Overview and Assessment of Unity Toolkits for Cave Automatic Virtual Environments and Wand Interaction

Kenneth A. Ritter III, Christoph W. Borst, and Terrence L. Chambers
College of Engineering, University of Louisiana at Lafayette, Lafayette, USA
Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, USA
E-mail: kar4499@louisiana.edu

Abstract

As the interest in Virtual Reality (VR) increases, so does the number of software toolkits available for various VR applications. Given that more games are being made with the Unity game engine than any other game technology, several of these toolkits are developed to be directly imported into Unity. A feature and interaction comparison of the toolkits is needed by Unity developers to properly suit one for a specific application. This paper presents an overview and comparison of three virtual reality toolkits available for developers using the Unity game engine. getReal3D, MiddleVR, and Reality-based User Interface System (RUIS) are analysed for VR interaction and display on multi-projection immersive environments like Cave Automatic Virtual Environments (CAVE)s. MiddleVR was found to have the highest performance and most versatile toolkit for CAVE display and interaction. However, taking cost into account, RUIS has an advantage as it is available for free under the Lesser General Public License (LGPL) Version 3 license.

1. Introduction

Virtual reality technology gives users the ability to interact with three-dimensional data, providing an interface that is potentially powerful to both static and dynamic information [1]. A key ingredient of a VR experience is interaction with a virtual world [2]. For human-computer interaction the most exciting, well-engineered, and commercially successful application of direct-manipulation interfaces lies in the world of video games [3]. Having over 45% of the global game engine market share, the Unity engine is the dominant global game development software, resulting in more games being made with Unity than with any other game technology [4]. Various VR application toolkits are available for the Unity game engine, and each has a unique set of features. There are several different approaches for interactions with input devices and displays. Before beginning this study, several questions had to be answered to get an idea of how to go about analyzing the toolkits. Questions such as: "What features should you look for when evaluating a toolkit?" and "Which toolkits are better suited for relevant interaction and display applications?"

A literature review was initially conducted to find out how other VR toolkits are being assessed. Performance, flexibility, and ease of use are figures of merit used to compare or assess different VR development applications[5]. Differences found when comparing VR platforms included: complexity, parallel decomposition, object implementation, and object control and interaction [6].Regarding interaction styles in development tools for VR applications, it was found that options available for design of interaction styles have been denoted as: command language, menu selection, form filling, and direct manipulation[7]. For Virtual Reality Aided Design (VRAD), a study was conducted regarding the ability of various VR tools for pointing, picking and line sketching[8]. Several advantages VRAD has against 2D CAD tools were listed as stereo vision, spatial input, six DOF, heat tracking, mixed reality, tangible interfaces, and interaction freedom[8]. To interact with objects in 3D virtual environments, hand-gesture-based user interface could potentially be employed in place of a mouse or keyboard [9]. With several interaction techniques available for selection and manipulating objects, the choice is dependent on the characteristics of the application [10]. The ray casting technique, referred to as wand interaction in this study, projects a ray into the direction that the user points to select and manipulate objects.

This report analyzes the getReal3D, MiddleVR, and RUIS toolkits and looks specifically at CAVE display and wand interaction. An overview for each toolkit is initially given followed by CAVE display, interaction, and steps to implement applications with the toolkit in Unity. Following the explanation of each toolkit, comparative analysis is given with a rubric system measuring different strengths and weakness of each toolkit. The CAVE display used for the study is the 3-sided CAVE in the Rougeou Hall VR lab at the University of Louisiana at Lafayette shown in *Figure 1*. One computer was used with multiple graphics cards capable of running the three projectors and desktop display simultaneously. For the comparative analysis, an interaction test bed task is made of an alternative energy technician who must remove the doors of an Electrotherm Green Machine to view the internal components. A direct comparison of the toolkits regarding interaction is not possible because they do not all support the same interaction devices. All the toolkits have a wand interaction which is used for configuring the scene. Due to obvious differences in interaction devices, trying to compare a given task with different ones would not be feasible. However, MiddleVR and RUIS both support the Razer Hydra therefore this device will be used for the interaction testbed scene. This is an assessment of what these toolkits provide.

