Voxon Unity Plugin User Guide

Unity is a popular 3D engine for developing applications. The *Voxon Unity Plugin* makes it possible for *Unity* applications to run as VX applications. A *Unity* VX App can take advantage of *Unity's* impressive 3D engine, assets store and its plentiful online tutorials.

To get started developing VX apps with *Unity*, you will need to download both *Unity* and the *Voxon Developers Kit*.

Note: The Voxon Unity Plugin is not located on the Unity Asset Store. It is an asset package included in the Voxon Developers Kit which is obtainable from our website (www.voxon.co), or download directly from https://github.com/Voxon-Photonics/Content-Developers-Kit

What build of Unity to use?

There are many versions of *Unity*. It is easiest to manage if you install *Unity Hub* which lets you install multiple build versions of *Unity* on a single computer.

It is recommended that you use *Unity* version **2020 LTS** (Long Term Support) with the *Voxon Unity Plugin.*

Note: When developing any software with *Unity*, do not change *Unity* builds mid project as switching *Unity* versions can result in compatibility problems, and potentially breaking your project!

What Voxon Runtime and files are needed?

The *Voxon Unity Plugin* uses a C# bridge to interact with the *voxiebox.dll*. For this interaction to work, a few files and system settings need to be in place:

- *C#-Bridge-Interface.dll* and *C#-Runtime.dll* files need to exist in the Voxon Runtime directory (by default located at *C:\Voxon\System\Runtime*).
- User Path Variable needs to have the Voxon Runtime added.
- The Voxon Runtime path needs to be added to the system registry.

When you download the *Voxon Developers Kit*, the files, path variable, and registry values are created but you can reset the path and registry values by looking at the files within the 'C:\Voxon\System\Setup'

Links to Unity Video Tutorials

As an alternative to reading this documentation, we have created a few video tutorials on using the V*oxon Unity Plugin.*

Video 1, Introduction:	https://www.youtube.com/watch?v=ztMUCE0STss
Video 2, Scene creation:	https://www.youtube.com/watch?v=BzZkC2DKo3k
Video 3, Advanced Unity:	https://www.youtube.com/watch?v=C7fbXV_RQ0E

How to add the Voxon Plugin to a Unity Project

As the *Voxon Unity Plugin* is released as a *Unity Asset Package* (not available on the *Unity Store*), you will need to add this asset to your *Unity Project*. For an existing project you can import the *Voxon Unity Plugin* by the top menu (Assets>Import Package>Custom Package).

1. Import the Voxon Asset Package

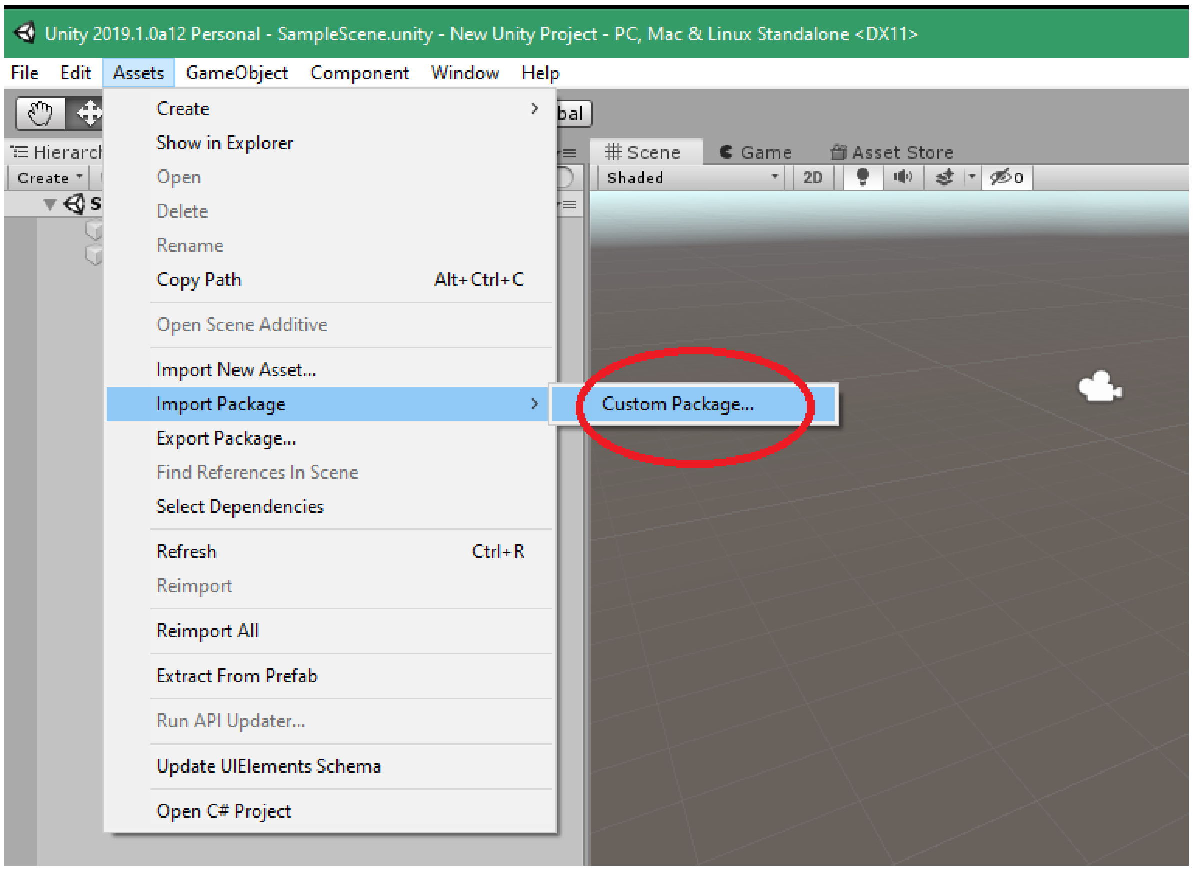
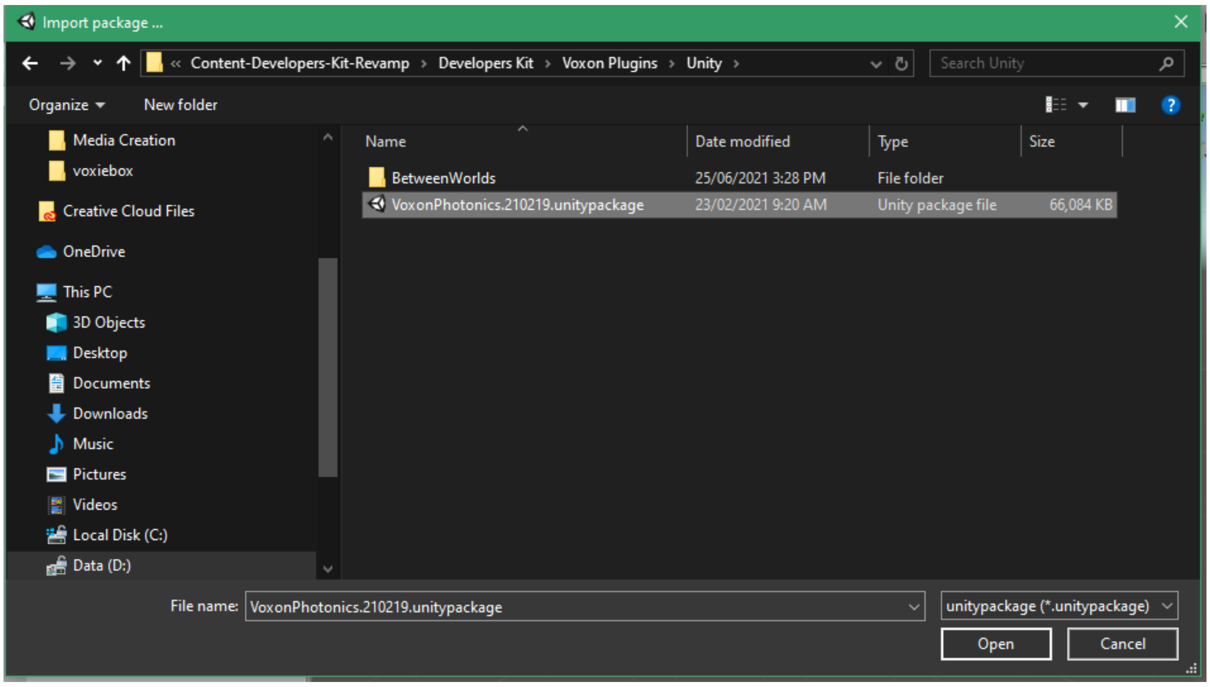


Figure 1 Use the Asset drop down menu to import the Voxon Unity Plugin into your project.

