Technie Virtual Console 2.0

Hello! Welcome to the Technie Virtual Console. Version 2.0 has had a bit of an overhaul, so even if you've already used it then please watch the introduction video and read this readme for more information on setup and changes.

Setup

To add the virtual console to your project simply add the `Virtual Console` prefab (from `Assets/Technie/VirtualConsole/Prefabs/Virtual Console.prefab`) to your scene.

It is assumed that you already have your project set up for VR and you have enabled VR support in the Unity Player Settings. The Virtual Console will work on any platform with Unity XR and hand controllers - that includes HTC Vive, Oculus Rift, Oculus Quest, PlayStation VR, Valve Index and WindowsMR.

Camera Detection

By default the console is set up to detect your vr camera automatically. However if you have multiple VR cameras you may want to specify which vr camera to use manually. Do this via the 'Camera Detection' settings in the inspector on the root of the prefab (the Virtual Console component).

- 'Automatic' uses the first active and vr-enabled camera in the scene. This will be fine for most projects.
- 'Use Explicit Camera Reference' allows you to drag in a camera to use (useful if the camera is in the same scene)
- 'Use Explicit Camera Name' allows you to set the name of the camera to use (useful if the camera is in a different scene or a prefab that's loaded later)

Example Scene

There's an example scene in `Assets/Technie/VirtualConsole/Scenes/Example1.scene`. This contains the Virtual Console prefab, the SteamVR camera prefab, a minimal environment, and some trigger boxes. If you are not using SteamVR then delete the [CameraRig] prefab and replace it with the Oculus Rift camera rig from the Rift SDK instead.

Putting a hand inside the trigger boxes will trigger different kinds of log/stat information to be created to demonstrate the Virtual Console.

Functionality

The 'Virtual Console' prefab can be included normally within your scene, and enabled/disabled at runtime by enabling/disabling the root game object. All customisation options are exposed on the `VirtualConsole` component on the root of the prefab.

By default the console will delete itself if running in a release build (ie. 'Development Build' in Build Settings dialog is unticked). This functionality can be turned off with the `Only in debug builds` option on the VirtualConsole component.

If you would rather not have any additional loading overhead for release builds then you can spawn the Virtual Console prefab dynamically as your game loads.

General Controls

A virtual console panel will appear attached to each hand, with stats and information on it. Each hand will also have a 'stylus' attached to it (a small blue sphere). Use this to poke the buttons on the panels to press them. Version 2.0 does not use any button input so that it does not interfere with your game's normal input handling.

Every panel has a few common controls:

- The current panel name is shown at the top. Use the left and right arrows next to it to change the current panel to a different one.
- The 'minimise' button ('_') next to this can collapse the panel down to a small rectangle. This is useful to hide it if you want to capture a screenshot or if it's getting in the way.
- The 'maximise' button ('+') at the top right will toggle the panel between small and large sizes. This is especially useful if you have a lot of console text that you need to see and the small panel can't show it all.
- When minimised, the panel will be replaced with a 'restore' button which will return it to it's previous size.

Console Panel

This displays messages sent to Unity's console via the usual `Debug.Log()`, `Debug.LogWarning()` and `Debug.LogError()`. Uncaught exceptions will also be shown, with the stack trace reformatted to be more space efficient.

The buttons at the bottom can be used to toggle visibility of info, warnings and errors separately. The 'Clear' button will clear all log messages shown on the VR console.

Stats Panel

Stats are a different way of inspecting internal game state. They are particularly useful for watching fast changing internal values - such as what state a state machine is in, or internal cooldown values for gameplay mechanics. Values which if output to Debug.Log would heavily spam the console and hide other output.

These are logged by calling `VrDebugStats.SetStat()`. Stats are a name with an associated value (plus an optional category). Setting a stat either creates a new one if one does not already exist, or overwrites the existing value if one does exist.

Pressing the 'Clear' button on the vr panel will clear all stats. The arrows can be used to cycle through different categories of stats.

For an example usage, see
`Assets/Technie/VirtualConsole/Scripts/Example/InfoExample.cs`

Auto Break Panel

The auto break panel effectively lets you press the 'pause' button in Unity, but without having to leave VR. This is especially useful if you're trying to debug something that requires holding an object or a controller in the air and you can't reach your keyboard (or have run out of hands).