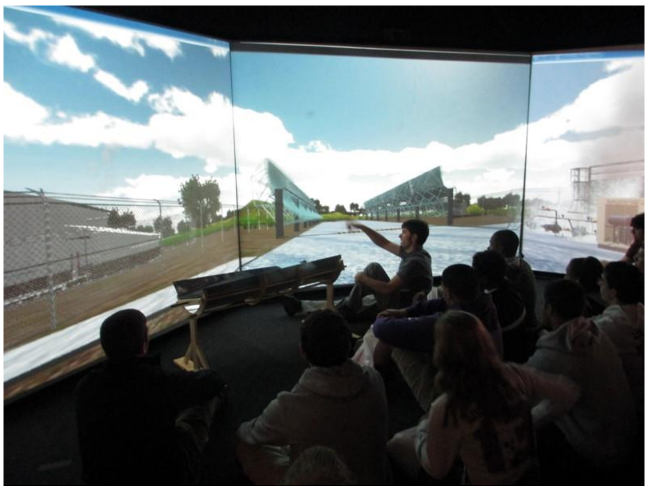


Figure 1: 3-sided CAVE in Rougeou Hall VR lab at the Univesity of Louisiana, Lafayette.

2. Methodology

To properly assess the current, early 2015, toolkits and compare relevant tasks the following methodology was created:

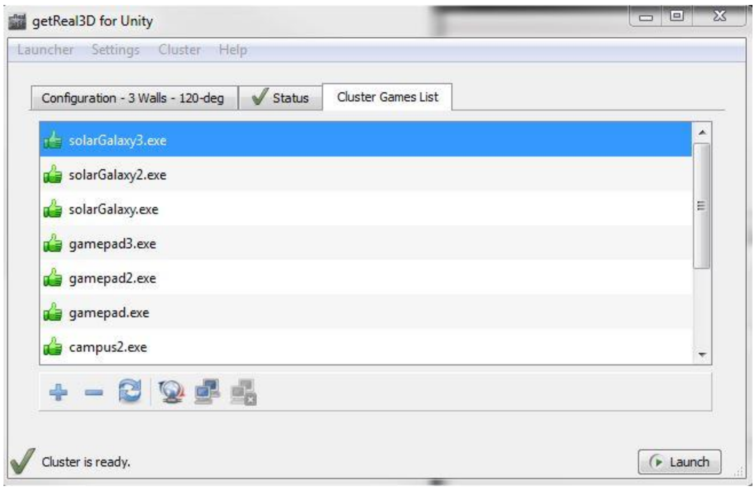
1. Technology review for Unity VR toolkits and interaction devices
2. Survey key features for relevant tools and select ones to compare
3. Create Unity testbed scene with relevant interaction tasks
4. Install tool kits and assess key performance measures
5. Create rubric system for comparison

3. getReal3D

3.1 Overview

Aimed at 3D immersive environments for more realistic training and simulation scenarios, Mechdyne Corporation released getReal3D as a Unity toolkit that brings 3D and viewer-based perspective to a variety of

immersive 3D displays. The toolkit supports stereo 3D, multiple video channels, viewer-centric perspective (head tracking), and tracked interaction.
The main components of the getReal3D toolkit are the configuration file, daemon, launcher, and plugin. The configuration file contains the display system, hardware, input devices, and many other settings that define how and where games will run. The getReal3D daemon must always be running on the workstation on which the game is running on. The launcher, shown in *Figure 2,* allows the user to run and deploy Unity games and consists of three sections: configuration, status, and games [11]. Finally the plugin is embedded inside the Unity project to provide run-time VR behaviors [12]. The getReal3D prefab also includes a skybox prefab to correctly display CAVE skyboxes in Unity.



solarGalaxy3.exe
solarGalaxy2.exe
solarGalaxy.exe
gamepad3.exe
gamepad2.exe
gamepad.exe
campus2.exe

Cluster is ready.