Click on the ‘Assets’ menu, then select ‘Import Package’ and then ‘Custom Package…’

2. Navigate to the Voxon Asset Package



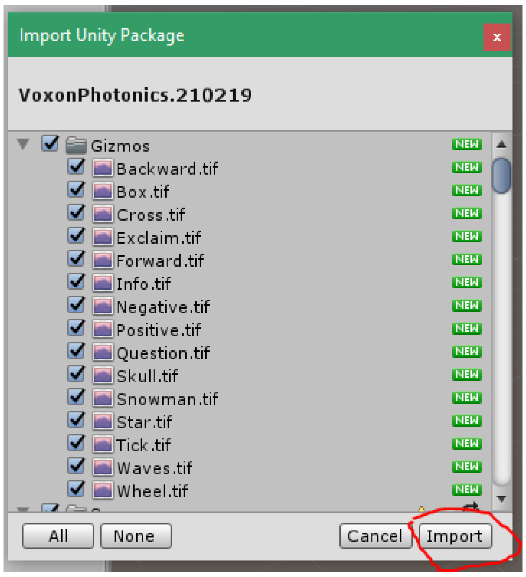
Name	Date modified	Type	Size
BetweenWorlds	25/06/2021 3:28 PM	File folder	
VoxonPhotonics.210219.unitypackage	23/02/2021 9:20 AM	Unity package file	66,084 KB

File name: VoxonPhotonics.210219.unitypackage
unitypackage (*.unitypackage)

Figure 2 Navigating to where the Voxon Unity Plugin is located (VoxonPhotonics210219.unitypackage was the latest asset package name of writing this document)

The *Voxon Unity Plugin* is located within the *Voxon Developers Kit* (‘…\Voxon\Developers Kit\Voxon Plugins\Unity’).

3. Import all the package files.


Import Unity Package

VoxonPhotonics.210219

- Gizmos NEW
  - Backward.tif NEW
  - Box.tif NEW
  - Cross.tif NEW
  - Exclaim.tif NEW
  - Forward.tif NEW
  - Info.tif NEW
  - Negative.tif NEW
  - Positive.tif NEW
  - Question.tif NEW
  - Skull.tif NEW
  - Snowman.tif NEW
  - Star.tif NEW
  - Tick.tif NEW
  - Waves.tif NEW
  - Wheel.tif NEW

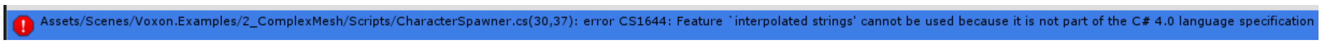
Figure 3 Make sure all the contents of the package are ticked and then click 'Import'

By default, make sure all the files are ticked to be imported and import them into your project. When you are more familiar with using the plugin, you can import only the files you need.

4. Ensuring the plugin is installed

The plugin is now added to your project.

If you are getting the following "cannot be used because it is not part of the C# 4.0 language specification" error(s) in the debug window, it means the project's Scripting Runtime is out of date.

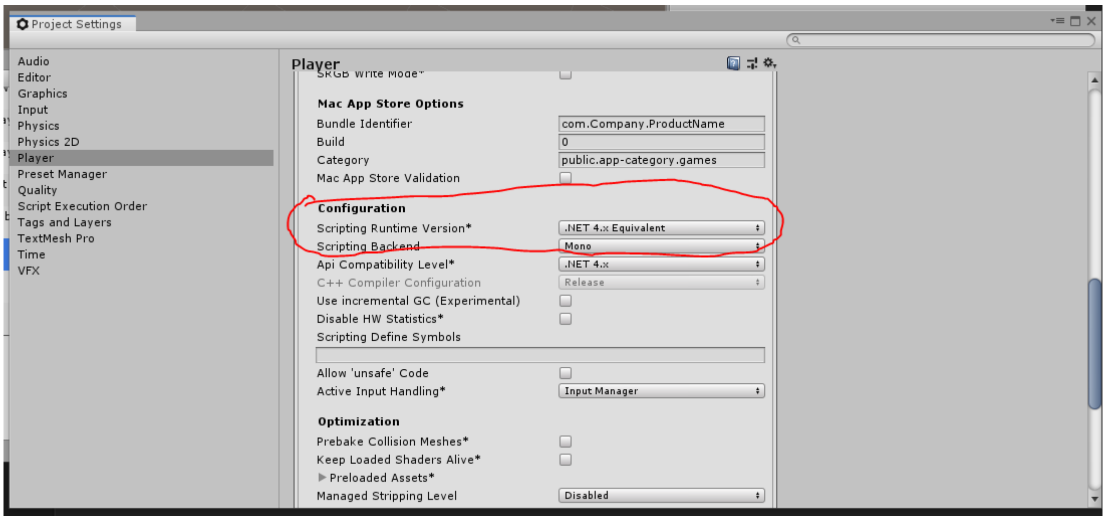


```
Assets/Scenes/Voxon.Examples/2_ComplexMesh/Scripts/CharacterSpawner.cs(30,37): error CS1644: Feature `interpolated strings' cannot be used because it is not part of the C# 4.0 language specification
```

Figure 4 Error message from Unity when the Project's Scripting Runtime is out of date.

To fix this within *Unity*, make sure that your project's 'Scripting Runtime Version' is at least '.NET 4.x Equivalent'. You can find this setting by following:

'Edit->Project Settings->Player->Other Settings->Configuration->Scripting Runtime Version'.



sRGB Write Mode*

Mac App Store Options

Bundle Identifier: com.Company.ProductName
Build: 0
Category: public.app-category.games
Mac App Store Validation

Configuration

Scripting Runtime Version*: .NET 4.x Equivalent
Scripting Backend: Mono
Api Compatibility Level*: .NET 4.x
C++ Compiler Configuration: Release
Use incremental GC (Experimental)
Disable HW Statistics*
Scripting Define Symbols
Allow 'unsafe' Code
Active Input Handling*: Input Manager

Optimization

Prebake Collision Meshes*
Keep Loaded Shaders Alive*
Preloaded Assets*
Managed Stripping Level: Disabled

Figure 5 The Project's Player settings within Unity

If the plugin has been initialised successfully, you will notice some new directories in your project and a 'Voxon' menu option in the top menu bar.

