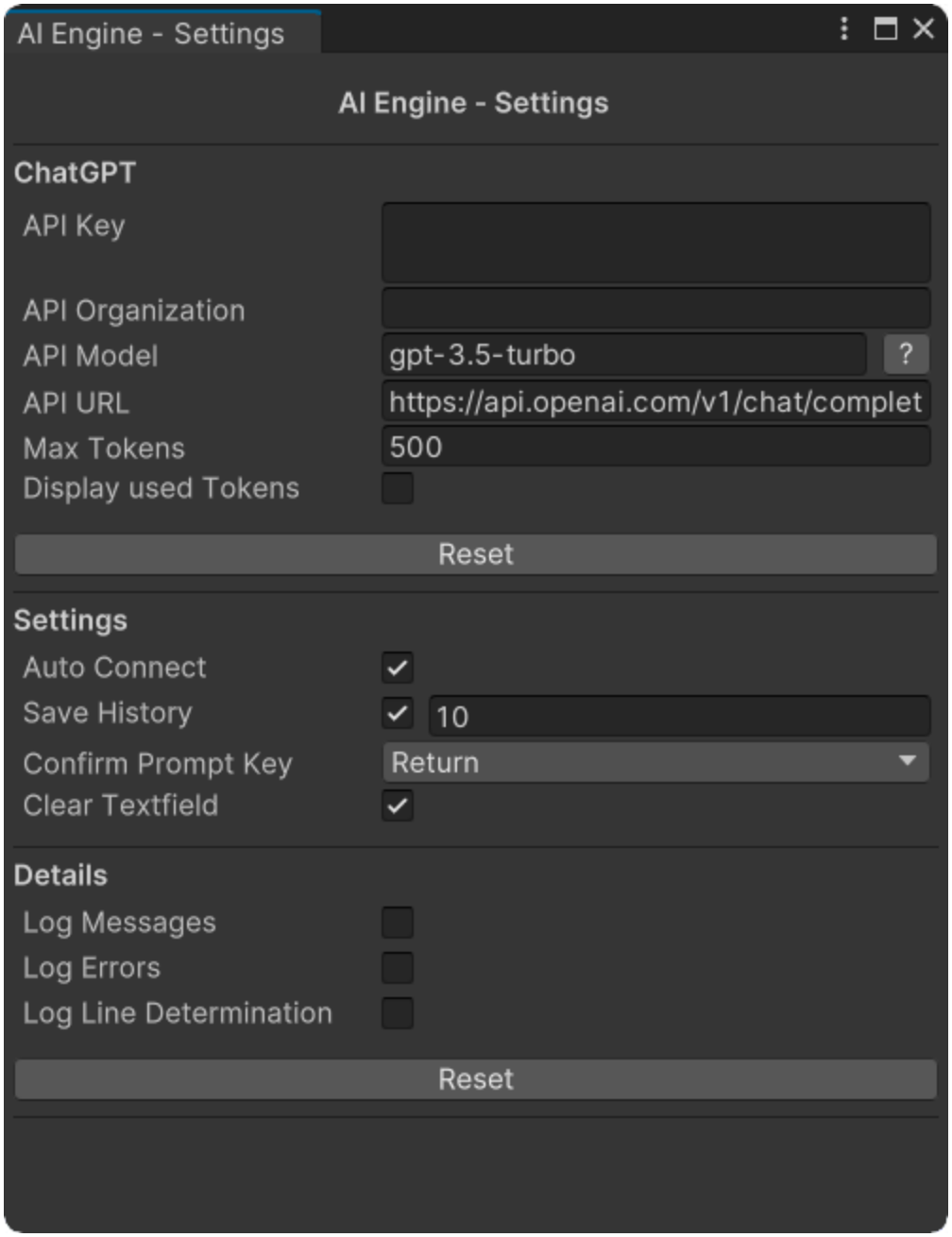
Connecting to OpenAI

To get started, open the AI Engine window by navigating to the top menu bar and selecting Tools > Pampel Games > AI Engine.

By default, the tool will automatically connect to OpenAI. However, when opening it for the first time, an error message reading "Not Connected" will appear.



