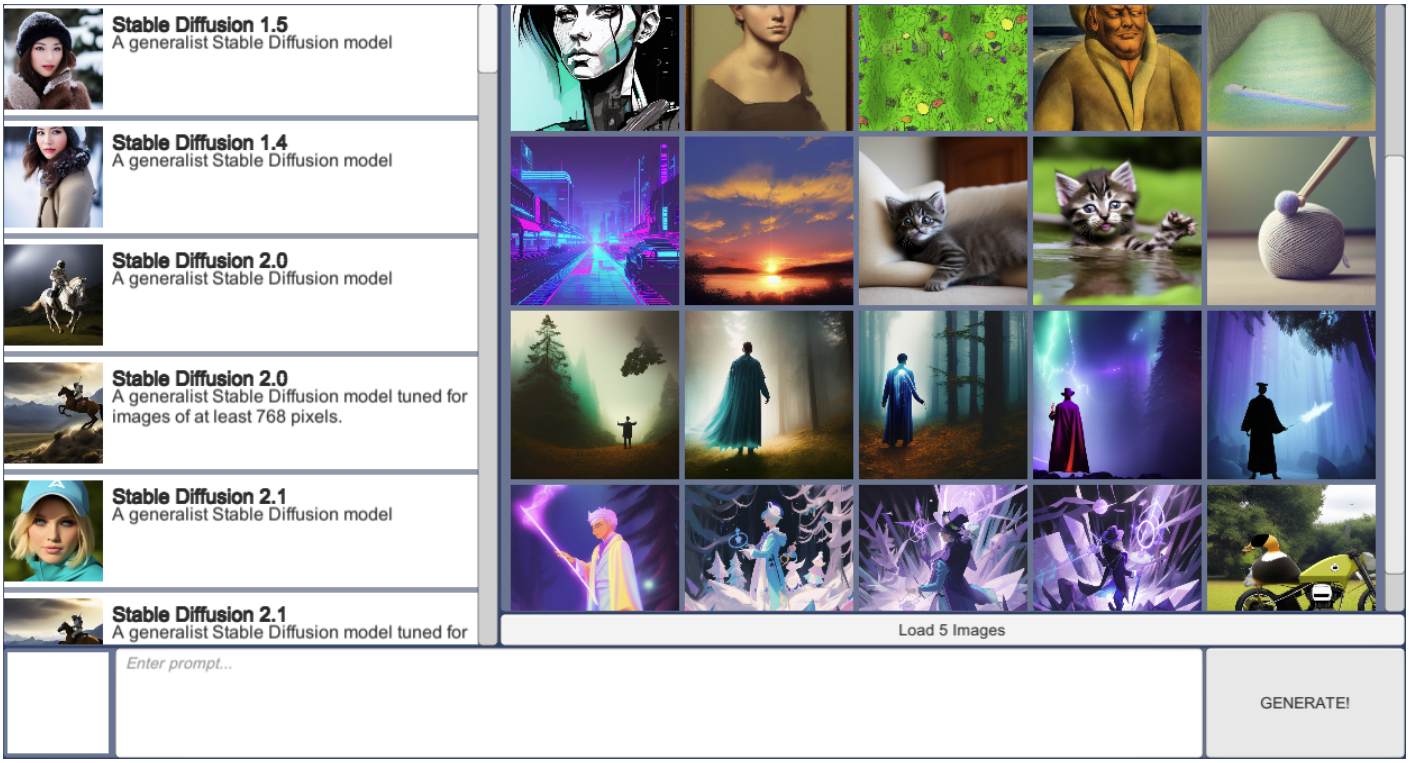
Kindly Integration with Unity Game Engine

Kindly offers seamless integration with the Unity Game Engine, allowing developers to harness the powers of ChatGPT and Stable Diffusion directly within their Unity projects.

This integration opens up a world of possibilities for creating dynamic stories, dialogues and visually stunning images.

In this documentation, we will explore the features of the Kindly Unity Package, including the demo scenes that demonstrates its functionality.

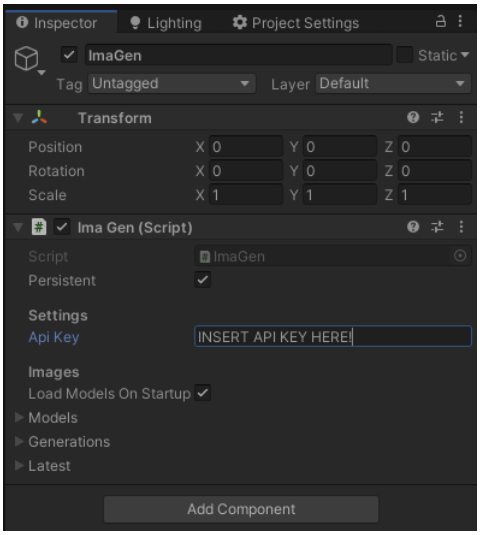


Ready To Start?

If you are ready to start integrating Kindly into your Unity project, head over to the Setup page to get started.

Setup

1. Import the Kindly Unity Package into your project.
2. Create an instance of the Kindly client using your API key for authentication.