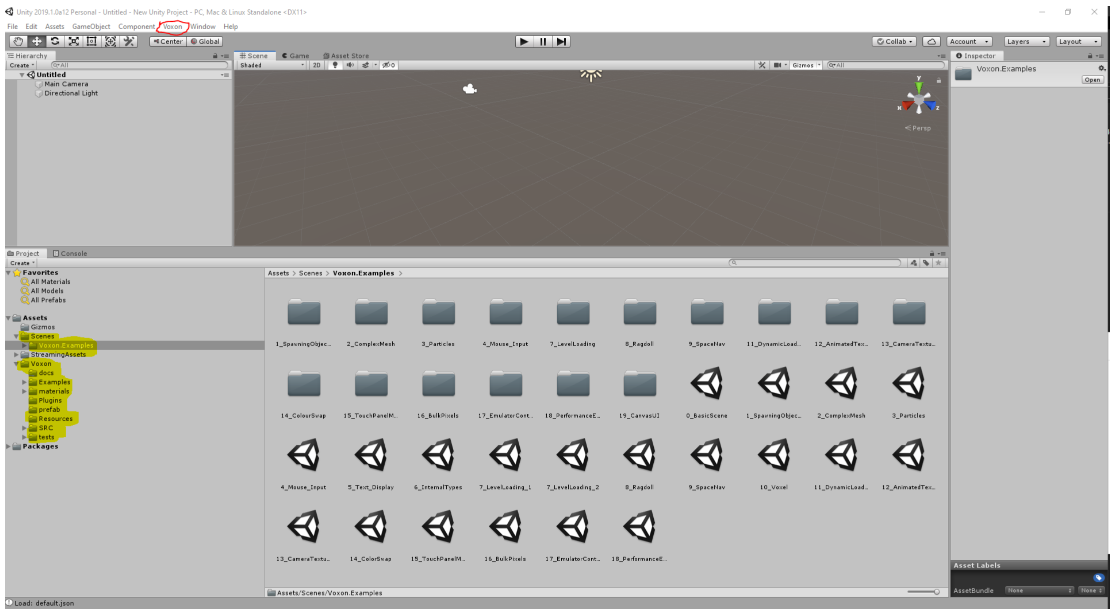


Figure 6 The new directories and files added to your project from the plugin.

5. Add the _camera prefab to your current scene

To render anything on the VX1, you will need to set a Voxon camera. Included in the plugin is a '_camera' prefab. This prefab captures anything within it and sends that data to a VX1.

Look under the Voxon prefab folder within your assets folder and drag the '_camera' prefab into your scene.

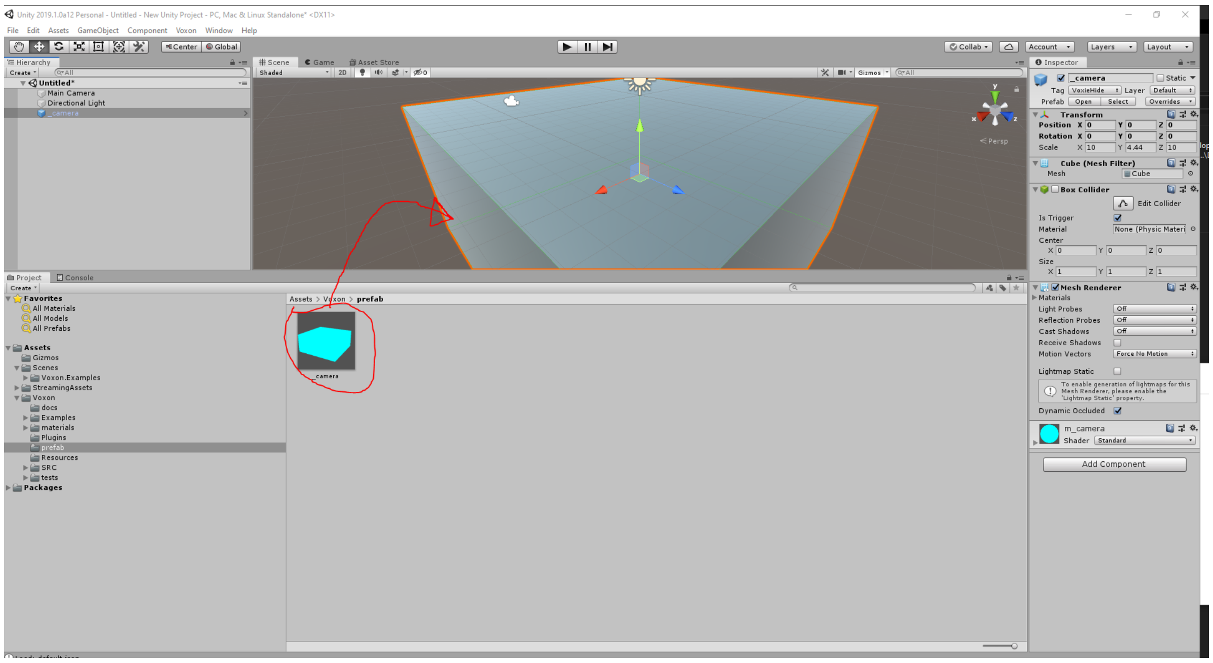
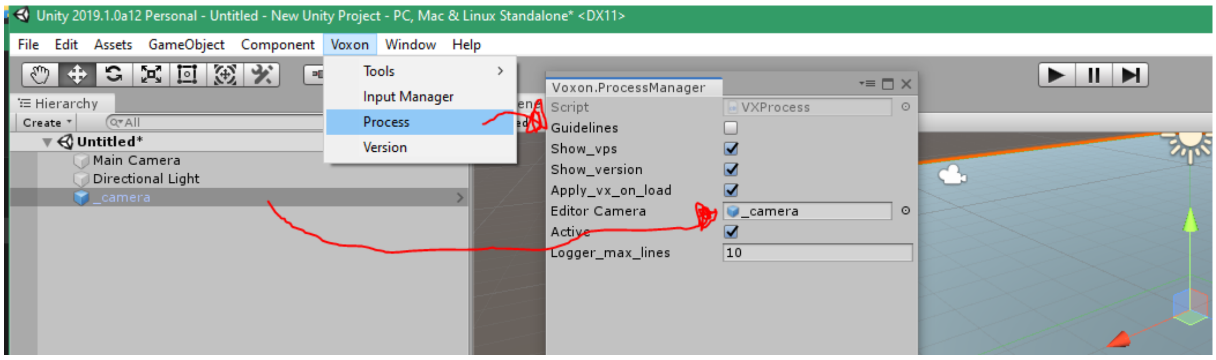


Figure 7 Drag the _camera prefab into your scene to add a Voxon camera.

6. Setup the Process Manager

Set the '_camera' object to be the 'Editor Camera' in the 'Process Manager'. Click on the Voxon menu tab and select 'Process'. Drag the '_camera' object to the 'Editor Camera' field.



Voxon.ProcessManager

Script	VXProcess
Guidelines	☐
Show_vps	✓
Show_version	✓
Apply_vx_on_load	✓
Editor Camera	_camera
Active	✓
Logger_max_lines	10

Figure 8 To activate a camera open up the Process Manager and Drag the _camera GameObject to the Editor Camera field.

7. Run your project!

That should be all that is needed. Save your project, then press the 'Run' button.

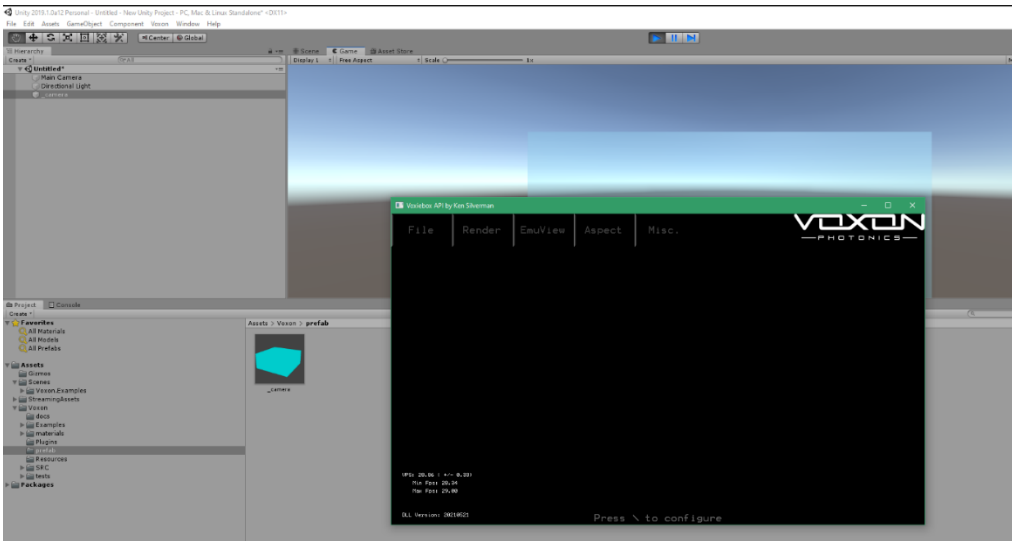


Figure 9 Test your program.