Figure 2: The getReal3D launcher interface.

3.2 CAVE display

Games created within the Unity game engine can run in multiple virtual reality environments using the getReal3D toolkit. Given the VR-specific data for a particular VR device, the toolkit generates the necessary cameras for the desired display system. This camera configuration allows multi-view, the ability to synchronize game state, where a single Unity games state is synchronized across multiple running instances [12]. Although not used in this study, the getRealCameraUpdater.cs script provided can be used to create a viewable application in a 72 display CAVE2 [13].

3.3 Interaction

The getReal3D toolkit uses user-center perspective where the game view will be in the perspective of a tracked user inside the VR environment. The software comes with the "trackd" package, which is a combination of trackd Server and trackd Daemon. The trackd Server is started to open a connection to tracker and/or controller devices and sends the data to a trackd Daemon that collects and stores the data in shared memory. For different interaction devices, shown in *Table 1,* the user must load the required configuration files for trackd.conf and trackdserver.conf. The tracker and controller input must be configured in these files.

Table 1: trackd Supported Devices [14].

Tracker Systems	Input Controllers
3D-Bird™	CubicMouse™
FASTRAK®	FlyBox
Flock of Birds®	Intersense Wand
InterSense IS600	NeoWand™
InterSense IS900	PINCH® Gloves
Is_Trackers (generic	
support for Intersense)	SpaceBall 3003
laserBIRD®	SpaceBall 4000
MotionStar®	SpaceBall 5000
pcBIRD®	SpaceOrb
SpacePad®	SpaceGrips™
OptiTrack™	V-Wand™
PPT™	Wand™
	Wanda™
	WorkWand™
	USB Gamepads
	FlyStick2
	Wiimote and Nunchuck

In order to run the game in the editor, the user needs to start the trackd simulator shown in *Figure 3*. The trackd simulator provides a GUI with head/hand sensors, joystick valuators, and buttons which are needed to navigate and manipulate objects in the Unity editor.

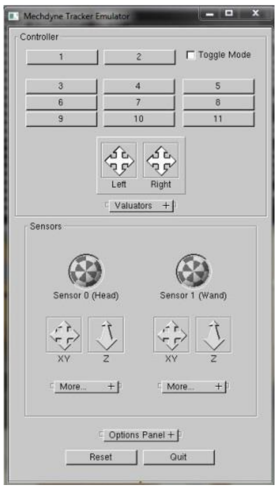


Figure 3: The trackd Simulator.

3.4 Using getReal3D in Unity

The steps required to create, navigate, and manipulate a Unity scene in a CAVE display are given below. The getReal3D developer license is required to use the getReal3D launcher to test cluster synchronization. To run the current getReal3D version 3.01 tool kit, Windows 7, DirectX 11, and Unity 4.5 with a Pro license are needed. To start with the getReal3D toolkit, the user must import the custom package into Unity and add the required prefabs into the Unity scene. This will create a first person, head centered, perspective with standard navigation.

1) Open or restart Unity 3D and import the getReal3D toolkit into the scene

- The user must not play the scene inside the editor before importing the package or the import will fail to update the core plugin (gr_plugin.dll).

