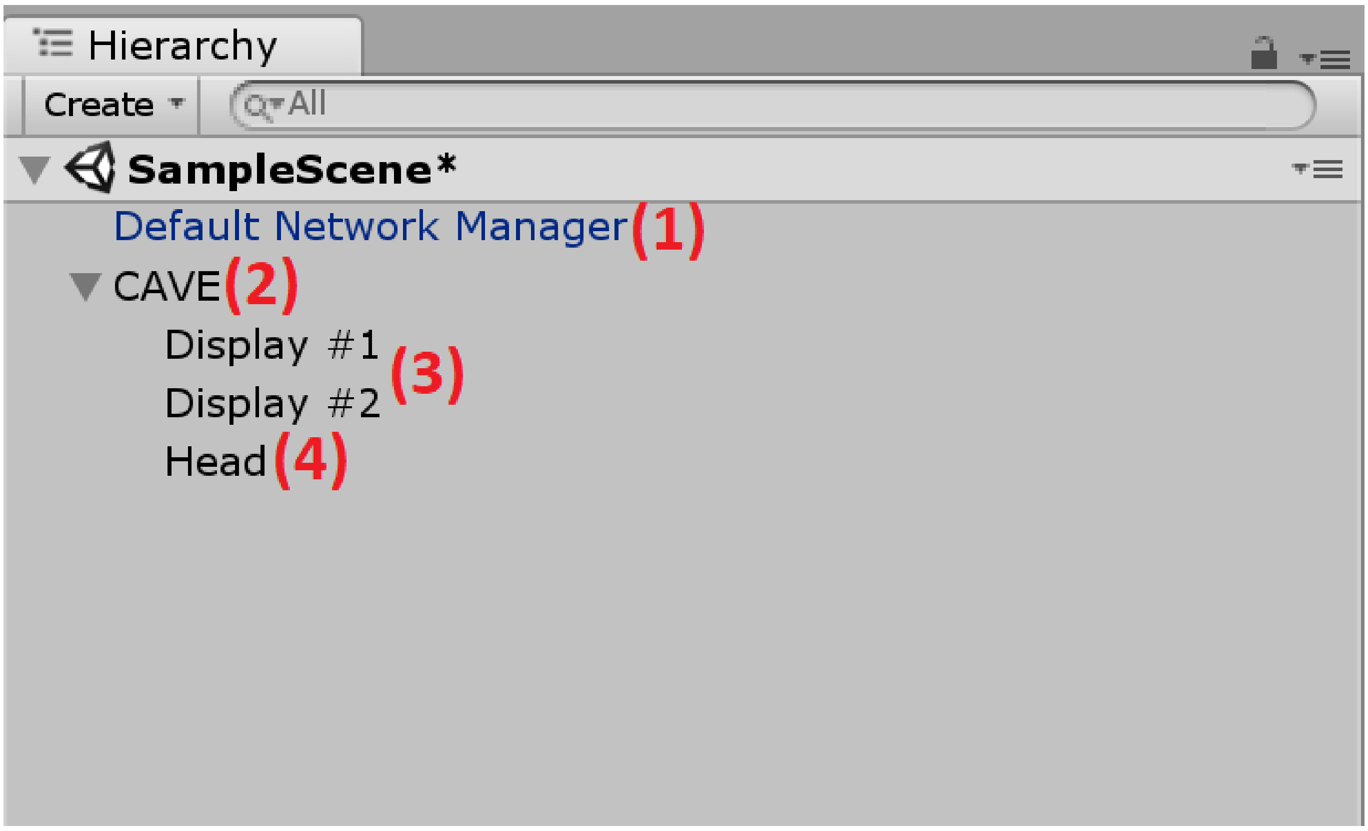
Basic Custom Setup

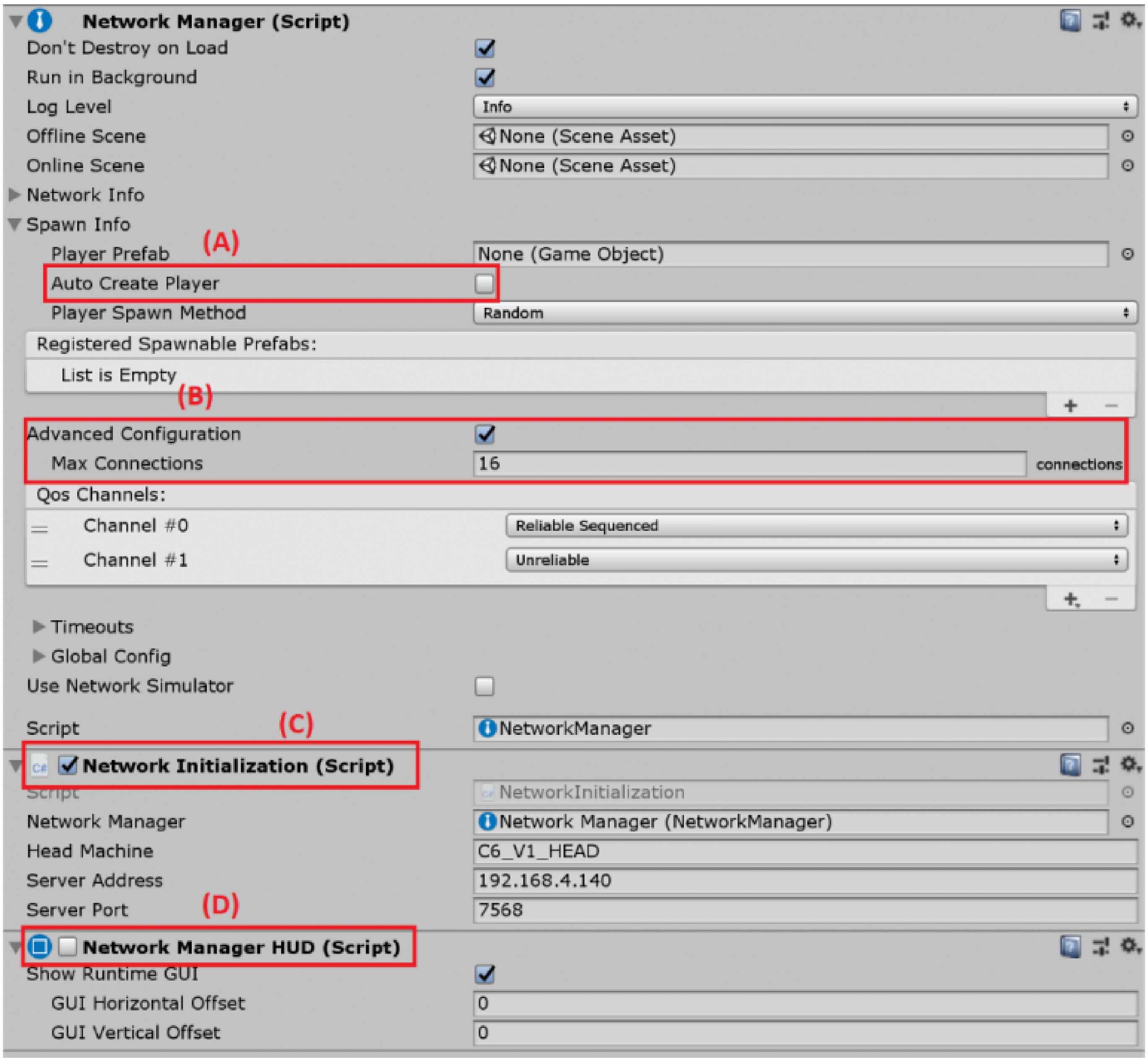
A basic setup consists of four parts:



1. A GameObject with Network Manager script and Network Initialization script (see below)
2. A GameObject with UCNetwork script attached
3. An assortment of GameObjects with PhysicalDisplay scripts attached
4. A GameObject with HeadConfiguration script attached (And often a VRPNTrack script)

Parts **2-4** involve simply configuring the respective scripts as described in their sections, but Part **1** involves a few more scripts and configuration.

Object **1** in the Basic Setup:



Network Manager (Script)
Don't Destroy on Load ✓
Run in Background ✓
Log Level: Info
Offline Scene: None (Scene Asset)
Online Scene: None (Scene Asset)
Network Info
Spawn Info
Player Prefab: None (Game Object)
(A) Auto Create Player ☐
Player Spawn Method: Random
Registered Spawnable Prefabs:
List is Empty
(B) Advanced Configuration ✓
Max Connections: 16 connections
Qos Channels:
Channel #0: Reliable Sequenced
Channel #1: Unreliable
Timeouts
Global Config
Use Network Simulator ☐
Script: NetworkManager

(C) Network Initialization (Script) ✓
Script: NetworkInitialization
Network Manager: Network Manager (NetworkManager)
Head Machine: C6_V1_HEAD
Server Address: 192.168.4.140
Server Port: 7568

(D) Network Manager HUD (Script) ☐
Show Runtime GUI ✓
GUI Horizontal Offset: 0
GUI Vertical Offset: 0

A. Disable “Auto Create Player” on the Network Manager Script

B. Enable “Advanced Configuration”, ensure that “Max Connections” is enough for your setup

C. Configure your Network Initialization script as described

D. For debugging, it is useful to use a `NetworkManagerHUD` script to manually start a server or client, which is normally done by the `NetworkInitialization` script based on the machine name Unity is currently running on.