Your project should run on the Voxon simulator window.

Note: Always save your project before testing. It can crash often!

Always Press the 'Esc' key to exit the simulator once you have finished testing. Do not press the 'Play' triangle button on *Unity* to end the test. Do not use the VoxieMenu 'File > Shutdown'. Always use the 'Esc' key otherwise the system can become unstable.

This is a problem we are aware of and are working on a solution, but it demands a total reworking of the Voxon Unity Plugin. Not an easy fix!

Tip: when using the *Voxon Unity Plugin* it is recommended you turn off 'exclusive mouse' on the Voxon Simulator. This setting is under the 'Misc.' Tab.

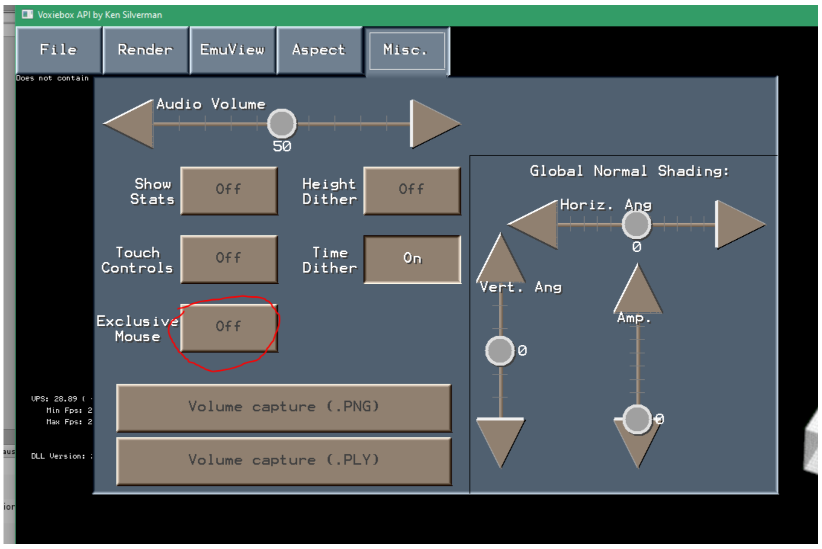


Figure 10 Turn off Exclusive Mouse to reveal your cursor when developing with the Voxon Simulator

Press (File > Save Settings) to keep this setting for future testing. This will make it easier to use the mouse as you develop your application.

Voxon Example Scenes

A variety of example scenes can be found under ‘Scenes > Voxon.Examples’ in the project files. New examples are added often, and these are a great place to learn how to use the plugin.

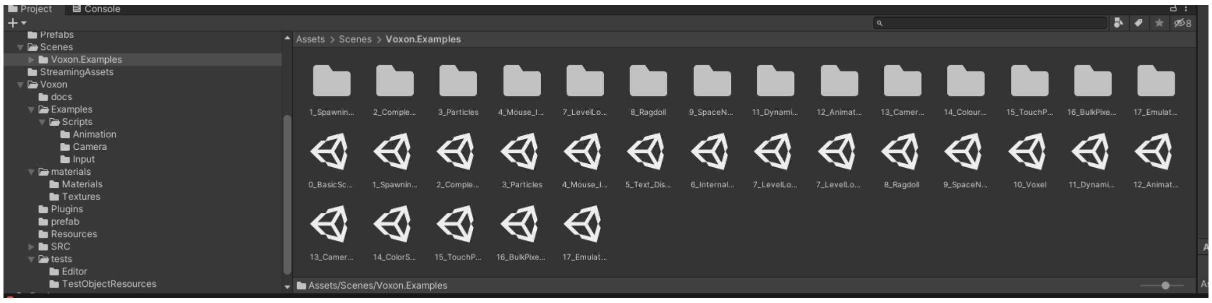


Figure 11 The ‘Voxon Example’ folders provide many examples of using the Voxon Unity Plugin

The 'Voxon' File Menu

The Voxon File Menu holds various items to do with the *Voxon Unity Plugin.*

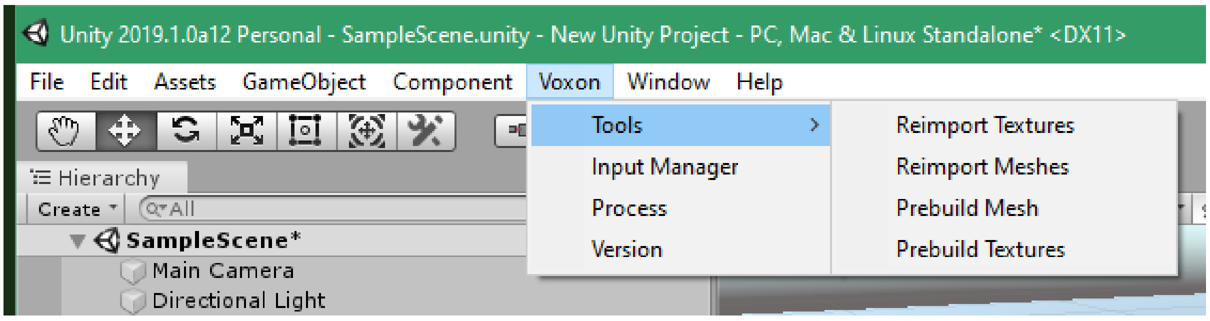


Figure 12 Introducing the 'Voxon' file menu.

Here is a general breakdown of all the options:

Tools -> Reimport Textures: Once the plugin is initialised all textures added to the project are also added to an internal list managed by the *Voxon Unity Plugin*. There are some instances where the textures do not get added. This can happen if there were textures added to the project before the plugin was imported or sometimes an error occurs. Pressing the 'Reimport Textures' button will reimport all textures in the entire Unity Project. **Run this tool if textures are missing from the volumetric render or you receive texture related errors in the Unity Debug menu.**

Tools -> Reimport Meshes: Like the 'Reimport Texture' tool. All meshes added to the project after the plugin is set up will be amended to an internal list. **Run the Reimport Meshes tool if the internal list is missing meshes or has become corrupt.** Meshes added before the plugin was initialised will not be added until this tool has been run.

Tool -> Rebuild Mesh: Reduces scene loading times by creating a cache of all the mesh data in the scene. If no mesh cache is created Unity must prepare all the mesh data for the *Voxon Runtime* every time the scene is loaded or tested. By selecting 'Rebuild Mesh' a cache of all the mesh is created.

Tool -> Prebuild Textures: Reduces scene loading times by creating a cache of all the texture data in the scene. If no texture cache is created *Unity* must prepare all the texture data for the *Voxon Runtime* every time the scene is loaded or tested. By selecting 'Prebuild Textures' a cache of all the textures is created.