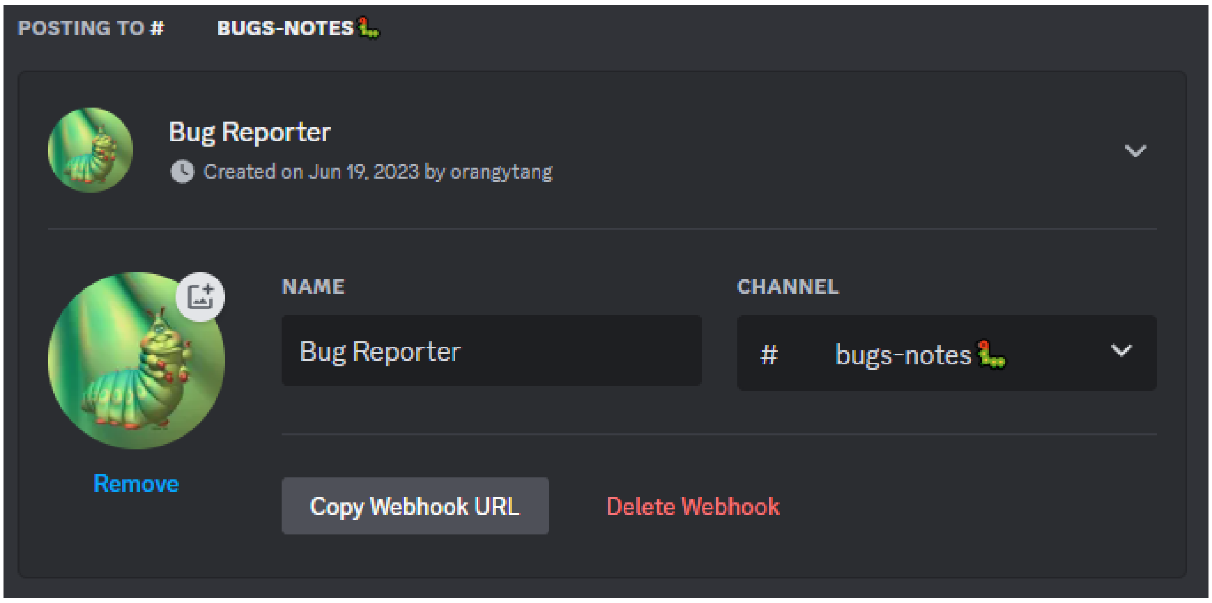
The 'Delayed Pause' is the same, but has a timer before it pauses (like a selfie timer on a camera). Set how long you want the delay to be, then press 'Delayed Pause'. You now have a few seconds to pick up that thing, position your head and hands to recreate the bug you're trying to fix, etc. before Unity will pause and you can take the headset off to debug it.

Screenshot Panel

Here you can take a screenshot from the VR camera which is saved as an image in your project. Adjust the screenshot timer and press ‘Take Screenshot’, and your current view will be captured.
Screenshots will be saved to a ‘Screenshots’ folder in the root of your project.

Discord Bug Report

Frequently we find that doing testing in VR is a two-person job, with one person in the VR headset playing the game, and another person on the PC taking screenshots and writing notes. With this bug report panel, one person can do both at the same time, which is easier and captures more bug information. You can capture a screenshot and enter a bug title and description from this panel, then send the bug report to your Discord server, along with collected logs and other automatically captured information.