Additional Command Line Options:

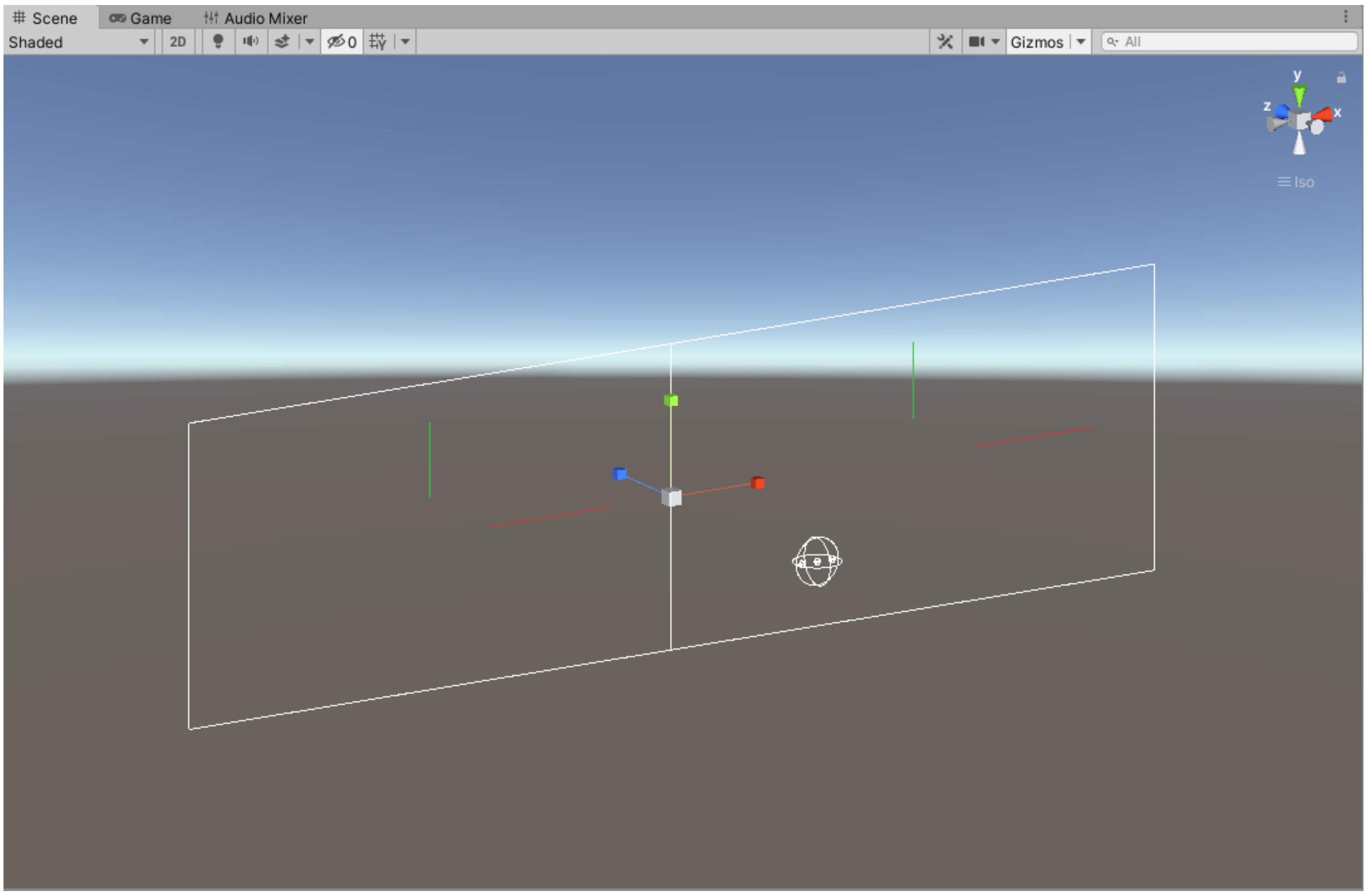
Various command line options are available to customize your setup. See them here.

Additional Player Settings – Enabling Active Stereo

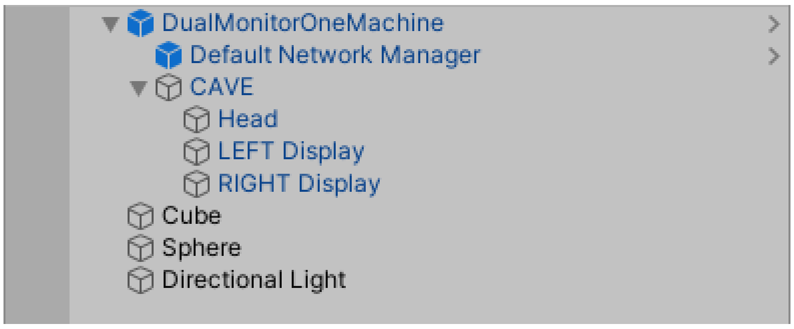
To enable active stereo setups in Unity: go to *Edit->Project Settings->Player* and enable *Virtual Reality Supported*, add *Stereo Display (non head-mounted)* to the list of SDKs, and remove the other SDKs.