Input Manager:

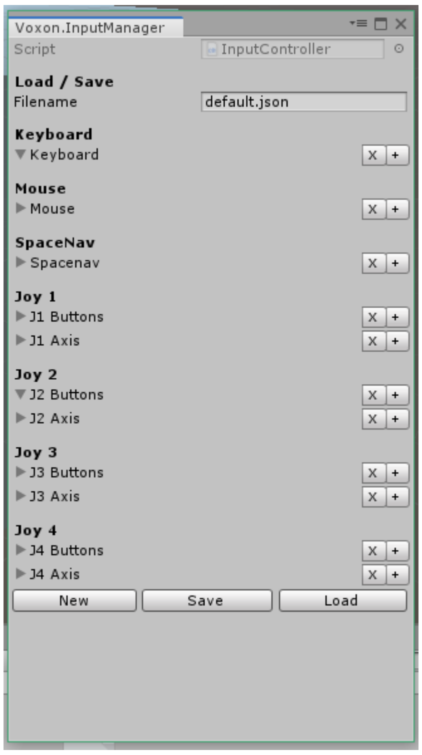

Voxon.InputManager
Script InputController
Load / Save
Filename default.json
Keyboard
▼Keyboard
Mouse
▶Mouse
SpaceNav
▶Spacenav
Joy 1
▶J1 Buttons
▶J1 Axis
Joy 2
▼J2 Buttons
▶J2 Axis
Joy 3
▶J3 Buttons
▶J3 Axis
Joy 4
▶J4 Buttons
▶J4 Axis
New Save Load

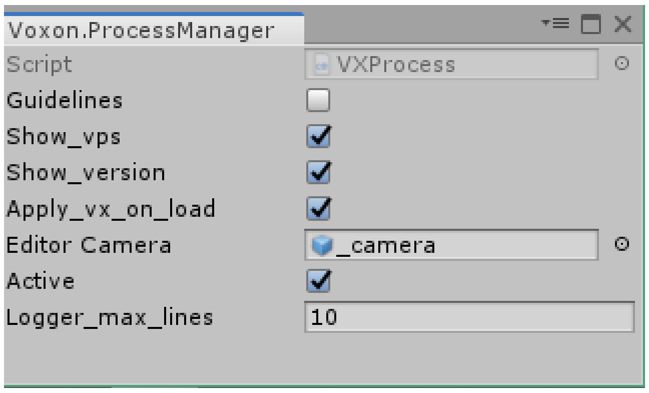
Figure 13 The Voxon Input Manager for adding your own inputs

Standard *Unity* inputs don't work for a VX app, so the Input Manager was created as a way of passing through input states from *Unity* to the VX app.

The Input Manager is used to set various inputs (keyboard, mouse, Space Navigator and USB game controllers) to be used while the VX app is running. All input buttons need to be defined in the Input Manager.

To reference these keys in scripts use the 'Voxon.Input.GetKey()' function or relevant functions depending on the input type. See examples scenes.

Process Manager:



Voxon.ProcessManager

Script	VXProcess
Guidelines	☐
Show_vps	✓
Show_version	✓
Apply_vx_on_load	✓
Editor Camera	_camera
Active	✓
Logger_max_lines	10

Figure 14 The Voxon Unity Plugin Process Manager

There are many settings in the Process Manager to do with rendering to the display.

Script: The script that is used to manage the rendering process. 'VXProcess' is the default but it is possible to write your own custom script.

Show_vps: displays the VPS information onto the secondary (touch) screen.

Show_version: displays the Unity Plugin and *voxiebox.dll* version onto the secondary (touch) screen.

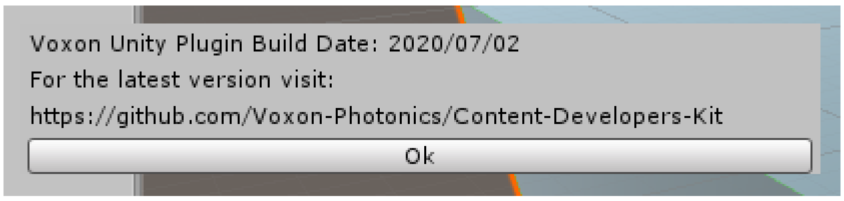
Apply_vx_on_load: if enabled all GameObjects that do not have a 'VxGameObject' script attached will have one attached when the app is started. Any GameObject without a 'VXGameObject' (or 'VXComponent') script attached will not render on the volumetric display.

Editor Camera: Is to be referenced to the GameObject that is the 'camera' for the VX app. Use the '_camera' prefab as a starting point. Anything within the bounds of the camera GameObject will be displayed.

Active: if ticked, when you test your app it will launch the Voxon Simulator. **Disable this setting if you want to do debugging without loading the simulator** (good for quick changes and safer with a lower risk of crashing).

Logger_max_lines: the number of lines the internal Voxon logger can output. The text is displayed during a VX app runtime on the secondary (touch) screen. The default is set to 10.

Version: shows the current version of the Voxon Unity Plugin.



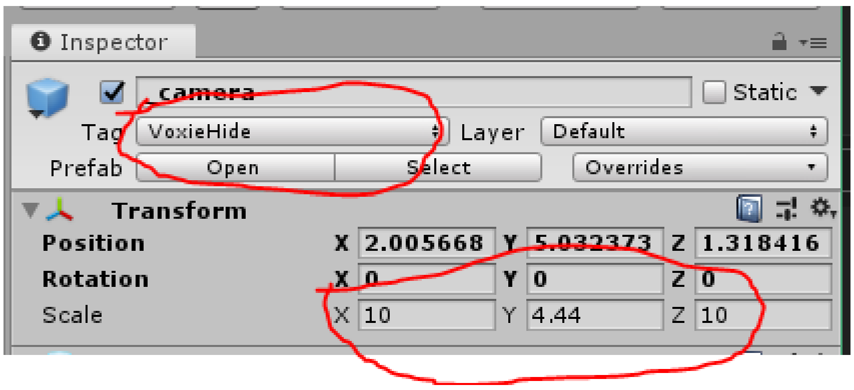
Voxon Unity Plugin Build Date: 2020/07/02
For the latest version visit:
https://github.com/Voxon-Photonics/Content-Developers-Kit

Figure 15 The Version information for the Voxon Unity Plugin

The _camera prefab

The camera prefab is a special GameObject that represents the Voxon display volume. The dimensions (10 x 4.44 x 10) adhere to the default aspect ratio of the volumetric display (1 x.4 x 1). It also uses the special *Unity Tag* 'VoxieHide' which prevents the object from being rendered on the volumetric display. The '_camera' prefab is located under the Voxon > prefab folder.

Technically any GameObject can act as a camera, just remember that the dimensions of the camera are derived from the scale of the GameObject. Make sure that the GameObject has the 'VoxieHide' Tag.



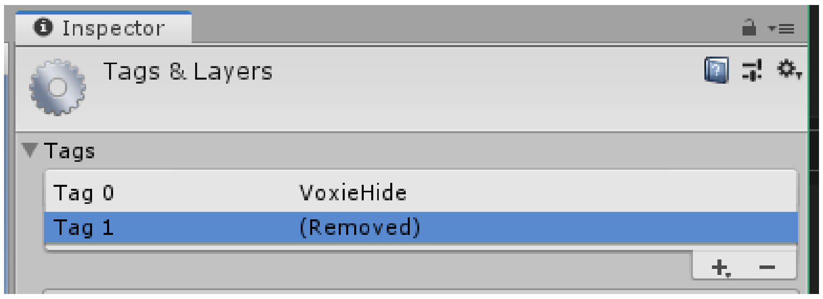
Position X 2.005668 Y 5.032373 Z 1.318416
Rotation X 0 Y 0 Z 0
Scale X 10 Y 4.44 Z 10

Figure 16 The 'VoxieHide' tag and Scale of a GameObject is what makes it work as a Voxon camera object.

Special tags are used to interact with the Voxon Unity plugin. A single tag can be applied to any GameObject. The option is located just beneath the name of the GameObject.

VoxieHide – The 'VoxeHide' tag is used to prevent GameObjects from being rendered on the volumetric display. It's the secret tag that makes a Voxon Camera work, but it can be used for other purposes.

If your *Unity Project* does not have the 'VoxieHide' tag you will need to add it to as a custom tag.


