Vanguard Theatre – User Guide

info@uniblox.com
www.uniblox.com

Table of contents

About

Vanguard Theatre is an Asset and Scene management tool for Unity. It has the following features:

- Manage all your asset bundles from inside an Editor Window.
- Set individual settings for each Asset Bundle, including what platforms to build it to.
- Build all Asset Bundles at the same time, or build them individually.
- Automatically searches for dependencies before building with the option to automatically add any missing dependencies.
- Upload Asset Bundles to an FTP server from inside Unity.
- The option to automatically include each platform's Asset Bundles in its streaming assets.
- Automatic version control for Asset Bundles – Bring your players the latest assets, right away.
- Fully automatic and smart system for loading and unloading assets during runtime with support for your very own loading-screen.
- Load different Asset Bundle variants depending on the platform, player settings, etc.
- For the non-programmer: Not a single line of code required!
- For the ambitious programmer: Full source code included; fully documented!

Setup

Introduction

This section will detail all of the necessary steps to get Vanguard Theatre up and running inside your Unity project.

Adding Vanguard Theatre to your project

After purchasing Vanguard Datastream on the Unity Asset Store, press the “Download” button. Vanguard Datastream will be downloaded and the “Import Unity Package” window will open. Make sure that everything is selected, and press “Import”. Once importing is done, move the “Editor Default Resources” folder from the “Vanguard” folder to the root of your project window (the Assets folder).

Managers

Vanguard Theatre utilizes “manager” scripts, in the form of singletons; FTPManager, AssetBundleManager and StageManager. Thess managers require you to have a scene in your project which needs to remain loaded at all times and which contains these managers. To speed things up, we’ve included a “Management” scene with the plugin, located under *Vanguard > Theatre > Scenes*. Simply load this scene (and keep it loaded) to complete this part of the setup. If your project already includes such a scene, or if you want to create the scene yourself, you can also add the Prefabs from *Vanguard > Theatre > Prefabs* to this scene.

FTPManager

This manager makes all of this plugin’s FTP related features possible. To set it up, simply open up the inspector of the FTPManager in your “Management” scene which was set up in the previous section, and fill it in. Hover over an element in the inspector to see that element’s tooltip.

AssetBundleManager

Used to configure settings relating to Asset Bundles and to load and unload Asset Bundles at runtime. Set it up in the same way as you did with FTPManager.

StageManager

Used to load “Stages” – collections which contain scenes. This manager also has events which fire before and after a Stage is loaded. This allows you to, for example, change a Scene’s required Asset Bundles before loading depending on the platform and to do initialization after loading. Set it up in the same way as you did with FTPManager.

Demo

Introduction

Vanguard Theatre includes a demo. You can find it under *Vanguard > Theatre > Demo*. This demo showcases most of its features in an example scenario which also functions as a short tutorial. It should help you familiarize yourself with all that Vanguard Theatre has to offer. You'll be setting up 2 scenes. The objects in these scenes will be spawned at runtime from Prefabs, which will be loaded from Asset Bundles.

Asset Bundle Window

The following steps will guide you through the process of setting up the Asset Bundles which we'll be using in the demo.