2) Drag the getRealPlayerController.prefab into the Unity scene to desired start position.
   - This comes with controller scripts and three objects, Hand, Head, and Main Camera. This is a first person view.
3) Drag the WandManger.prefab to the Hand object
   - Contains three objects, CollidingWand, GrabbingWand, and PointingWand, with interaction scripts.
4) Delete or disable any Camera objects in the scene.
   - getReal3D manages all the VR cameras.
5) Add a Collider, Box or Mesh, and Rigidbody to all objects to be interacted within the scene.
6) Click menu item getReal3D – Scene Checker
   - This will open a window to show the status of the current scene and give suggestions for remaining steps to run on getReal3D cluster.
7) Start trackd simulator
   - This provides a GUI with head/hand sensors, joystick valuators, and buttons, which is needed to navigate and manipulate objects in the Unity editor
8) Press play to test scene in Unity editor
   - For navigation and object manipulation, use mouse to interact with Mechdyne Tracker Emulator. Press 2 to change wand type.
9) Build executable
   - Player settings automatically configured:
     i. Default Is Full Screen : Off
     ii. Run In Background : On
     iii. Capture Single Screen : Off
     iv. Display Resolution Dialog : Hidden By Default
     v. Use Player Log : On
     vi. Resizable Window : On
     vii. Force Single Instance : Off
10) Start getReal3D for Unity
   - Games including the getReal3D for Unity plugin will only run from the getReal3D launcher.
11) Place preconfigured configuration file in the getReal3D for Unity 3DConfigs folder.
   - This is made by getReal3D after giving computer hardware, projector and screen specifications, and layout configuration.
12) Click on configuration tab and select configuration.
13) Click on Cluster Games List and add the Unity game executable.
14) Select Left display as primary display.
15) Click Launch.

The game should appear on the CAVE display and is controllable by a USB gamepad remote. Using Xbox remote, the user navigates with the left analog stick, jumps by pressing Y and changes wand type by pressing B. If GrabbingWand, or the blue wand is selected, user presses A to grab objects. This remote is not tracked but was the only supported gaming remote available for this study.

4. MiddleVR

4.1 Overview

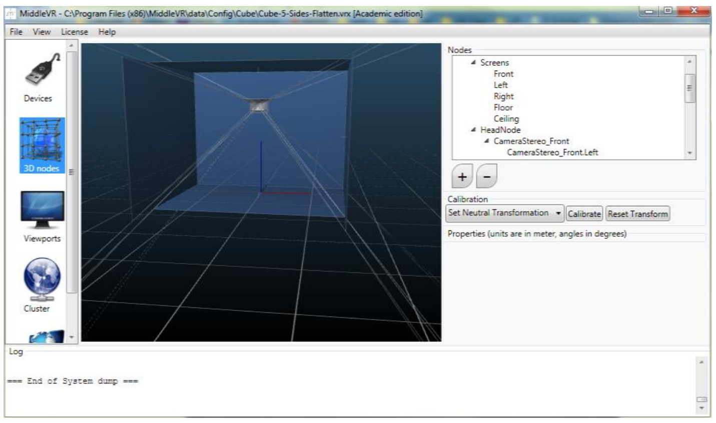
MiddleVR is VR middleware that simplifies the creation and deployment of VR applications and is adaptable to many different VR hardware and 3D applications[15]. MiddleVR supports many VR systems including CAVE, immersive Cube, Holostage, Holobench, workbench, Powerwall, Head-mounted displays, 3D TVs, and zSpace using popular interaction devices such as Kinect, Razer Hydra, and Leap. A full list of supported devices is given in *Table 2*.

Table 2: Interaction Devices supported by MiddleVR [15]**.**

Kinect (v1 & v2)
Oculus Rift (DK1 & DK2)
Leap Motion (SDK 1 & SDK 2)
TrackIR
Razer Hydra
Vuzix Tracker
SpacePoint Fusion
SpaceMouse
Intersense IS-900
A.R.T native driver
Vicon native driver
Optitrack native driver
Colibri
Organic Motion's Markerless Mocap
VRPN (Vicon, ART, Optitrack, Intersense…)
Haption's haptic devices

4.2 CAVE display

MiddleVR offers predefined configurations for 5-sided CAVEs, either with viewports displayed on one computer or with 5 computers in a cluster. *Figure 4* shows the configurator tool with a 5-sided CAVE configuration. An unlimited amount of display configurations can also be set up in the MiddleVR configurator.