Dual Monitor Setup

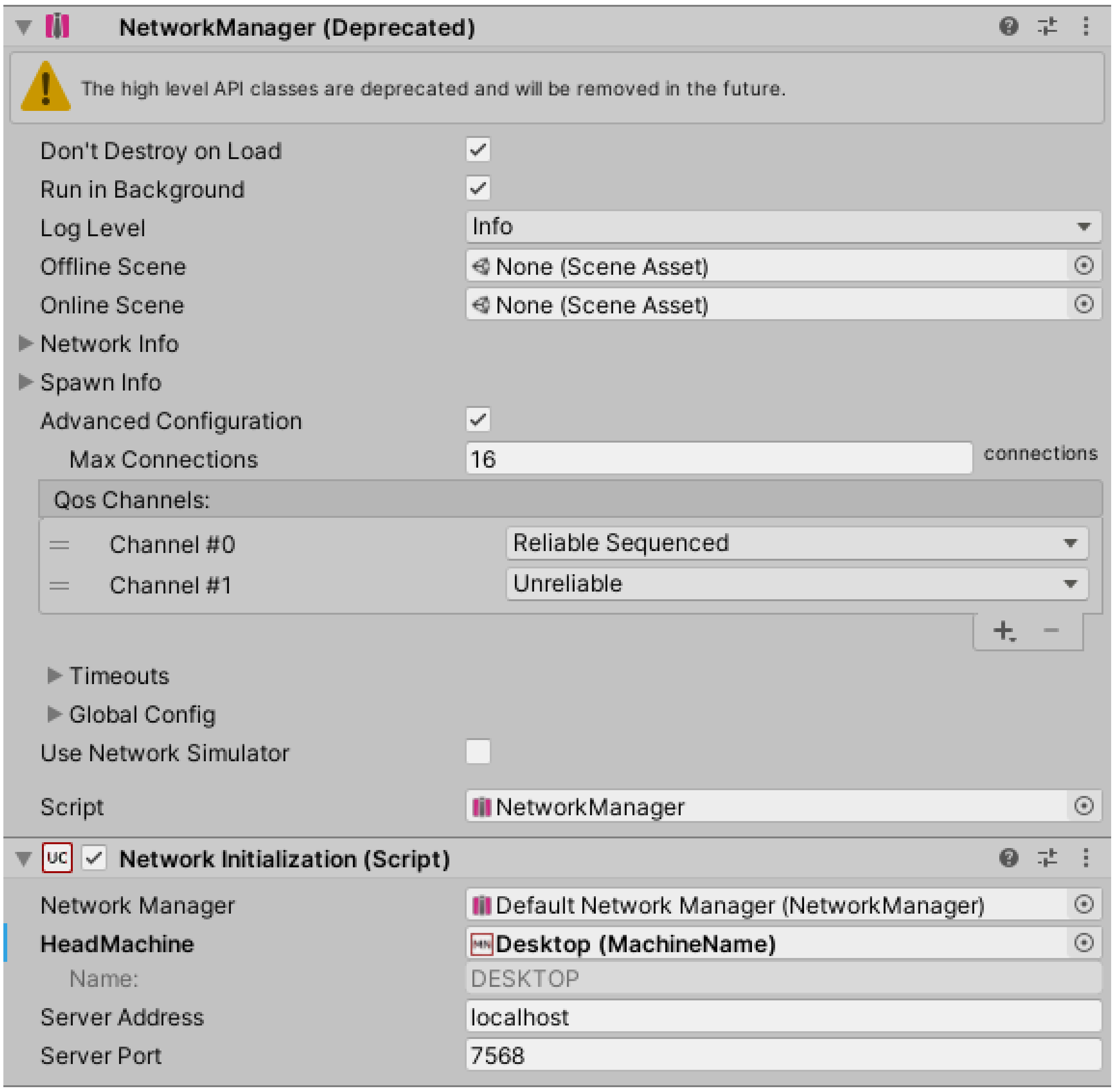
Setting up UniCAVE to run on a single machine with two displays is relatively straightforward. An example is included with the **DualMonitorOneMachine** prefab in the Prefabs folder of the UniCAVE plugin.



The prefab contains the following components:


DualMonitorOneMachine
Default Network Manager
CAVE
Head
LEFT Display
RIGHT Display
Cube
Sphere
Directional Light

Network Manager



NetworkManager (Deprecated)

The high level API classes are deprecated and will be removed in the future.

