Proposal:
VX1- Unity / Unreal Engine 4 Plugin

Authored By: Ben Weatherall for Voxon Productions

Context

Voxon Photonics are creators of the Voxon VX1 – the world's most advanced three-dimensional (3D) volumetric display. Incorporated in 2013, Voxon Photonics has teams based in Australia and the US that have been developing the 3D volumetric technology for more than nine years.

Voxon Photonics' 3D volumetric technology is unlike anything ever seen. In literal terms, Voxon Photonics has created a futuristic display technology capable of producing truly 3D digital images that can render animated or static imagery as genuinely three-dimensional holograms – high definition, infinitely complex and manipulable, and viewable with the naked eye from any angle. No illusions. No special glasses.

Problem

In order to speed adoption and engagement with the VX1, a simple standardised workflow needs to be available to clients to create new content for the device. The current solution offered is a proprietary SDK with the users able to develop their own solutions from scratch.

Related Work

Previously Gordon Larson worked with Voxon to develop a plugin for unity that would allow developers to utilise this popular development environment to speed the creation of content for the VX1.

Solution

The proposed solution is to continue the work of Gordon in developing a Unity and UE4 plugin to allow developers to create content for the VX1 using an established game development workflow.

The solution will revolve around using a plugin that developers can attach to a project, which will define a capture volume (whose aspect ratio reflects the VX1's) that will send all relevant geometry to the VX1.

This solution will allow easy content creation for the VX1 while taking advantage of the features built into the Unity and Unreal Engines.

Aims and Objectives

The aim is to create two plugins, one for Unity and another for UE4, which will allow the 'drag and drop' addition of VX1 functionality to a project, allowing developers to create content for the VX1 without needing to develop a new workflow specifically for the device.

To achieve this end we must ensure the plugins have the following features:

- 'Drag and Drop' integration
- A Capture Volume (Content within Capture volume to appear in VX1)
- Audio Support
- Controller Support
- Mouse and Keyboard Support
- Textured Models
- Light Models
- Compatibility with development environments (UE4 / Unity)
- Compatibility with VoxieOS (VX1's default operating environment)
- Code must be performant
    - Minimum Performance threshold of 30 fps

Tasks and timetable

Unity Plugin Version 1:

- Estimated Completion Time: 80 hours
- Features:
    - Unity Prefab plugin
        - Requires a (user) scalable Capture Volume, position-able and scalable within editor
        - Capture Volume must load aspect ratio from Voxon configuration file
        - Capture Volume must send all colliding geometry to VX1
        - Capable of setting non-capture of geometry
    - Use voxiebox.dll for graphics display via VX1 (emulator / physical device)
    - Game Audio
    - Xbox Controller input
- Performance:
    - Should be rendered at minimum of 30 fps

Unity Plugin Version 2:

- Estimated Completion Time: 40 hours
- Features:
    - Colour Textures
    - Lighting

Unity Plugin Version 3:

- Estimated Completion Time: 40 hours
- Features:
    - Only display VX1 window (no graphics / silent mode)
    - Loadable from within VoxieOS
    - Keyboard support
    - Mouse support

Unity Plugin TBA:

- Estimated Completion Time: To be assessed approaching completion of Unity Plugin
- Features:
    - Improve compatibility with Unity Editor
        - Enable ‘run in window’

Unreal Plugin Version 1:

- Estimated Completion Time: To be assessed approaching completion of Unity Plugin
- Features:
    - UE4 plugin
        - Requires a (user) scalable Capture Volume, position-able and scalable within editor
        - Capture Volume must load aspect ratio from Voxon configuration file
        - Capture Volume must send all colliding geometry to VX1
        - Capable of setting non-capture of colliding geometry
    - Use voxiebox.dll for graphics display via VX1 (emulator / physical device)
    - Game Audio
    - Xbox Controller input
    - Colour Textures
    - Lighting
    - Improve compatibility with UE4 Editor
        - Enable ‘run in window’
    - Only display VX1 window (no graphics / silent mode)
    - Loadable from within VoxieOS
    - Keyboard support
    - Mouse support
- Performance:
    - Should be rendered at minimum of 30 fps

		Hours																			
		8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152	160
Unity Plugin	Capture Volume	■																			
	Display on VX1		■	■																	
	Game Audio							■	■	■											
	Controller Input	■	■	■	■	■	■	■	■												
	Available as prefab									■	■										
	Textures											■	■	■	■						
	Lighting													■	■	■					
	Keyboard Support								■								■	■	■	■	
	Mouse Support								■									■	■	■	■
	Only VX1 Display	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
	VoxieOS Compatibility																	■	■	■	■
	Editor Compatibility																				

Challenges

VoxieOS Compatibility / VX1 Exclusive Display

VoxieOS compatibility will occur during the last 32 hours of the project, testing for this compatibility however will occur throughout the process as part of the VX1 Display exclusivity. Issues identified during VX1 testing will be resolved at this time. Depending on problems identified, this task may require additional time.

Editor Compatibility

Due to the more restrictive nature of coding for the Unity and Unreal Engines' editors is difficult to gauge how long will be required to complete non-conflicting integration between the VX1 SDK and each engines 'play in window' modes. This problem will need more investigation to determine the cause of the Editor cursor disappearing after play.

UE4 Plugin

Inclusion of the Unreal Engine 4 plugin's timeline has not been included as a thorough investigation of compatibility has yet to be completed. It's possible all systems developed to allow for Unity compatibility will be easily ported to UE4 and thus the timeline will be small, or the approached used for Unity may be wholly incorrect for the Unreal Engine. I believe a level of progress on the Unity plugin is required before pursuing UE4 compatibility.