Ilumisoft

Graphics Control

Get Started

Welcome

Thank you for using Graphics Control!

On the next pages, you will find descriptions about how to use this package.

If you like the project, we would be grateful if you would take a minute and give us a rating in the Asset Store. This really helps us in order to create and improve our Unity assets.

If you encounter any problems or errors or if you have any questions, please get in touch with us.

Setup

Graphics Control has been designed to work out of the box, therefore you just need to import the package to get started:

1. Make sure you are using the latest release of **Unity 2021.3 LTS or higher**.
2. Create a new project or open the project you want to use.
3. Import **Graphics Control** from the **Asset Store** or the **Package Manager**.

That's it! In case you encounter any problems during import or getting shown any error messages, please contact us.

Get Started

In the following guide we will explore how the package works.

Import the package

If not already done, open the **Package Manager**, select **My Assets** and download and import the **Graphics Control** package.

Open and explore the Sample

Next open the Sample scene provided with the package (Plugins->Ilumisoft->Graphics Control->Sample->Scenes) to get an overview of the provided features and options. It contains a simple scene made of cubes, a graphic settings panel and a post process volume.

When you enter playmode, you will notice that the panel automatically gets filled with a set of settings that you can modify. Get to the **Bloom** setting and set it to "off". When you now click the **Apply** button, you should notice that the bloom has been disabled in the scene. Additionally the new value is stored and will be recovered when leaving and entering playmode again.

While still in playmode take a look at the Scene Hierarchy and unfold the **DontDestroyOnLoad** section. There you will find an instance of the **Graphic Settings Manager**. It controls which settings are available and controls their lifecycle. In the next step we will explore how you can modify the available settings.

Modify the settings

Exit playmode, go to Edit->Project Settings->Graphics Control. Here you can define the prefab that will be used to create the Graphic Settings Manager, control whether it should be created automatically when the game is started and also set the prefab used for the Graphic Settings Panel.

Double click the Graphic Settings Manager prefab to open it. Notice that each setting available has its own component, providing you with some default settings. If you want to remove a setting from the game you can simply remove the associated component here. Additionally if you want to add a setting to the game, click AddComponent->Graphics Control->Settings and select the one you need. The order of the components is the order that will be used by the Graphic Settings Panel.

Writing Custom Settings

While Graphics Control handles many common settings for you out of the box, you can easily write your own custom settings if you need to. In the following example we will write two custom settings.

Example 1: Writing the Soft Particles Setting

Graphics Control provides you with two types of settings: The very simple Graphic Toggle Setting and the more generic Multi Option Setting. Graphic Toggle Setting only accepts two states (on or off), while Multi Option Setting allows you to create more complex settings.

The first example will be a setting to control the Soft Particle Quality Option (Project Settings->Quality->Soft Particles), allowing you to enable or disable Soft Particles. Since Soft Particles can either be on or off, GraphicToggleSetting will be the perfect choice as a base class.

Create a new script and name it SoftParticlesSetting.cs with the following content:

```
using Ilumisoft.GraphicsControl;

using UnityEngine;

[DisallowMultipleComponent]

[AddComponentMenu("Graphics Control/Settings/Soft Particles Setting")]

public class SoftParticlesSetting : ToggleGraphicSetting

{

      public override string GetSettingName()

      {

      return "Soft Particles";

      }

}
```

Great, you have defined your first custom Graphic Setting! Next open the Graphic Settings Manager Prefab, click AddComponent->Graphics Control->Settings and select the Soft Particles Setting component to add it. Save the changes. When you now run the sample scene in Playmode again, you will notice that there is a Soft Particles option now.

Still there is a catch. The setting is available, but it will not be applied yet. That's because we have not defined what should happen when our SoftParticlesSetting is on or off.

To do so we need to write another simple script, a Graphic Settings Applier. They allow you to define how the settings you create should be applied. Most of the default settings coming with GraphicsControl are applied by the GlobalSettingsApplier component, but when creating your own custom settings, the cleanest way is to simply write an additional component instead of modifying our one.