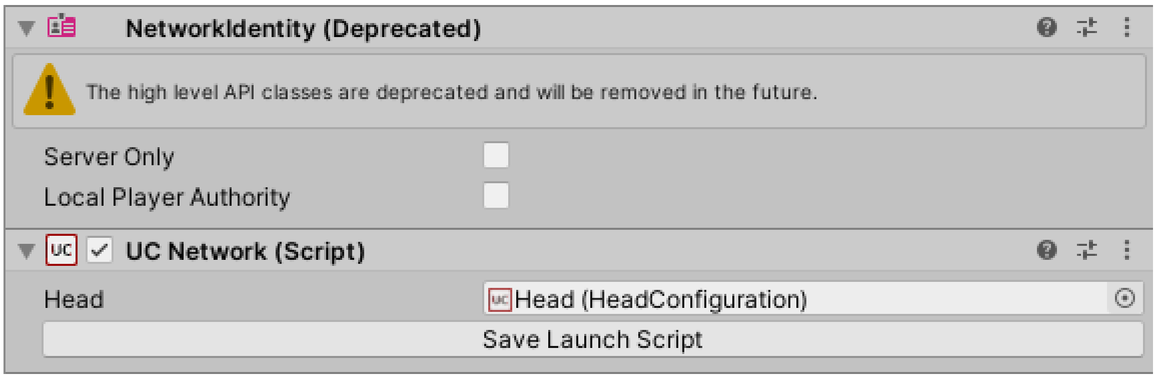
Don't Destroy on Load	✓
Run in Background	✓
Log Level	Info
Offline Scene	None (Scene Asset)
Online Scene	None (Scene Asset)
Network Info	
Spawn Info	
Advanced Configuration	✓
Max Connections	16 connections
Qos Channels:	
Channel #0	Reliable Sequenced
Channel #1	Unreliable
Timeouts	
Global Config	
Use Network Simulator	
Script	NetworkManager

Network Initialization (Script)

Network Manager	Default Network Manager (NetworkManager)
HeadMachine	Desktop (MachineName)
Name:	DESKTOP
Server Address	localhost
Server Port	7568

The Network Manager simply needs to be setup with a `MachineName` matching your local machine. This same `MachineName` will be used for both `PhysicalDisplays`.

CAVE



NetworkIdentity (Deprecated)

The high level API classes are deprecated and will be removed in the future.

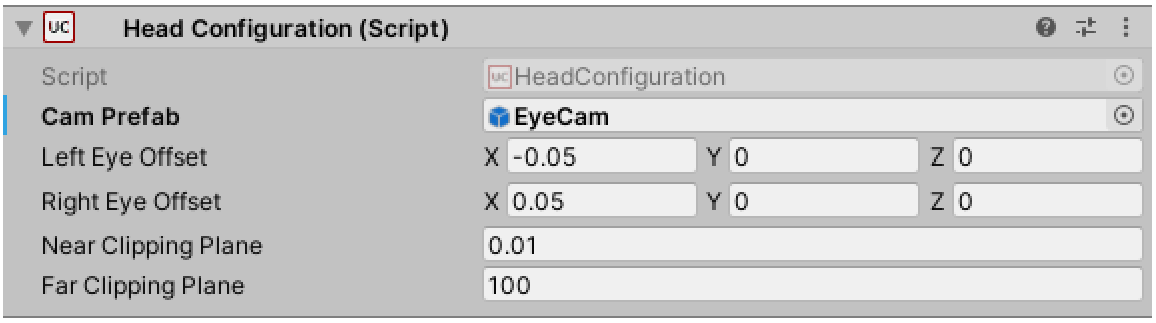
Server Only	
Local Player Authority	

UC Network (Script)

Head	Head (HeadConfiguration)

The CAVE object simply serves as an organizational object that holds the necessary `UCNetwork` component.