POSTING TO # BUGS-NOTES

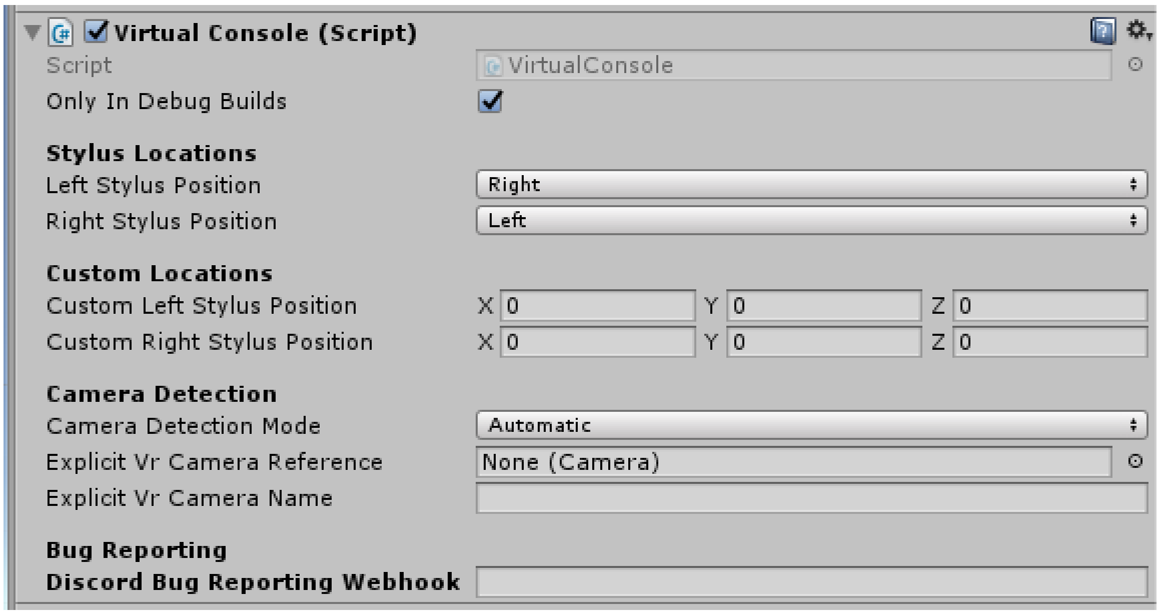
Bug Reporter
Created on Jun 19, 2023 by orangytang

NAME: Bug Reporter

CHANNEL: # bugs-notes

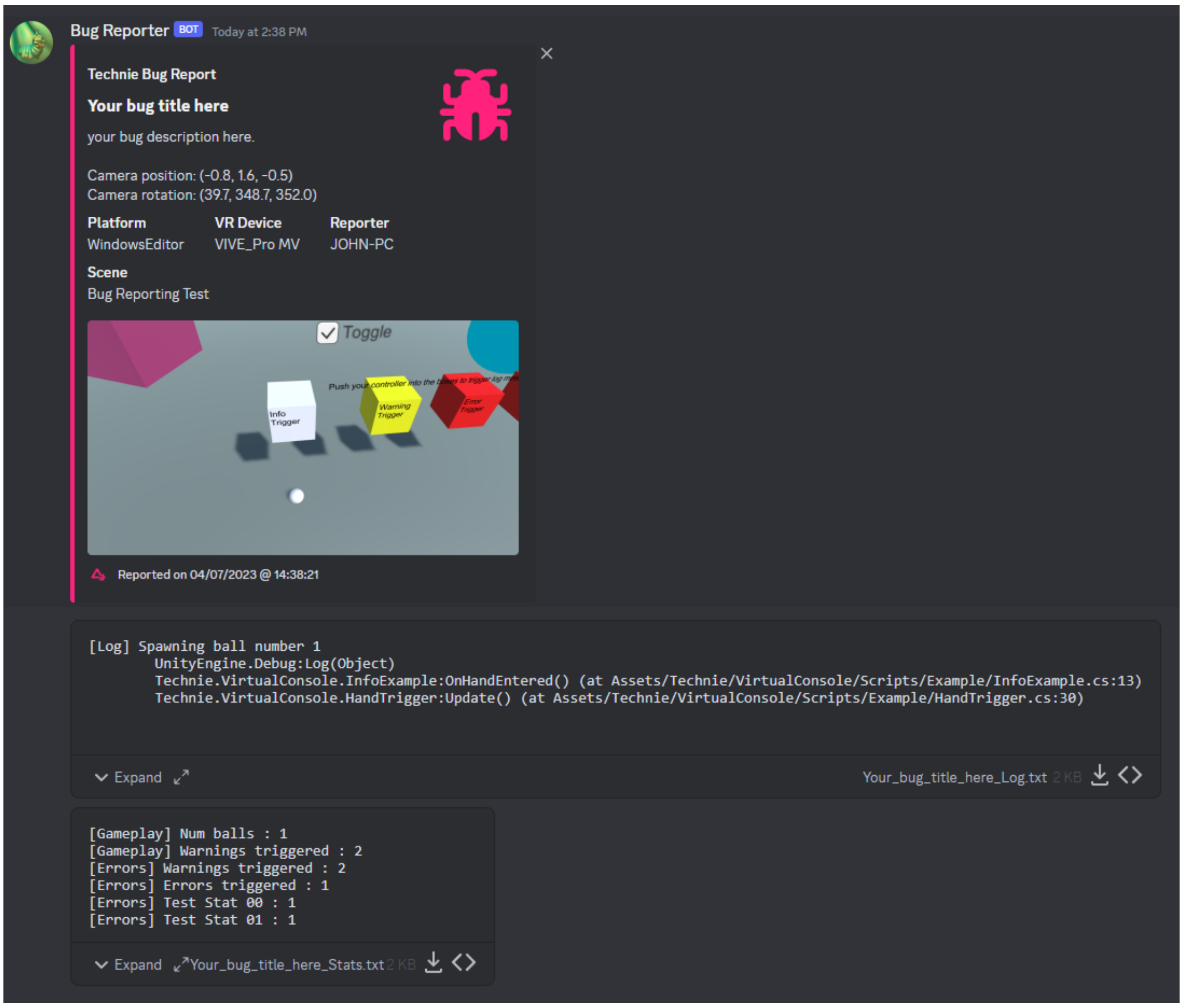
You will first need to create a Discord webhook. To do this go to your team's Discord server, and choose 'Server Settings'. You will need to be an admin to do this. Then choose 'Integrations' from the list, and then 'Webhooks' and 'New Webhooks'. Assign a name and a channel - you may want to create a channel just for bugs. Then copy the webhook url.