Create a new script and name it SoftParticlesSettingApplier.cs with the following content:

```
using Ilumisoft.GraphicsControl;

using UnityEngine;

public class SoftParticlesSettingApplier : GraphicSettingsApplier

{

      public override void ApplySettings()

      {

            if
      (GraphicSettingsManager.Instance.TryGet<SoftParticlesSetting>(out
            var setting))

            {

                  // Get the selected option

                  bool selectedOption = setting.GetSelectedOption();

                  // Enable/disable soft particles depending on the
                  selected option

                  QualitySettings.softParticles = selectedOption;

            }

      }

}
```

Open the Graphic Settings Manager Prefab again, click AddComponent and add the SoftParticlesSettingApplier. That's it, when changing the setting at runtime, it will now be applied.

Example 2: Writing the Shadow Setting

For the second example, we will write a setting allowing the user to control the shadow settings. When opening the Quality tab in the Project Settings, there are three options that can be set for the shadow setting: Disable Shadows, Hard Shadows Only, Hard and Soft Shadows.

To reflect these three options for our setting, we cannot use the ToggleGraphicSetting from the first example. Instead we use the MultiOptionGraphicSetting.

MultiOptionGraphicSetting is a very flexible base class. It allows you to define the value type of the setting and can be used for any setting which does not has a binary state, e.g. resolution, AA level, etc.

To get strated create a new script, name it ShadowSetting.cs and add the following content:

```
using Ilumisoft.GraphicsControl;

using UnityEngine;


public class ShadowSetting : MultiOptionGraphicSetting<int>

{

    public override string GetSettingName()

    {

        return "Shadows";

    }


    public override void Initialize()

    {

        // The names of the available options

        var names = new string[]

        {

            "Off",

            "Hard Shadows Only",

            "Hard and Soft Shadows"

        };
        // Add each option
        for (int i = 0; i < names.Length; i++)
        {
            AddOption(names[i], i);
        }

        // Set the index according to the current option
        SetIndex((int)QualitySettings.shadows);
    }

    public override void LoadSetting()
    {
        // Restore the index from the Graphic Settings Storage
        int index = GraphicSettingsStorage.GetInt(key: GetSettingName(),
defaultValue: 0);

        // Set the index
        SetIndex(index);
    }

    public override void SaveSetting()
    {
        // Store the current index in the Graphic Settings Storage
        GraphicSettingsStorage.SetInt(key: GetSettingName(), value:
GetIndex());
    }
}
```

As you can see the script is a lot larger than the one from the previous example. MultiOptionGraphicSetting is much more powerful than ToggleGraphicSetting, but therefore we need to write a bit more code as a tradeoff. In the Initialize method, we create the options we have, where the first option is represented by a value of 0, the second by 1 and the last by 2.Then we set the initial index by the current shadow quality setting.

With LoadSetting and SaveSetting we can restore the stored value or save it respectively.

Again we also need to write an applier, therefore create a new script called ShadowSettingApplier.cs with the following content:

```
using Ilumisoft.GraphicsControl;

using UnityEngine;


public class ShadowSettingApplier : GraphicSettingsApplier

{

    public override void ApplySettings()

    {

        if (GraphicSettingsManager.Instance.TryGet<ShadowSetting>(out var
setting))

        {

            var shadowQuality = setting.GetSelectedOption();


            QualitySettings.shadows = (ShadowQuality)shadowQuality;

        }

    }

}
```

Here we get the selected option (which is an integer) and convert it to the ShadowQuality to apply it.

In the last step, open the Graphic Settings Manager Prefab again, click AddComponent and add the ShadowSetting and the ShadowSettingApplier to the prefab. Now when running the sample scene you can change the shadow settings!

Support

Do you have a question or need help? Don't hesitate to get in touch with us via email!

Email: support@ilumisoft.de