Head Configuration

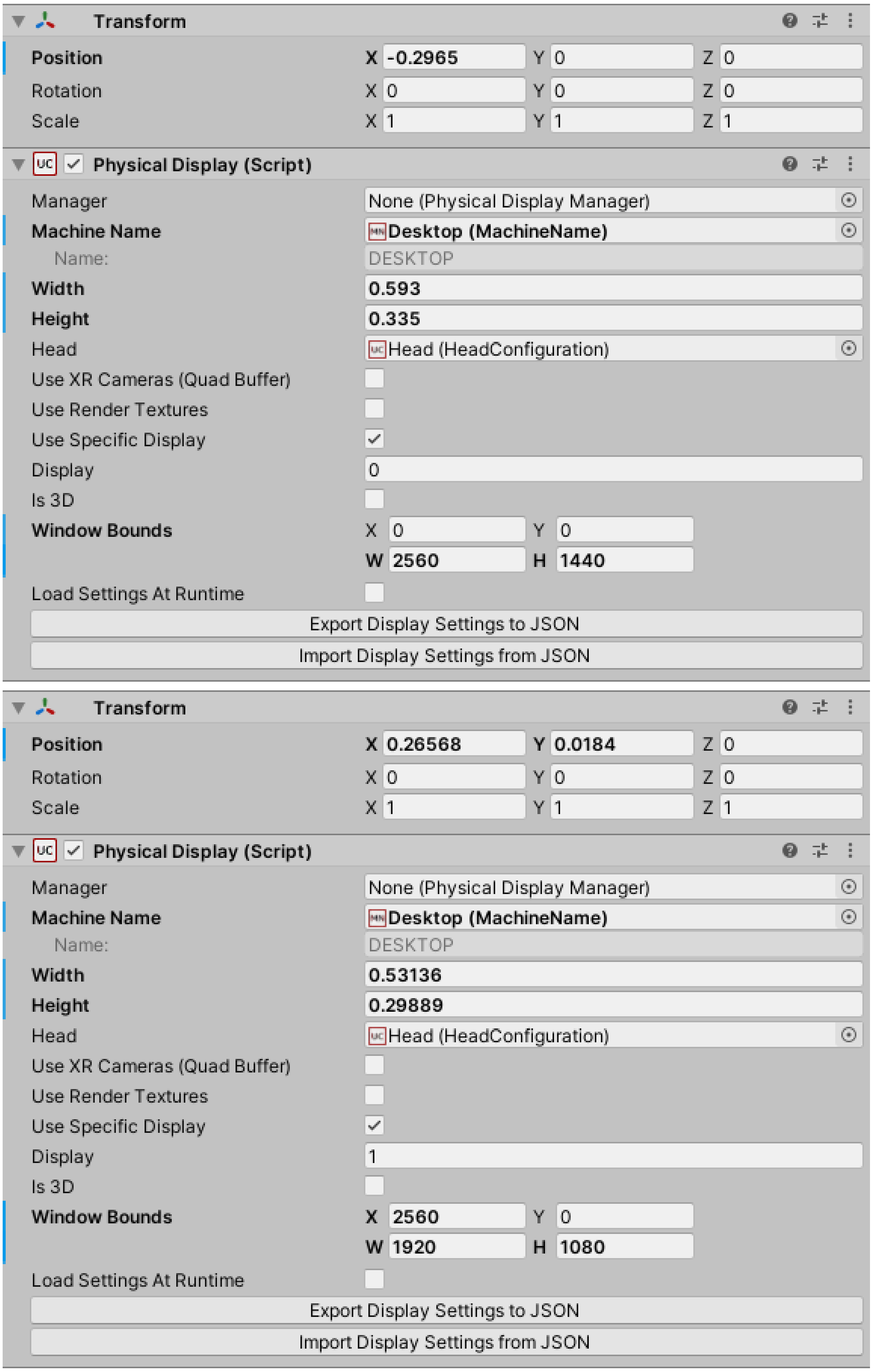


Head Configuration (Script)

Script	HeadConfiguration
Cam Prefab	EyeCam
Left Eye Offset	X -0.05 Y 0 Z 0
Right Eye Offset	X 0.05 Y 0 Z 0
Near Clipping Plane	0.01
Far Clipping Plane	100

The Head Configuration component needs to be given a reference to a Camera prefab. These Cameras will be spawned for each `PhysicalDisplay` when entering play mode.

Physical Displays



Transform
Position X -0.2965 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

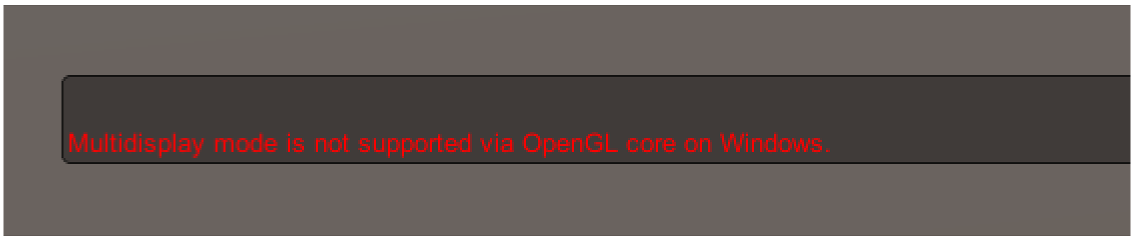
Physical Display (Script) ✓
Manager: None (Physical Display Manager)
Machine Name: Desktop (MachineName)
Name: DESKTOP
Width: 0.593
Height: 0.335
Head: Head (HeadConfiguration)
Use XR Cameras (Quad Buffer): ☐
Use Render Textures: ☐
Use Specific Display: ✓
Display: 0
Is 3D: ☐
Window Bounds: X 0 Y 0 W 2560 H 1440
Load Settings At Runtime: ☐
Export Display Settings to JSON
Import Display Settings from JSON

