Unity wireless remote

1. Setup

Please install the mobile app to your device.

Android: https://play.google.com/store/apps/details?id=com.nick.wirelessRemote

iOS: https://itunes.apple.com/app/id1134666444

Import the package into your project, and make sure everything is selected. Once you've done that, open the config panel under 'Window/Remote'

This panel should pop up. It will automatically search for any devices running my wireless remote app.

WARNING: Make sure your PC and remote app device are on the same local network!!

You can click on any of the devices in the list in order to mark them as the device you want to connect to. Once you're in play mode, you can disconnect from your device and connect to any other one at any time.

If you already didn't do this, we have to drag in the prefab from the folder *Wireless Remote/Prefabs/Wireless Input Controller.prefab* into your scene. This prefab will handle incoming data and convert wireless touches into a new custom input system. so make sure no other input module or eventsystem is already in the scene, which might overwrite mine.

Your device is now ready for a connection when the play button is pressed.

Everything should be setup now. When you press play, a connection will be made with your remote device, and the screen from your unity editor

will be cast onto your device. You can check the incoming data of your device in the Wireless Input Controller's prefab inspector in your scene.

If you have any troubles with connecting or discovering devices, try to close the wireless remote config panel and reopen it, go back to the device list if it's not showing and press the refresh button. Also close the wireless remote app on your device and restart it.

2. UGUI and wireless remote

To interact with the unity GUI, I've created a custom input module. This will handle all touch events and translate them to interact with the GUI. This is a component of the standard Wireless Input Controller prefab that is needed in every scene.

If the UGUI is not showing on your remote device, go to your canvas settings and set the render mode to Screen Space - Camera, drag your main camera into the slot and set the plane distance to 1.

3. Coding

We can also grab different data that is send from out device. Currently the device will send:

- gyroscope data
- Accelerometerdata
- Touch data

All of the data is processed and will be updated every frame in the WirelessInputController class. To acces your device's rotation:

```
Quaternion deviceRotation =

Quaternion.Euler(WirelessInputController.DeviceData.GyroData);
```

To acces your device's accelero data:

```
Vector3 deviceAccelero =
WirelessInputController.DeviceData.AcceleroData;
```

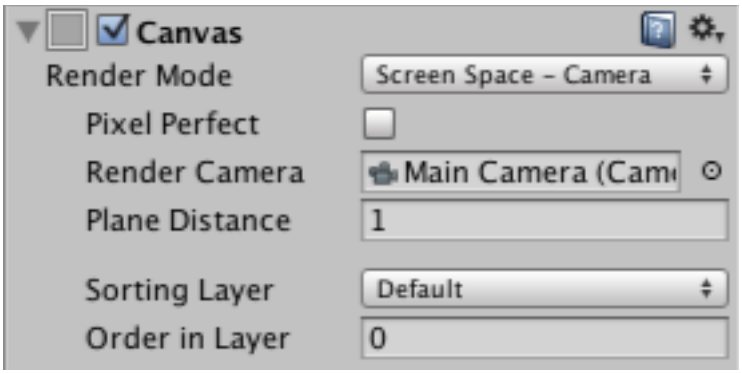
I can't overwrite unity's Input or Touch events, so I had to create my own types. I replaced Touch with WirelessTouch, which also has all the functionality the Touch class provides. When the code is compiled to a mobile device, the WirelessTouch class will grab the native Touch data, so there is no need to change your code. Same applies for the other data above.

```
for(int i = 0; i < WirelessInputController.DeviceData.TouchData.Length; i
++)
 {


WirelessTouch t = WirelessInputController.DeviceData.TouchData[i];


}
```

4. Settings



In the Config Panel we also have a settings tab:

Screenshare quality: The compression quality of the images send from the editor to the wireless device. A lower value will give poor quality, but

faster. A Higher value will give better quality, but will be much slower. Very high qualities might result in too much data to be processed so the device can't handle it. A flickering screencast will be seen. Just lower this setting to get the best result for your device.

Device orientation: Lock the device to the specified orientation. This is helpful when you're using the gyro or accelero data in your project, and have to tilt your devices in ridiculous angles :). That way the device won't flip it's screen.

5. Example Scenes

Under the folder Wireless Remote/Example you can find 3 example scenes with code on how this stuff works.

That's it!

Any feedback is greatly appreciated. If you want to report a bug, suggest a feature or simply want to say hi, email me at nick.peelman@gmail.com