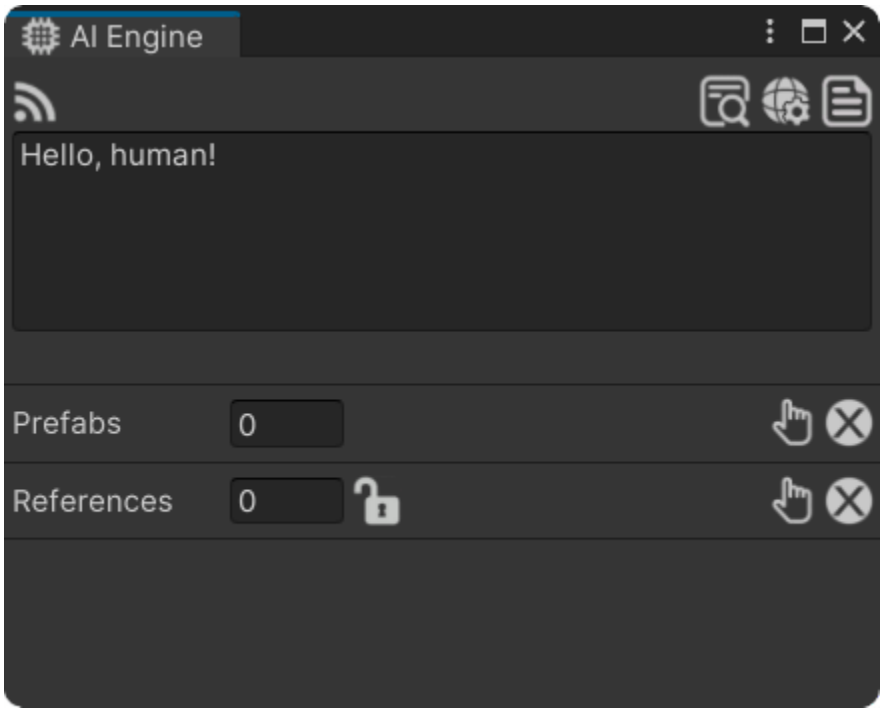
Click on the globe-shaped button on the upper right corner to open the settings window.



If you haven't already, proceed to register on the OpenAI website to obtain an API key. If you register through an organization, ensure that you provide information in the API Organization field as well.

Returning to the AI Engine window, click on the connection icon located in the top left corner. If the setup has been completed correctly, the "Not Connected" status will be replaced by a green checkmark, and the text input field will display a welcome message from the AI.

This process may take a few moments to complete.



If the connection fails to be established, please refer to the console for more detailed information.

Creating a Task

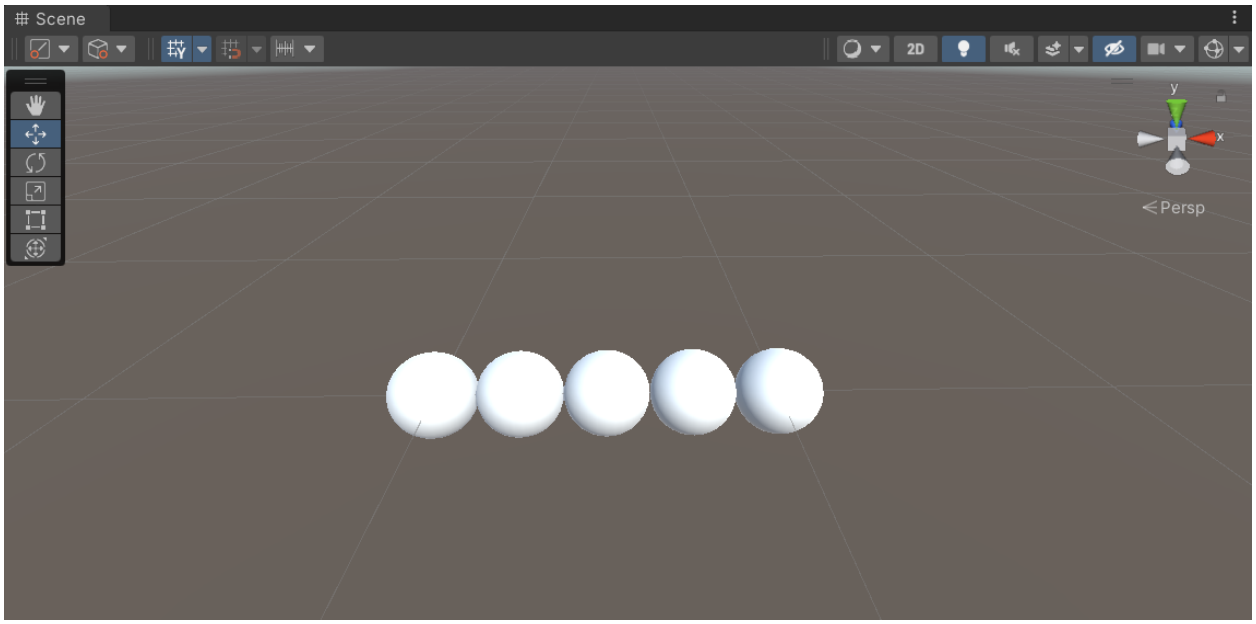
Click with your mouse into the text field and type in a task. For example:

“*Create 5 spheres. Position them equally in a straight line with length 5*”

Press the "Return" key to confirm. Depending on the length of your task as well as the current traffic on the OpenAI website, generating a response may take a few seconds.