Transform
Position X 0.26568 Y 0.0184 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Physical Display (Script) ✓
Manager: None (Physical Display Manager)
Machine Name: Desktop (MachineName)
Name: DESKTOP
Width: 0.53136
Height: 0.29889
Head: Head (HeadConfiguration)
Use XR Cameras (Quad Buffer): ☐
Use Render Textures: ☐
Use Specific Display: ✓
Display: 1
Is 3D: ☐
Window Bounds: X 2560 Y 0 W 1920 H 1080
Load Settings At Runtime: ☐
Export Display Settings to JSON
Import Display Settings from JSON

For two monitors, we need two `PhysicalDisplays`. In this case, no `PhysicalDisplayManager` is needed. Instead, each `PhysicalDisplay` has its 'UseSpecificDisplay` option checked. The left display is set to 0 (for Display 1) and the right display is set to 1 (for Display 2).

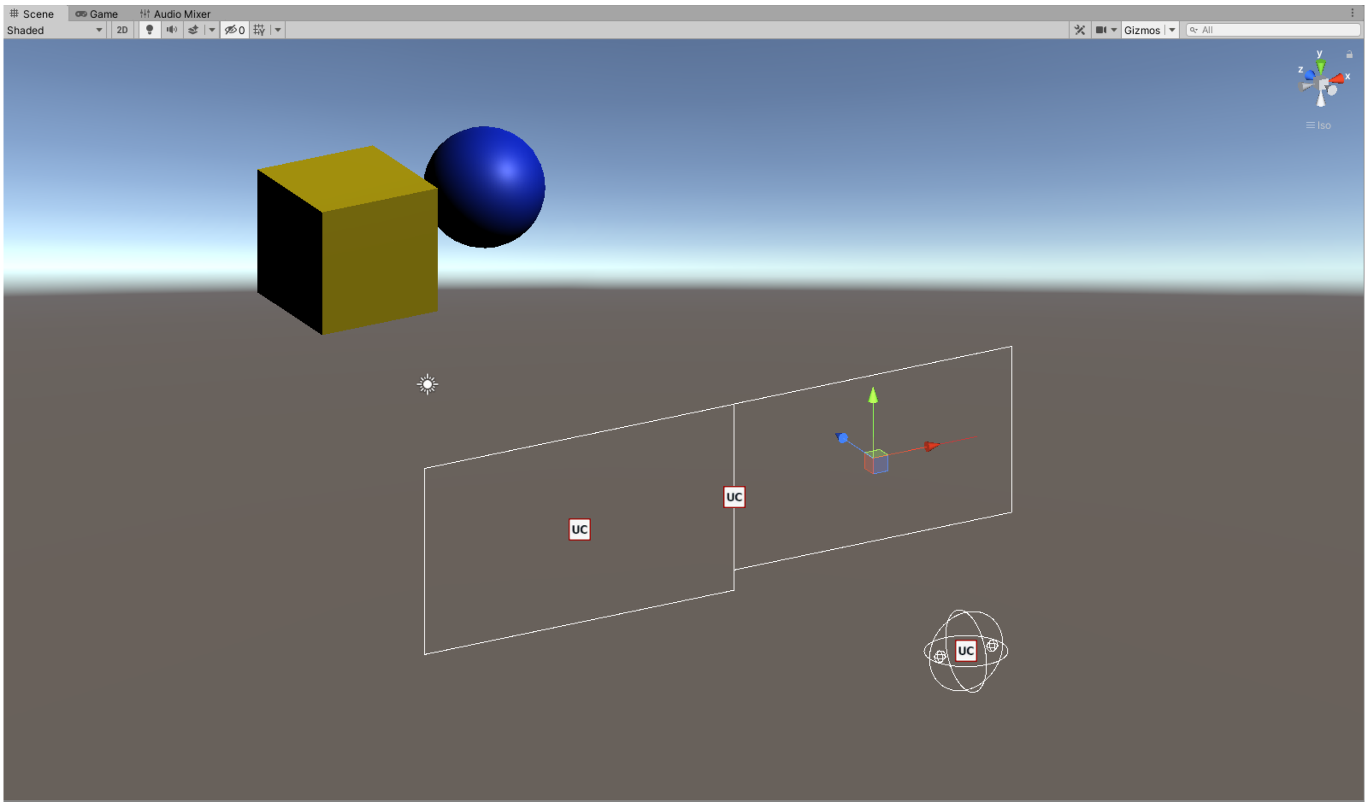
To ensure a seamless image, the resolution, size, and position of the displays have to be entered as well. In this case, the two monitors are different resolutions and sizes. Simply measure your displays and enter those values (in meters) into the appropriate boxes: `Width` and `Height` for the physical dimensions of your display, `WindowBounds X` and `Y` for the pixel offsets of each display in Windows, and `WindowBounds W` and `H` for the horizontal and vertical resolution of each display in pixels.