MiddleVR - C:\Program Files (x86)\MiddleVR\data\Config\Cube\Cube-5-Sides-Flatten.vrx [Academic edition]
File
View
License
Help
Devices
3D nodes
Viewports
Cluster
Nodes
+
−
Calibration
Set Neutral Transformation
Calibrate
Reset Transform
Properties (units are in meter, angles in degrees)
Log

Screens
Front
Left
Right
Floor
Ceiling
HeadNode
CameraStereo_Front
CameraStereo_Front.Left

```
=== End of System dump ===
```

Figure 4: A 5 sided CAVE configuration with stereo cameras.

MiddleVR also offers predefined-configurations with set cameras, screens and viewports for several HMD devices including: Oculus Rift (DK1, DK2), NVIS-SX60, NVIS-SX111, Sony-HMZ-T1, Vuzix VR 920, and Vuzix Wrap 1200VR.

4.3 Interaction

The VRManager prefab has a VRMenu and VRWand as child objects. To use the VRWand to interact with things, the objects must have a VRActor script attached to it. Once the wand intersects with an object with this script attached, it will change color. The object is then grabbable and can be picked up by pressing a controller button and released by letting go of the button. The device manager holds the reference to all declared devices. When programming to get access to trackers' data, or the state of a joystick, initially a reference must be made of the corresponding object to the device manager [15].

4.4 Using MiddleVR in Unity

The steps to create a navigation and manipulation Unity scene in a CAVE display are shown below. After installing MiddleVR, either a Pro or Academic license is needed for output on several viewports such as a CAVE system. A 30-day trial is available to test the system with the MiddleVR toolkit. The trial version was used for this study. MiddleVR help files include a user guide, tutorial, and forum. The basic MiddleVR workflow is to create a description of the VR system in the configurator, which will output a configuration file that will configure the 3D application to match this description. The description includes all devices, trackers, physical screens and cameras used in the specific VR environment. Below is a condensed step-by-step version.

1) Import the MiddleVR package into the Unity scene
    - Package located by default at C:\Program Files (x86)\MiddleVR\data
2) Drag the VRManager prefab into the Hierarchy or directly into the scene
    - The VRManager has a VRMenu and VRWand as child objects with different options that can be configured in the Inspector.
3) Open MiddleVR configuration tool to create the desired VR system configuration.
    - For preconfigured configurations, click on the Simulations window and select from configuration list.
4) Go to the Devices window and add a device from the list. The Tracker Simulator – Mouse in the device window, shown in Figure *6a,* can be used to create a fake 3D tracker for testing purposes.
5) Go to 3D nodes window and add screens, cameras, and trackers to match the user's system. *Figure 6b* is the 3D node setup of the three sided CAVE system used for this study.
6) Go to Viewports and add the viewports ports position and resolution for the VR system. *Figure 6c* is the Viewport setup of the three sided CAVE system used for this study.
7) Save the configuration file and add the configuration file extension to the Unity editor Config File place holder in the Inspector for the VRManager.
8) Test the scene in the Unity Editor. The scene in the Unity editor can be tested, however if the viewport the user has defined in MiddleVR is different from the Unity editor in terms of aspect ratio, the view will appear distorted.
9) Build the scene in Unity. After dragging the VRManager to the Hierarchy, the player setting should set automatically, but to make sure refer to *Figure 7* for the correct settings.
10) There are two ways to run the application
    - Execute the .exe file that was created when building the game. This will use the VR configuration file that is specified in the VRManager.
    - Run the .exe file through the Simulations window in the MiddleVR configurator. This allows the user to select the VR system to use at runtime. The user can easily change the interaction controller or display configuration and run a simulation.

Getting the VRManager containing the camera, menu, and wand to appear where desired is not straight forward. Unlike other Unity objects, it does not matter where the user drags the VRManager object in the scene. A new VRManager object is created once the play button is pressed that has no relation to the actual object in the scene view. To fix this, create a GameObject UserNode and place it in the desired starting position and set this node as the VRSystemCenter in the VRManager properties. This way, when MiddleVR will create the GameObjects corresponding to the set configuration's 3D Nodes, it will use this GameObject as the VRSystemCenterNode.