Inspector
Tags & Layers
Tags

Tag 0	VoxieHide
Tag 1	(Removed)

Figure 17 You may need to add VoxieHide to your project's custom tags

Special VX Scripts

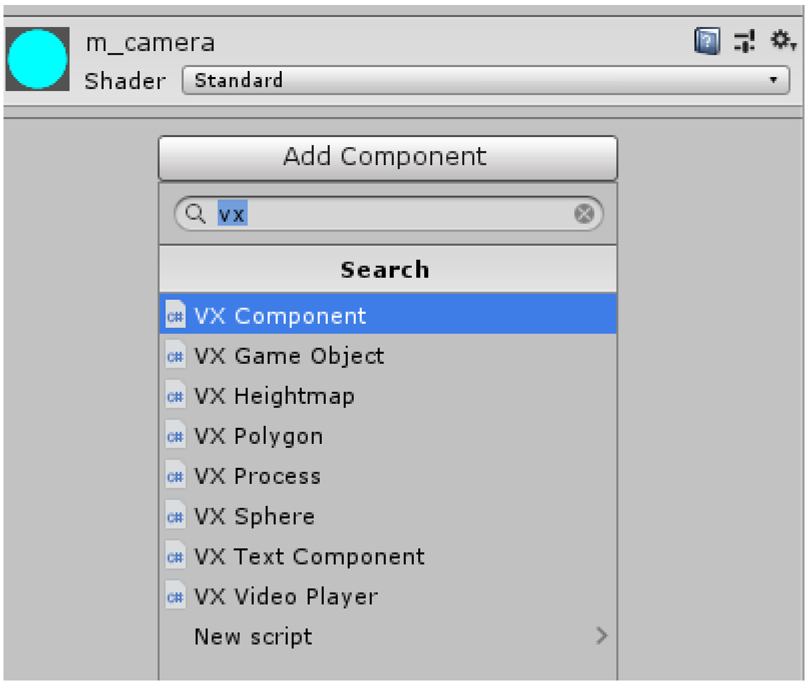
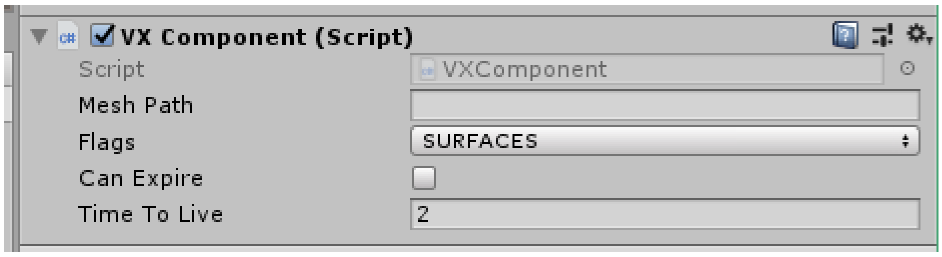

m_camera
Shader: Standard
Add Component
vx
Search
VX Component
VX Game Object
VX Heightmap
VX Polygon
VX Process
VX Sphere
VX Text Component
VX Video Player
New script

Figure 18 A list of the special 'VX scripts' to be used with Unity

Included with the *Voxon Unity Plugin* there are a bunch of scripts starting with 'VX' these 'VX scripts' are used to help the *Voxon Unity Plugin* and developers work.

Summary of the included scripts

VX Component: (or 'VX Game Object') needs to be attached to every GameObject that you want to render on the volumetric display. It also has a few different rendering methods and an auto expire mode.



Script	VXComponent
Mesh Path	
Flags	SURFACES
Can Expire	☐
Time To Live	2

Figure 19 The VX Component Script element

Mesh Path:	Automatically generated field when the object has mesh.
Flags:	Instructs how the GameObject should be rendered:

DOTS:	Draws only a voxel at every vertex.
LINE:	Draws the wireframe of the vertices.
SURFACES:	Draws the mesh's surfaces, default and recommended for most instances.
SOLID:	Fills the surfaces with solids (warning this method draws many voxels and requires a lot of CPU power)

Can Expire:	If enabled, hides the GameObject after a desired amount of time based on the 'Time To Live' field.
Time To Live:	The time in seconds before the GameObject is not rendered on the volumetric display.

VX Game Object: Any GameObject with the *VX Game Object* script added to it will render onto the volumetric display. **A VX GameObject script will also attach a VX Component script to any sub GameObjects associated with the parent GameObject.** This is useful when working with hierarchies.

VX Heightmap, **VX Polygon**, **VX Sphere**, **VX Heightmap:** Scripts which have not been fully developed and are reserved..

VX Text Component: Can be used to render text onto the volumetric display using the font used from the *voxiebox.dll*. This text can be rotated and scaled at any angle and relies on three vectors.

Pr – The right vector. The X value of this determines the width of the text.

Pd – The down vector. The Y value of this vector determines the height of the text.

PP – The left top position. This value determines the position of the text.

Color – The HEX value of the RGB colour to be displayed. This value is intended to be a hex number so it is a 16 base number presented as a 10-base number the default is '16777215' which is 0xFFFFFF in hex so it would be white. Red would be 16,711,680 (0xFF0000) etc…

Text – The actual text to display.

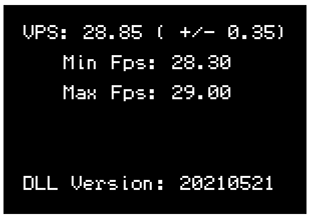
See the **5_Text_Display** example scene to see this script working in a demo.

VX Process: this is a special script which the *Voxon Process Manager* uses to execute its renderer. It is a singleton (can only be instantiated once) and not to be attached to various GameObjects. **Must only be used for the Process Manager.**

Note: If you have *Apply_vx_on_load* enabled on the *Voxon Process Manager* any GameObjects in your scene that do not already have a *VX Game Object* or *VX Component* script attached will automatically get a script attached on launch.

Performance Monitoring

The VX1 operates at 15 volumes per second. ‘volumes per second’ is the volumetric / 3D equivalent of ‘frames per second’ used in traditional computing. 15 volumes per second is the 2D screen equivalent of 15 fps. It is how many times the CPU is drawing the screen / volume per second.



```
VPS: 28.85 ( +/- 0.35)
    Min Fps: 28.30
    Max Fps: 29.00


DLL Version: 20210521
```

Figure 20 The VPS counter displayed by Show_vps

It is important that your VX app runs at least 15 volumes per second (above 30+ VPS is recommended).

The VPS amounts to how many volumes the VX1 can render and it is dependent on your computer's CPU load. Ultimately VPS is related to the amount of work your VX application is asking the CPU to do.

While using the simulator you can check the current VPS at any time. If you have ‘Show_vps’ checked in the *Voxon Process Manager*, it will display on the secondary screen. You can also see the VPS by turning on ‘show stats’ under the “Misc” tab.

Keeping your app performant is very important and while you are developing here are some tips to keep your VPS coundl high:

- In general, the more voxels (3d pixels) you are rendering the more the CPU is working.
- The resolution on the VX1 is quite low compared to a traditional 2D screen. The volumetric display is about the equivalent 1000 x 1000 x 200 pixels. So often texture and the number of vertices in a model are too high.

- Reduce a mesh's vertices amount - ervery mesh that is being rendered onto the volumetric display is made up of vertices. Your models may not have to be as high poly count..
- Reduce texture size. Every texture being rendered needs to be interpreted by the voxel frame buffer. A smaller texture size speeds up this process.
- Test often and optimise your project as you work. Always be aware of your VPS count.
- Sometimes it is not the rendering of the volumetric graphics slowing an app down. Check to see if there are any other processes happening within your app that are causing performance issues.
- Try to display what is needed. Less is more with the VX1.

Building your project to use on a VX1

Building your project to run as a VX app on real VX hardware is as straightforward as building any Unity application.