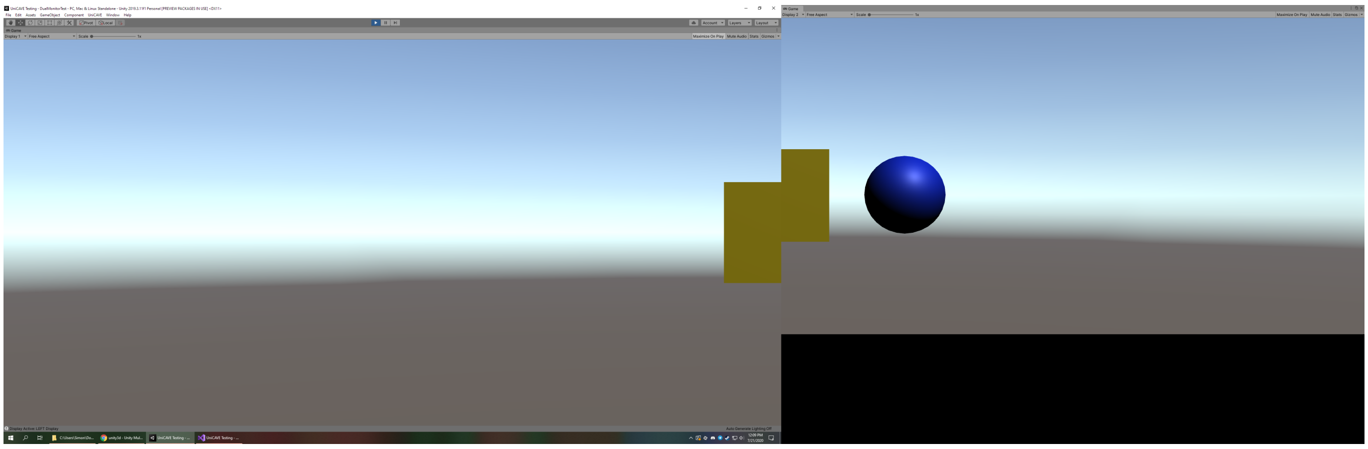
Multidisplay mode is not supported via OpenGL core on Windows.

Note: your project must be using DirectX in order to support identifying displays by their number. Make sure this is the selected Graphics API in Project Settings->Player->Other Settings->Rendering. If you are not using the correct API, Unity will display an error message in the console.

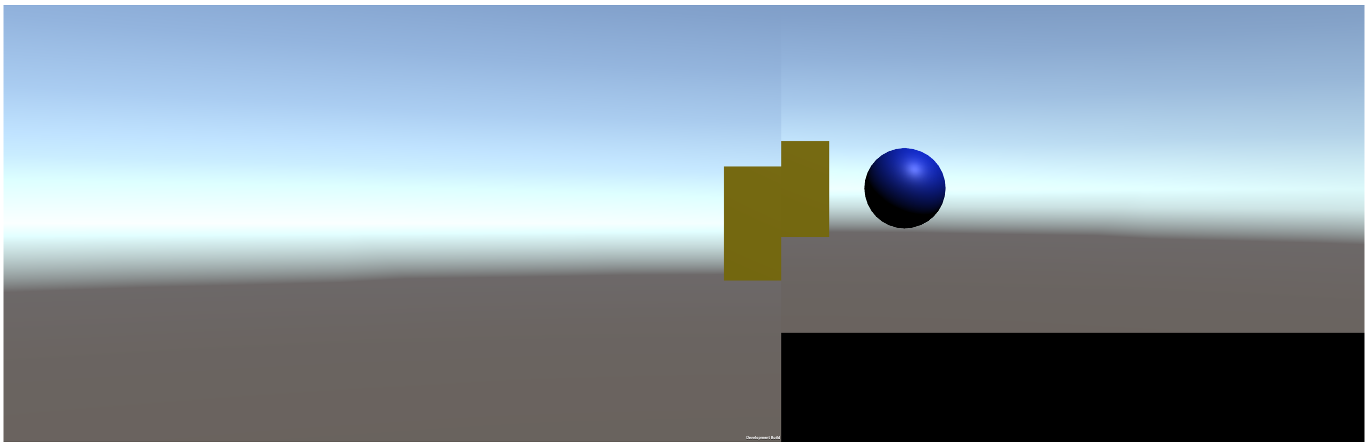
Testing your Setup



Here's what my setup looks like in the Scene view.



You can test the positioning of your displays in the Editor by entering playmode and opening multiple Game tabs. Simply maximize a tab in each display and set the Display number (in the upper left corner) to match each `PhysicalDisplay`. This won't be entirely accurate because Unity's menu bar takes up some space, but it's a good way to check you've entered the right values.



When ready, make a build as a final test. If everything is setup correctly, both displays should run in fullscreen and display a seamless image. It looks offset in the screenshot, but due to the different resolutions and physical dimensions of my setup, the final display is seamless.