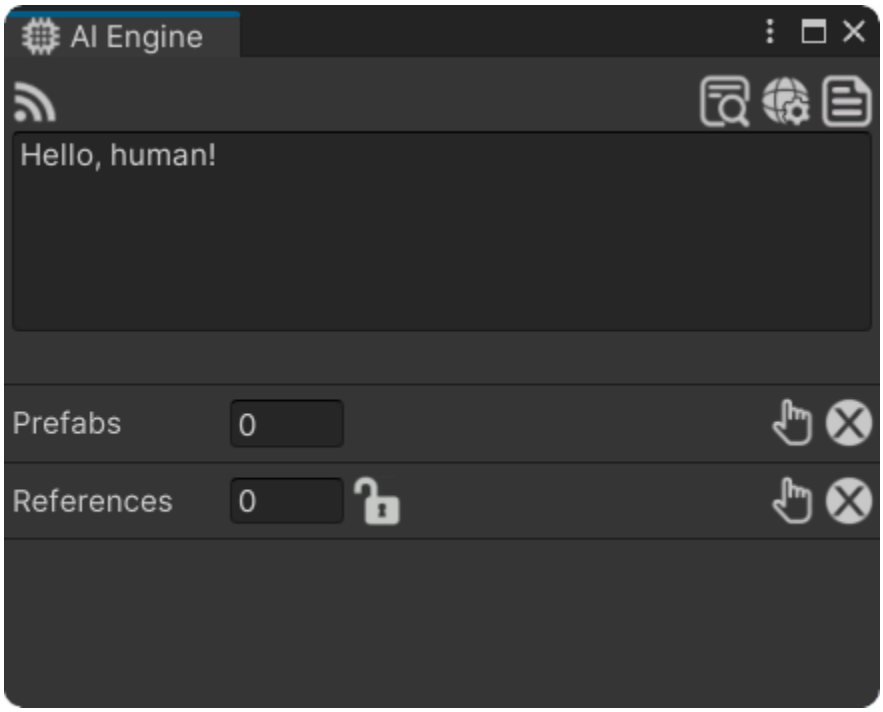
Once the AI has generated a response, the AI Engine will proceed to convert the text into one or multiple functional methods and execute them accordingly.

Congratulations on successfully completing your first task!



AI ENGINE IN DETAIL

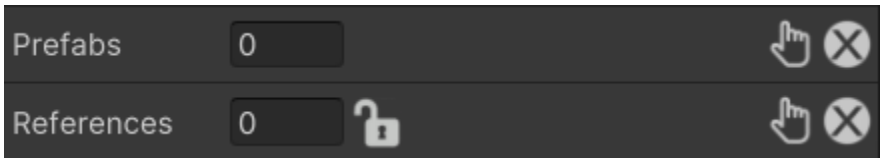
General



As outlined in the Quick Start guide, the first step in using AI Engine is to establish a connection to the OpenAI API via the options icon on the top right corner. Once this is accomplished, you can enter a task into the text input field, which will be transmitted to the AI for processing. After receiving a response, AI Engine will automatically convert the text into functional methods and execute them.

While the internal workings of the algorithm may be rather complex, for end users the general functionality is pretty straightforward. Any buttons or variables in the AI Engine component and its windows that are not self-explanatory have a helpful tooltip that appears when you hover your mouse over them.

References



The AI does not have direct access to your project folders or scenes. This means that if you want to instantiate prefabs from the project folder or modify references from the scene, you will need to specifically reference those GameObjects.

However, it is also possible to let the AI retrieve a reference from your scene by itself.
For example, you could say "*Get the main camera from the scene*" or "*Get the terrain component from the scene*" and then apply modifications on those objects in one go.

Any GameObjects that AI Engine creates will automatically be added to the references as long as they are not being locked via the lock icon next to the references count.
To add prefabs or scene references manually, simply drag them onto the text input field. AI Engine will automatically detect its type and assign it to the corresponding slot.

Hovering over the integer fields will display a detailed list of all references as a tooltip. Additionally, you can select all references using the hand icons located on the right side.