- First, let's take a look at the Asset Bundles window. Go to *Window > Asset Bundles* to open it.
- There, click on the "+ New Asset Bundle" button. This will add a new Asset Bundle, named "new".
- Open up the Asset Bundle by clicking on the arrow next to it. You'll now see the variants of this Asset Bundle.
- Press the "Rename" button next to the variant and rename it to "sd".
- To view the assets included in the variant, select the variant by clicking on it and make sure that the "Assets" tab is selected on the right side of the window. You'll now most likely only see a button titled "+ Add asset", since we have yet to add any assets to this variant.
- Add the following assets to this variant:
    - *Vanguard > Theatre > Demo > Assets > Prefabs > SD > Ball*
    - *Vanguard > Theatre > Demo > Assets > Materials > SD > DemoMaterial*
    - *Vanguard > Theatre > Demo > Assets > Textures > SD > DemoTexture*
    - *Vanguard > Theatre > Demo > Assets > Shaders > DemoShader*
- Add a new Asset Bundle named "cube" and rename its variant to "sd". Add the following assets to the variant:
    - *Vanguard > Theatre > Demo > Assets > Prefabs > SD > Cube*
    - *Vanguard > Theatre > Demo > Assets > Materials > SD > DemoMaterial*
    - *Vanguard > Theatre > Demo > Assets > Textures > SD > DemoTexture*
    - *Vanguard > Theatre > Demo > Assets > Shaders > DemoShader*
- Add a new Asset Bundle named "cylinder" and rename its variant to "sd". Add the following assets to the variant:
    - *Vanguard > Theatre > Demo > Assets > Prefabs > SD > Cylinder*
    - *Vanguard > Theatre > Demo > Assets > Materials > SD > DemoMaterial*
    - *Vanguard > Theatre > Demo > Assets > Textures > SD > DemoTexture*
    - *Vanguard > Theatre > Demo > Assets > Shaders > DemoShader*
- To change the platforms for which a variant will be built, click on the "Platforms" tab. Here you will see a list with all the possible build targets. For the sake of this tutorial, select either "Windows" or "OSX", depending on the platform that you're currently using. You can apply the same build targets on every variant in every bundle by clicking the "Apply everywhere" button. Do this now.
- Everything is now accounted for, and the Asset Bundles can now be built. For this, we need the AssetBundleManager. Please make sure that this is set up according to the documentation.
- In the inspector of AssetBundleManager, make sure that "Use Streaming Assets" is turned on.
- Build the Asset Bundles by clicking "Build all asset bundles" down in the Asset Bundle window.
- After building has finished, take a look at your project window. You'll notice a new folder, named "Requirements". We'll need this in the next stage of the tutorial.

Scenes

Now that we have built all of the necessary Asset Bundles, we are ready to set up the scenes themselves. To save you some time, the scenes are almost fully set up already, with the exception of some Vanguard Theatre implementations. To change that, follow these steps:

- Let's take a look at the Requirements folder. The assets that you'll find in this folder were generated when you built the Asset Bundles, and are known as Requirements (who would have thought?). They can be thought of as references to the Asset Bundles themselves, although they don't exactly work like that. We won't be going into too much detail on Requirements right now, but if you're curious, they are detailed further in the documentation.
- Open *Vanguard > Theatre > Demo > Scenes > DemoScene1*.
- Click on BallSpawner in that scene's hierarchy to open it in the inspector.
- You'll see that it has a component named "Provider", and that it needs a reference to a Requirement. Set "ball.sd" from the Requirements folder as the reference.
- Do the same with CubeSpawner, but add "cube.sd" as the reference.
- Save the scene. After doing this, you'll notice that your project has gained another new folder. This one is named "SceneItems". It has one item in it with the same name as the scene you just saved. These SceneItems work as references to scenes, and also contain information on all the Requirements used in that scene. This is how Vanguard Theatre will know at runtime when to load and unload Asset Bundles.
- Open *Vanguard > Theatre > Demo > Scenes > DemoScene2*.
- In that scene, add "cube.sd" as a reference to Provider on CubeSpawner and add "cylinder.sd" as a reference to the Provider on CylinderSpawner.
- Save the scene. This scene's SceneItem is added to the SceneItems folder.

Stages

We're almost done with the scenes now. The next step is to get them to load properly at runtime, with the help of Stages. These are collections of SceneItems. They allow an active scene and multiple inactive scenes to be loaded by the StageManager at the same time. Please follow these steps to get this working:

- Add a new folder to your project called "Stages". There, navigate to *Create > Vanguard > Theatre > Stage*. Rename it to "DemoStage1".
- In the inspector, add the "DemoScene1" SceneItem as a reference to the active scene. We won't be using any inactive scenes.
- Create another stage, rename it to "DemoStage2", and add "DemoScene2" as the active scene.
- We'll need to do one more thing inside DemoScene1. Open it up, click on "Button_Load" inside the Canvas in the hierarchy, and add "DemoStage2" as a reference to the StageLoader component. Now we have a way of loading the second stage while running the demo.
- If you haven't set up StageManager yet, do so by following the "Setup" section of the documentation.
- Click on StageManager to open it in the Inspector. It asks for a loading scene and the first stage to load. For the stage, add "DemoStage1". For the loading scene, add *Vanguard > Theatre > Demo > SceneItems > DemoLoading*.