1. Check the build settings.

Click on the File menu and select 'Build Settings…'

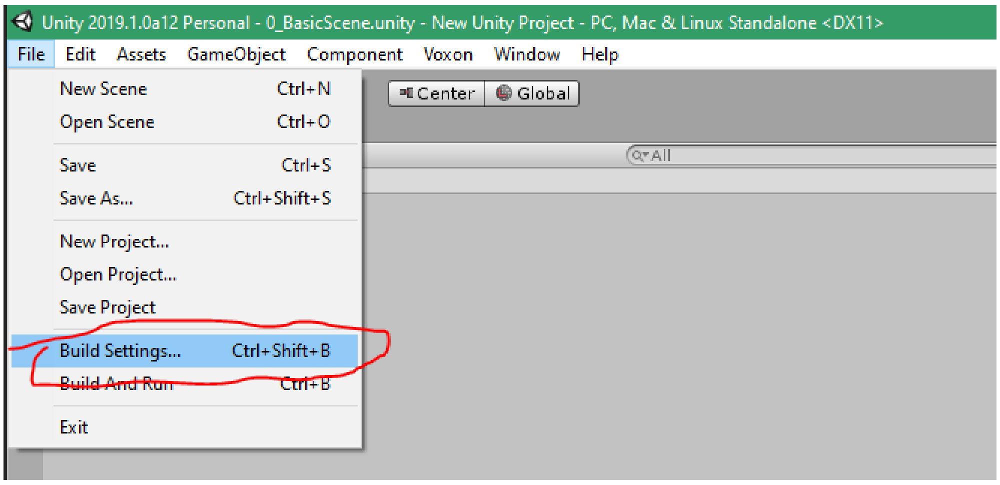
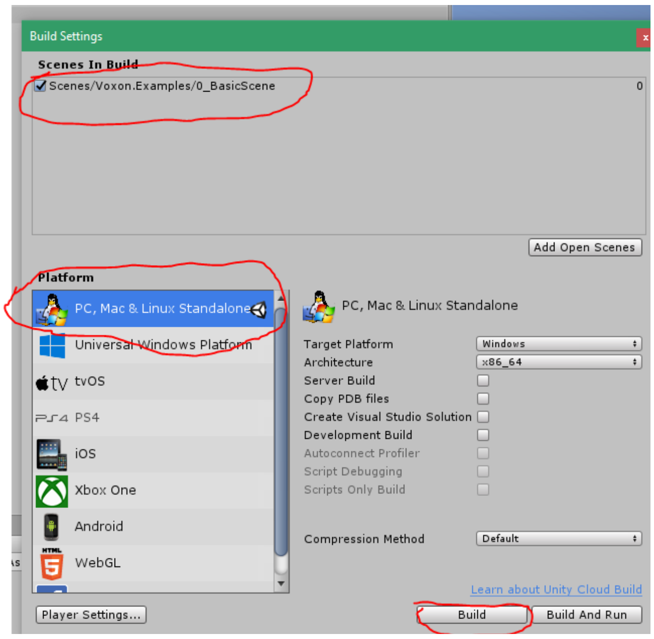


Figure 21 To build your app into an executable choose 'Build Settings..'.

2. Add the scene (or scenes) to be built into an app.


Build Settings

Scenes In Build

☑ Scenes/Voxon.Examples/0_BasicScene 0

Add Open Scenes

Platform

PC, Mac & Linux Standalone
Universal Windows Platform
tvOS
PS4
iOS
Xbox One
Android
WebGL

PC, Mac & Linux Standalone

Target Platform: Windows
Architecture: x86_64
Server Build ☐
Copy PDB files ☐
Create Visual Studio Solution ☐
Development Build ☐
Autoconnect Profiler ☐
Script Debugging ☐
Scripts Only Build ☐

Compression Method: Default

Figure 22 Building your app check that the Scene(s) you want to build are ticked, the Platform is for Windows and its Architecture is 64 bit.

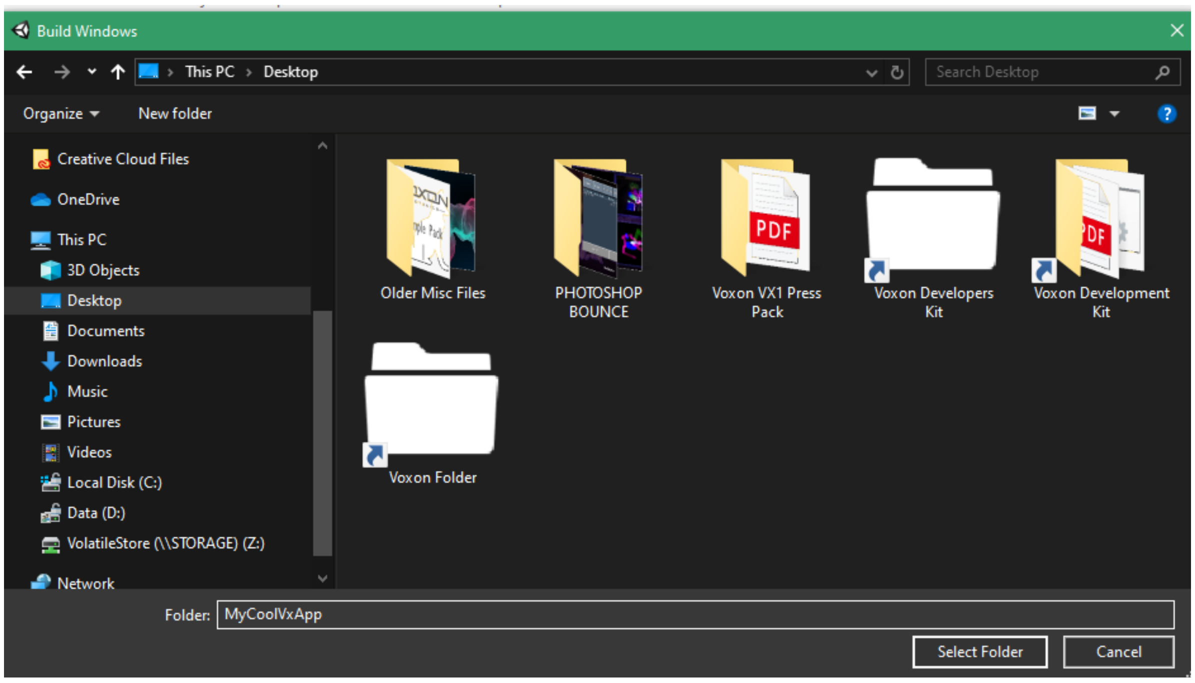
Make sure the scenes you wish to build are included in the 'Scenes In Build'. You can build multiple scenes into a single app. The scene at the top of the list is the scene which your app will start in.

Make sure the platform is set to 'PC, Mac & Linux Standalone'. Target platform is 'Windows' and Architecture is 'x86_64'.

Once you have these settings set click 'Build' to build your VX app!

Note: Make sure you have your Editor Camera active and set otherwise your VX app won't know what to display.

3. Choose a folder for the location of your build

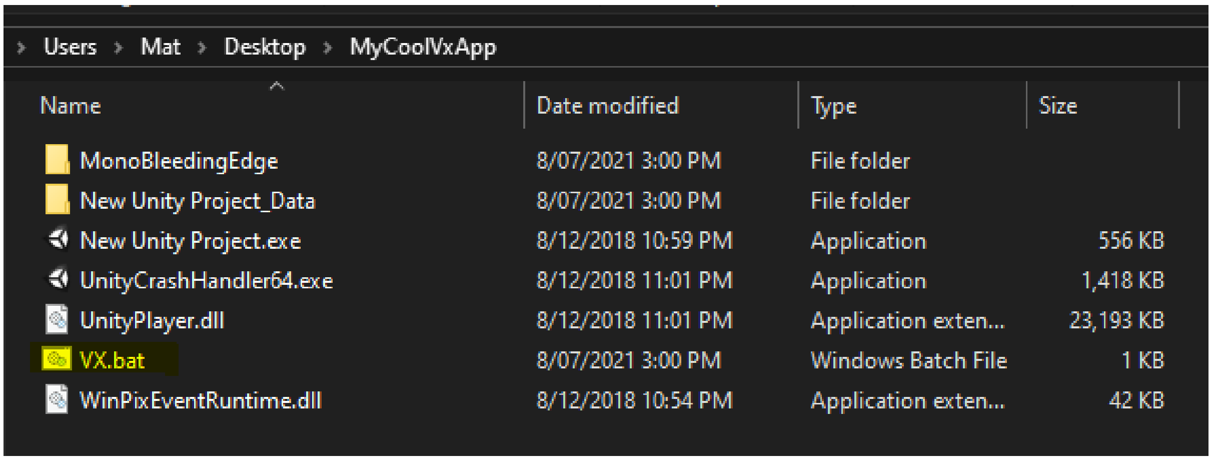


Folder: MyCoolVxApp

Figure 23 Choosing a destination to build your VX Unity app.