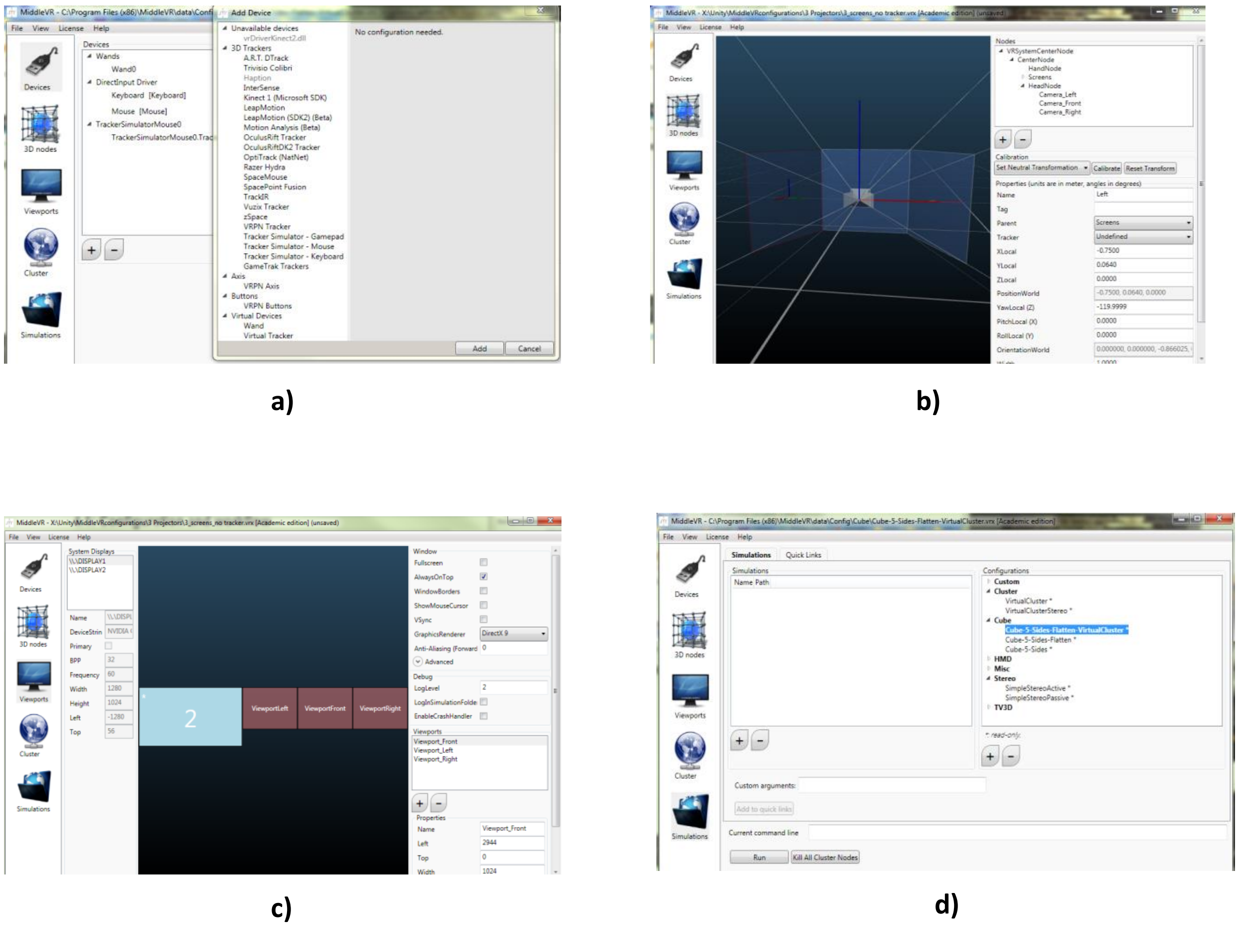
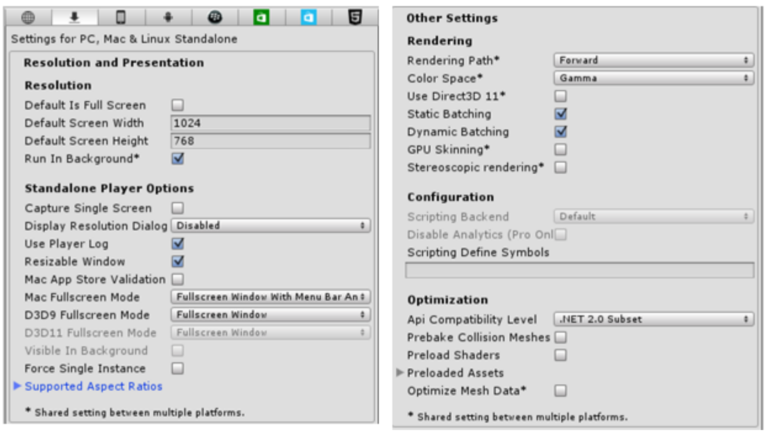


