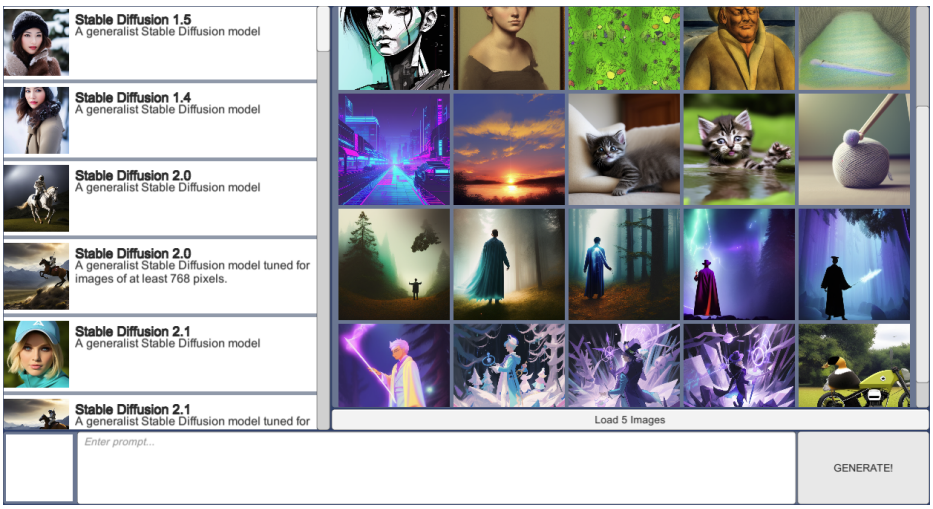
Kindly Integration with Unity Game Engine

Kindly offers seamless integration with the Unity Game Engine, allowing developers to harness the power of Stable Diffusion technology directly within their Unity projects. This integration opens up a world of possibilities for creating dynamic and visually stunning images. In this document, we will explore the features of the Kindly Unity Package, including a demo scene that demonstrates its functionality.



Getting Started

To begin using Kindly within Unity, follow these steps:

1. **Obtain API Key**: If you haven't already, sign up on the Kindly website to obtain your API key. This key will be used to authenticate your requests to the Kindly API.

2. **Install Kindly Unity Package**: Import the Kindly Unity Package into your Unity project. The package includes scripts and assets necessary for seamless integration with the Kindly API.

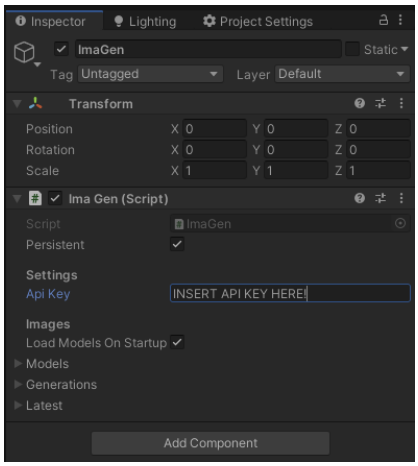
3. **Poke Around in the Demo Scene**: Open the included demo scene to explore and test the Kindly integration. The demo scene showcases the following functionality:

How to Use the Kindly Unity Package

Here's a brief overview of how to access the Kindly functionality within your Unity project:

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. Press Play and have fun!

API Reference

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()
    {
        ImaGen.Instance.OnModelsReceived += ListModels;
    }

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

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

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

Listing Image Generations for a User

Image Generations can be listed using the following method:

```
ImaGen.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
ImaGen object
public class DownloadImages : MonoBehaviour
{
    public int _page = 1;

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

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

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

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

Generating a New Image

Images can be generated using the following method:

```
ImaGen.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 Code

```
using UnityEngine;
using MADD;

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

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

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

    public void GenerateStableDiffusion()
    {
        if (_selectedModel == null)
        {
            Debug.LogWarning("You gotta select a model first!");
```

```
            return;
        }
        string prompt = _if.text;
        ImaGen.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;
    }
}
```

Usage and Rate Limit

Kindly offers a free tier that allows users to generate up to 100 images per month. For more extensive usage, you can choose the paid option, which provides 100 additional image generations for every £1.

Please note that the rate limits for the free and paid tiers are as follows:

Free Tier: 100 generations per month Paid Tier: Additional 100 generations for every £1 spent

That's All Folks!

The Kindly Unity Package brings the power of 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 on-the-fly.

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

Happy creating! 🎮🖼️