Choose a folder to build your app. You can call it whatever you like.

4. Test your app


Users > Mat > Desktop > MyCoolVxApp

Name	Date modified	Type	Size
MonoBleedingEdge	8/07/2021 3:00 PM	File folder	
New Unity Project_Data	8/07/2021 3:00 PM	File folder	
New Unity Project.exe	8/12/2018 10:59 PM	Application	556 KB
UnityCrashHandler64.exe	8/12/2018 11:01 PM	Application	1,418 KB
UnityPlayer.dll	8/12/2018 11:01 PM	Application exten...	23,193 KB
VX.bat	8/07/2021 3:00 PM	Windows Batch File	1 KB
WinPixEventRuntime.dll	8/12/2018 10:54 PM	Application exten...	42 KB

Figure 24 If your app has been built correctly you will have a VX.bat file to launch your app.

Navigate to the folder you built your app in. If the build was successful, you should see a bunch of files.

In this example *New Unity Project.exe* is the program's main executable (which will be named after whatever you called it). You can run this and it will work just fine, but for an optimised experience you will want to run the *VX.bat* file.

The *VX.bat* runs the VX application in 'batch mode' (runs the Unity .exe with a -batchmode parameter), backmode runs the application without creating a Unity window to render the graphics. (it's OK because the *Voxon Unity Plugin* knows what to do and works directly with the *Voxon Runtime*).

Note: Running a VX Unity app using the standard executable will still work but you will notice that there is a Voxon window created and a standard Unity window. This takes more CPU power as the computer must manage 2 instances of your program running. This is not ideal or practical.

Double click on the *VX.bat* and see what happens…

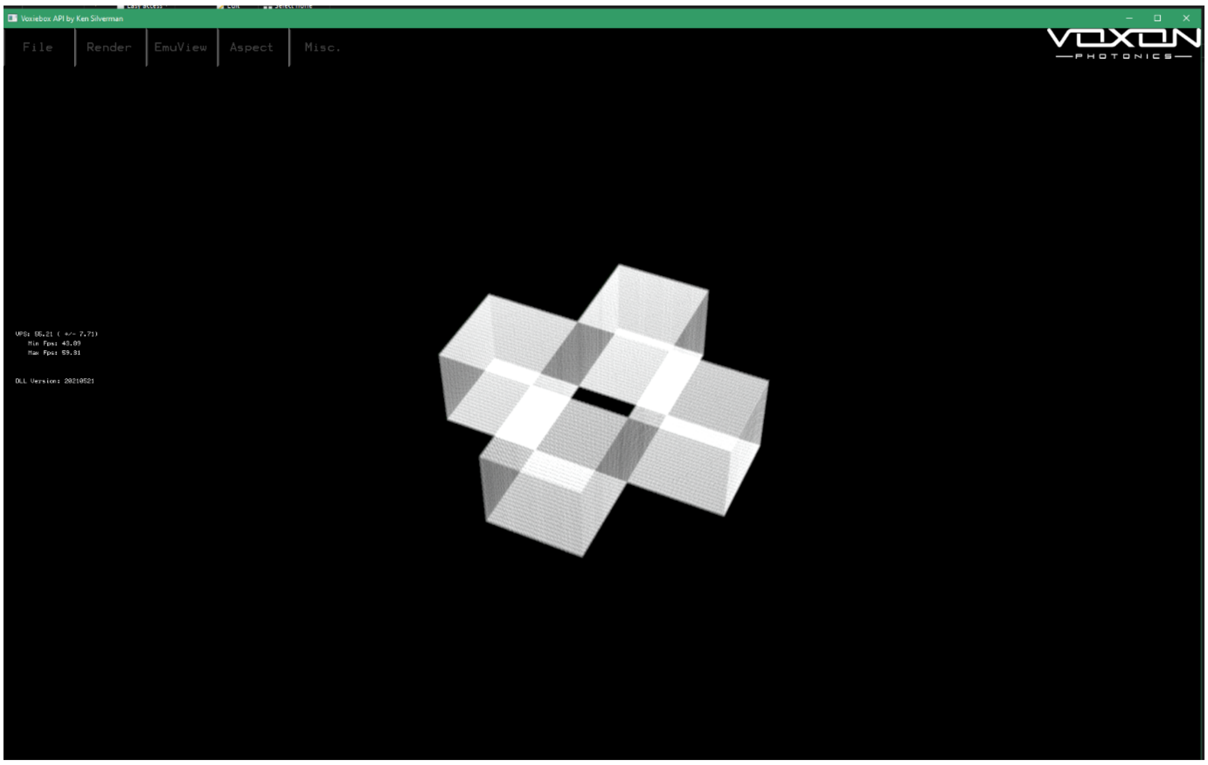
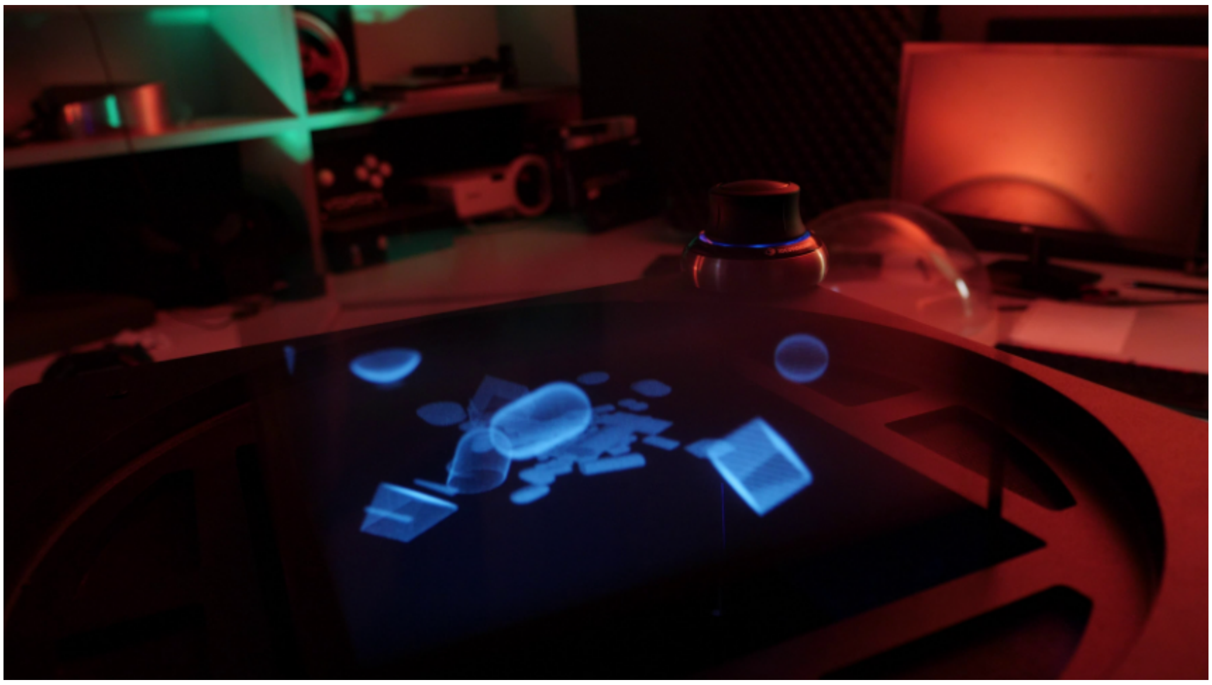


Figure 25 A VX Unity app being run on the Voxon Simulator

If your build settings were correct your app should run. Your app will work on any other system that has the Voxon Runtime installed.

If your VX app doesn't load, check that you actually added the scene to the 'Scenes In Build' bin in the build settings option and that your scenes have a '_camera' object attached.

Between Worlds Unity Example



Within the Developers Kit is a binary and package example of a VX Unity App called ‘Between Worlds’ you can use as a demonstration. It is located at ‘...\Voxon\Developers Kit\Voxon Plugins\Unity\BetweenWorlds Example’