That's it! You're done. To test out the demo, just click play (make sure that all required scenes are added to the build first). The first stage will be loaded. When you click the "Load second stage" button, the second stage will be loaded. Under the hood, the assets required for the ball will be unloaded, those for the cube will be kept, and those for the cylinder will be loaded. Vanguard Theatre will always load, keep or unload assets exactly when needed and as efficiently as possible.

Requirements

Introduction

Requirements are used as references to Asset Bundles during runtime. They let Vanguard Theatre know which Asset Bundles are used in which scenes and are used to load assets from Asset Bundles.

Usage

To use a Requirement, add a “Provider” component to a GameObject and add the Requirement as a reference. Add this provider as a reference to a different component and call *LoadAsset()* to load a single asset, *LoadAssetAsync()* to load a single asset asynchronously, *LoadAssetWithSubAssets()* to load an asset with sub assets and *LoadAssetWithSubAssetsAsync()* to load an asset with sub assets asynchronously.

Requirement groups

Introduction

RequirementGroups allow you to use multiple Requirments with one Provider. Use this when you need to use multiple Asset Bundles with one GameObject.

Usage

To use a RequirementGroup, add a “MultiProvider” component to a GameObject and add the RequirmentGroup as a reference. Use the MultiProvider in the same way you would use a Provider.

SceneItems

Introduction

SceneItems can be used to reference a scene in the inspector. They also contain information on all the Requirements used in the scene.

Usage

SceneItems are automatically generated upon saving a scene. They should not be renamed, but can be moved to wherever you want.

Stages

Introduction

A stage can contain an active scene and multiple inactive scenes which will all be loaded at once. Only one Stage can be loaded at the same time.

Usage

To create a stage, navigate to *Create > Vanguard > Theatre > Stage* in the project window. They can be renamed and moved in any way you please. In the inspector, add the relevant scenes. To load the Stage, call *StageManager.Instance.SwitchToStage()*.

StageManager

Introduction

Used to load Stages. This manager also has events which fire before and after a Stage is loaded. This allows you to, for example, change a Scene's required Asset Bundles before loading depending on the platform and to do initialization after loading.

Usage

To use the StageManager, first set it up according to the "Setup" section of the documentation. Next, in the inspector, add references to both a SceneItem to use as the loading scene and the first Stage which should be loaded. If the first stage is not set, no Stage will be loaded initially.

AssetBundleManager

Introduction

The AssetBundleManager is used by Vanguard Theatre to load Asset Bundles during runtime.

Usage

To use the AssetBundleManager, first set it up according to the "Setup" section of the documentation. In the inspector, you can toggle the use of the streaming assets folder. If used, Asset Bundles relative to the platform which is set as the build target will be automatically placed in this folder. Everything in this folder will be contained in the build of the application. This means that the AssetBundles can be loaded at runtime without requiring a network connection. AssetBundles will always be loaded from here, unless there's a newer version of the same AssetBundle available on the FTP server.

FTP

Introduction

Vanguard Theatre allows you to upload AssetBundles to an FTP server and it allows users to download those AssetBundles during runtime.

Usage

For this, you are required to have FTPManager set up in your project according to the "Setup" section of the documentation. For more information on the FTPManager, please view the documentation of Vanguard Datastream.

If you want a variant to be uploaded to the FTP server, follow these steps:

- Open up the Asset Bundles Window.
- Select the variant which you want to upload.
- Go to the "Settings" tab.
- Toggle on "Upload to FTP server".

If you want all variants in every AssetBundle to uploaded, press "Apply everywhere" to apply this variant's settings to every other variant.

Outro

Thank you for choosing Vanguard Theatre! We hope that everything is clear to you, and that you'll have a great time using this product. If it has left a positive impression, we'd like to ask you to rate this product on the Unity Asset Store. This way, more people will be able to find it!

Any questions, feature requests, complaints or compliments? You can reach us at info@uniblox.com.