2. Open one of the demo scenes in the Kindly package.
3. Press Play and have fun!

Listing Image Generation Models

Models are usually loaded automatically on startup, but you can also download them manually. Models will have the following format:

MODEL CLASS

```
public class Model
{
    public int id;
    public string name;
    public string description;
    public string image_url;
    public string[] triggers;
    public string[] categories;
}
```

Example

```
using UnityEngine;
using MADD;

// Models are automatically downloaded on startup by default, but you can disable this and download them yourself
public class DownloadModels : MonoBehaviour
{
    void OnEnable()
    {
        Kindly.Instance.OnModelsReceived += ListModels;
    }

    private void OnDisable()
    {
        if (!Kindly.Quitting)
            Kindly.Instance.OnModelsReceived -= ListModels;
    }

    public void DownloadModels()
    {
        Kindly.Instance.GetModels();
    }

    private void ListModels()
    {
        foreach (var model in Kindly.Instance._models)
        {
            Debug.Log(model.name);
        }
    }
}
```

Listing Image Generations for a User

Image Generations can be listed using the following method:

```
Kindly.Instance.DownloadImages(int page, int pageSize);
```

They will have the following format:

IMAGE GENERATION CLASS

```
public class Generation
{
    public int id;
    public Image image;
    public string prompt;
    public Model sd_model;
}
```

Example

```
using UnityEngine;
using MADD;

// Downloads 5 images for the user and adds them to the _generations list inside of the Kindly object
public class DownloadImages : MonoBehaviour
{
    public int _page = 1;

    void OnEnable()
    {
        Kindly.Instance.OnImagesReceived += ListImages;
    }

    private void OnDisable()
    {
        if (!Kindly.Quitting)
            Kindly.Instance.OnImagesReceived -= ListImages;
    }

    public void DownloadMoreImages()
    {
        Kindly.Instance.DownloadImages(_page, 5);
        _page++;
    }

    private void ListImages()
    {
        foreach (var image in Kindly.Instance._generations)
        {
            Debug.Log(image.id + " - " + Kindly.Instance.ApiUrl + image.image.url);
        }
    }
}
```

Generating a New Image

Images can be generated using the following method:

```
Kindly.Instance.GenerateImage(string prompt, string negativePrompt, int selectedModelID);
```

And must follow the following format when requesting a new image:

IMAGE GENERATION REQUEST CLASS

```
 public class GenerationRequest
{
    public string prompt;
    public string negative_prompt;
    public int model;
    public string apiKey;
}
```

The resulting image will have the following format:

IMAGE GENERATION CLASS

```
public class Generation
{
    public int id;
    public Image image;
    public string prompt;
    public Model sd_model;
}
```

Example

```
using UnityEngine;
using MADD;
```

```
public class ImageGenerator : MonoBehaviour
{
    public InputField _if;
    public Button _generateBtn;
    public Model _selectedModel;

    private void OnEnable()
    {
        Kindly.Instance.OnImagesReceived += EnableBtn;
    }

    private void OnDisable()
    {
        if (!Kindly.Quitting)
            Kindly.Instance.OnImagesReceived -= EnableBtn;
    }

    public void GenerateStableDiffusion()
    {
        if (_selectedModel == null)
        {
            Debug.LogWarning("You gotta select a model first!");
            return;
        }
        string prompt = _if.text;
        Kindly.Instance.GenerateImage(prompt, "", _selectedModel.id);
        _generateBtn.interactable = false;
    }

    private void EnableBtn()
    {
        _generateBtn.interactable = true;
    }

    private void Start()
    {
        if (_selectedModel == null)
            _generateBtn.interactable = false;
    }

    public void SetModel(Model model)
    {
        _selectedModel = model;
    }
}
```

Generating Text

Text can be generated using the following method:

```
Kindly.Instance.GenerateText(string prompt);
```

And must follow the following format when requesting a text response:

> **TEXT GENERATION REQUEST CLASS**
>
> ```
> public class TextGenerationRequest
> {
>     public string prompt;
>     public string apiKey;
> }
> ```

The resulting text will have the following format:

> **TEXT GENERATION CLASS**
>
> ```
> public class TextGenerationResponse
> {
>     public string status; // can be OK or KO
> ```

```
    public string text;
}
```

Example

```
using UnityEngine;
using MADD;

public class ImageGenerator : MonoBehaviour
{
    public InputField _if;
    public Button _generateBtn;
    public Model _selectedModel;

    private void OnEnable()
    {
        Kindly.Instance.OnTextReceived += PrintText;
    }

    private void OnDisable()
    {
        if (!Kindly.Quitting)
            Kindly.Instance.OnTextReceived -= PrintText;
    }

    public void PrintText()
    {
        Debug.Log(Kindly.Instance._generatedText);
    }

    private void Start()
    {
        Kindly.Instance.GenerateText("Hello world! how are you today?");
    }
}
```

Usage and Rate Limit

Kindly offers a free tier that allows users to generate up to 100 images and 1000 text responses per month.

For more extensive usage, you can choose the paid option, which provides 100 additional image generations and 1000 text generations for every £1.

So, to be clear:

- Free Tier: **100 Images** and **1000 Text** generations per month
- Paid Tier: Additional **100 Images** and **1000 Text** generations for every £1 spent

> **NEED MORE?**
> contact us at Madd Studio to discuss your requirements.

That's All Folks!

The Kindly Unity Package brings the powers of ChatGPT and Stable Diffusion image generation to your Unity games.

With the included demo scene, you can easily explore and test all the functionalities, including downloading and listing image generation models, managing user-specific image generations, and generating new images and text on-the-fly.

Now that you've seen it all, go out and try to integrating in your own projects!

Happy creating! 🎮🖼️