Back inside unity select the root of the Virtual Console prefab and paste the webhook url into the 'VirtualConsole' component


Virtual Console (Script)

Script	VirtualConsole
Only In Debug Builds	✓
Stylus Locations	
Left Stylus Position	Right
Right Stylus Position	Left
Custom Locations	
Custom Left Stylus Position	X 0 Y 0 Z 0
Custom Right Stylus Position	X 0 Y 0 Z 0
Camera Detection	
Camera Detection Mode	Automatic
Explicit Vr Camera Reference	None (Camera)
Explicit Vr Camera Name	
Bug Reporting	
Discord Bug Reporting Webhook	

You can now use the in-VR report bug panel to capture a screenshot, enter a title and description and press 'Submit Bug' to send it to your discord channel.


Bug Reporter BOT Today at 2:38 PM

Technie Bug Report

Your bug title here

your bug description here.

Camera position: (-0.8, 1.6, -0.5)
Camera rotation: (39.7, 348.7, 352.0)

Platform	VR Device	Reporter
WindowsEditor	VIVE_Pro MV	JOHN-PC

Scene
Bug Reporting Test

Reported on 04/07/2023 @ 14:38:21

```
[Log] Spawning ball number 1
        UnityEngine.Debug:Log(Object)
        Technie.VirtualConsole.InfoExample:OnHandEntered() (at Assets/Technie/VirtualConsole/Scripts/Example/InfoExample.cs:13)
        Technie.VirtualConsole.HandTrigger:Update() (at Assets/Technie/VirtualConsole/Scripts/Example/HandTrigger.cs:30)
```

Expand   Your_bug_title_here_Log.txt 2 KB

```
[Gameplay] Num balls : 1
[Gameplay] Warnings triggered : 2
[Errors] Warnings triggered : 2
[Errors] Errors triggered : 1
[Errors] Test Stat 00 : 1
[Errors] Test Stat 01 : 1
```

Expand   Your_bug_title_here_Stats.txt 2 KB

We are considering additional bug reporting panels for different backends (such as reporting to HacknPlan). If you have a particular backend you'd like to use for bug reporting then please contact us (see support section at the bottom of this readme).

Stylus Locations

By default the styluses attached to your hands will appear next to your thumbs. Depending on your game or control scheme, you may want them elsewhere. On the root of the Virtual Console prefab you can choose from several pre-set stylus positions. Experiment and see which works best for you.

If you need to you can specify exactly where the stylus goes. Choose `Custom` from the drop down, and then enter the position you want them to appear in the 'Custom Stylus Position' boxes. These positions are relative to the origin of the controllers.

Problems? Feature Requests? Bugs?

Send an email to `technie@triangularpixels.com` for support and feature suggestions.

Please include your Unity version, OS and VR platform in any support emails. If reporting a bug then if you include a (small!) reproduction project with instructions on how to reproduce your bug then we'll be able to fix things *much* quicker. Thanks!

Or head to our Discord at https://discord.gg/VzhNeZGxpv and join the Technie channel