Important:
Please keep in mind that if you have any prefabs or references in the lists and submit a task, those objects will automatically be mentioned with your prompts. To ensure clarity for the AI, it is highly recommended to only reference GameObjects that are actually being used.

When you mention prefabs in your prompts, call them prefab(s). References should be called “ref(s)”.

For example:

“*Instantiate 10 random prefabs*”

“*Add a box collider to all refs*”

LIMITATIONS

AI

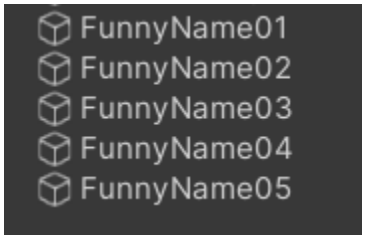
AI Engine's algorithm will fail to generate code if the AI produces inaccurate responses.

Keep in mind that AI-based language models are still in their early stages and even with clear and concise instructions, there is always a possibility that the generated code may not achieve the intended outcome.

Consider the following:

Task:
“*give all objects a funny name*”

Result:



Above that, the AI's response may include code that is intended to be executed at runtime only. For example instead of implementing a loop, it may use Update event functions along with a timer or coroutines.

At this point (ChatGPT model 3.5), it is not practical to direct the AI to avoid Unity event functions or to generate pure editor scripts. Doing so often leads to blown up scripts or the AI may simply disregard the instructions altogether.

To minimize risk of errors and improve AI responses, you can take a look into the Best Practices section.

Code Conversion

AI Engine's algorithm was built from the ground up with the specific purpose of converting AI-generated text responses into functional code. During the conversion process, characters have to be extracted from strings, which in turn can lead to incorrect conclusions.

To provide an obvious example, the AI may refer to a variable as "MonoBehaviour" or use a colon (" : ") when defining a string.

If you receive the following message after a conversion:

You can refer to the history window for additional information. The history window can be accessed by clicking on the left icon located in the top right corner.

Important Note:
If you encounter an AI-generated response that is accurate, but the algorithm is unable to generate the corresponding code, please consider sharing your history with the developers through the discord channel (link on the first page). Thank you!

The following technical limitations currently exist and are planned to be improved in the future if they are found to be practical limitations:

- Nested loops are limited to two loops (one outer loop and one inner loop).
- While loops and Do-While loops are not implemented (mainly as a safety precaution against stack overflows).
- Coroutines are not supported.
- Out ref parameters are only supported for existing methods but not for methods the AI creates (other parameters are supported regardless of the amount). The AI is instructed to only write a single method anyways, but this is not guaranteed.

BEST PRACTICES

Here are some tips to help you make the most of AI Engine.

Formulation

The formulation of prompts by the user is by far the most crucial factor for the successful functioning of AI Engine.

As a general rule, it is best to keep prompts as concise and precise as possible. This does not mean that you should avoid creating long prompts (not at all!). However, it is recommended that you avoid using unnecessarily long sentences and instead focus on clearly specifying what you require in a technical sense.

As mentioned in the References section, it is advised to avoid referencing prefabs or scene objects when they are not required for the task. This is because, as long as they are not empty, AI Engine will automatically mention them.

A few examples to help you phrase your prompts.
(In the end though, only experience will teach you how to formulate your prompts correctly.)

Not great: *"get the terrain from the scene"*
Better: "*get the terrain component from the scene*"

Not great: *"give all refs a funny name"*
Better: "*give all refs funny names*"

Not great: *"rotate all refs to the y + axis"*
Better: "*set the rotation of all refs to the y + axis*"

If you find the AI generates methods that are only executable at runtime (this often happens for long methods that require nested looping), try to directly instruct the AI to use a loop.

Not great: *"check the height of the terrain every 5 meters"*
Better: "*check the height of the terrain every 5 meters in a for loop*"

Workflow

While playing around with the AI Engine tool can certainly be enjoyable, the ultimate reason for using it should be to enhance your workflow and increase your efficiency.

Over time, you will become familiar with the capabilities and discern what tasks can be automated versus what should still be done manually. For instance, it may not be worth waiting for a response just to instantiate a single prefab into the scene.

Where the engine really shines are tasks that require precision and accuracy, as well as those that involve large amounts of data or repetitive actions.

An example for a great use case:

"*Create 100 prefabs. Place them around a circle shape with radius 10 with the center at position 50,100,50. Raycast them to the ground.*"

TIP: Consider integrating AI Engine into your editor interface so that it is readily accessible whenever you need it.