Figure 5: MiddleVR Configurator. a) Devices. b) 3D Node. c) Viewport. d) Simulation.

The different windows available in the MiddleVR configurator are shown in *Figure 6*. There is also a Cluster window where a server and clients can be added for clustered VR configurations.



Settings for PC, Mac & Linux Standalone

Resolution and Presentation

Resolution

Default Is Full Screen	☐
Default Screen Width	1024
Default Screen Height	768
Run In Background*	✓

Standalone Player Options

Capture Single Screen	☐
Display Resolution Dialog	Disabled
Use Player Log	✓
Resizable Window	✓
Mac App Store Validation	☐
Mac Fullscreen Mode	Fullscreen Window With Menu Bar An
D3D9 Fullscreen Mode	Fullscreen Window
D3D11 Fullscreen Mode	Fullscreen Window
Visible In Background	☐
Force Single Instance	☐

Supported Aspect Ratios

* Shared setting between multiple platforms.

Other Settings

Rendering

Rendering Path*	Forward
Color Space*	Gamma
Use Direct3D 11*	☐
Static Batching	✓
Dynamic Batching	✓
GPU Skinning*	☐
Stereoscopic rendering*	☐

Configuration

Scripting Backend	Default
Disable Analytics (Pro Onl	☐
Scripting Define Symbols	

Optimization

Api Compatibility Level	.NET 2.0 Subset
Prebake Collision Meshes	☐
Preload Shaders	☐
Preloaded Assets	
Optimize Mesh Data*	☐

* Shared setting between multiple platforms.

Figure 6: Player settings in Unity for MiddleVR.

Figure 7 shows the player settings for using the MiddleVR toolkit in Unity. To make sure that Unity does not override the MiddlerVR's window configuration, the default of full screen must be unchecked and display resolution dialog must be disabled. If active stereo with OpenGL Quad-Buffer is being used, then the VSync must be deactivated in the Quality settings window.

5. Reality-based User Interface System (RUIS)

5.1 Overview

Reality-based User Interface System (RUIS) is an open source toolkit for creating virtual reality and motion controlled applications, incorporating several interaction devices directly into the Unity scene [16]. These include a display manager for handling several display devices easily added within the Unity editor. RUIS supports the use of Kinect 2, Oculus Rift DK2, and PlayStation Move together in the same coordinate system[17].

5.2 CAVE display

The RUIS prefab has a display manager for a three sided cave consisting of a front, left, and right display. More displays can easily be added but unlike getReal3D and MiddleVR, no top and bottom displays can be used. *Figure 8* shows the Unity DisplayManager interface.

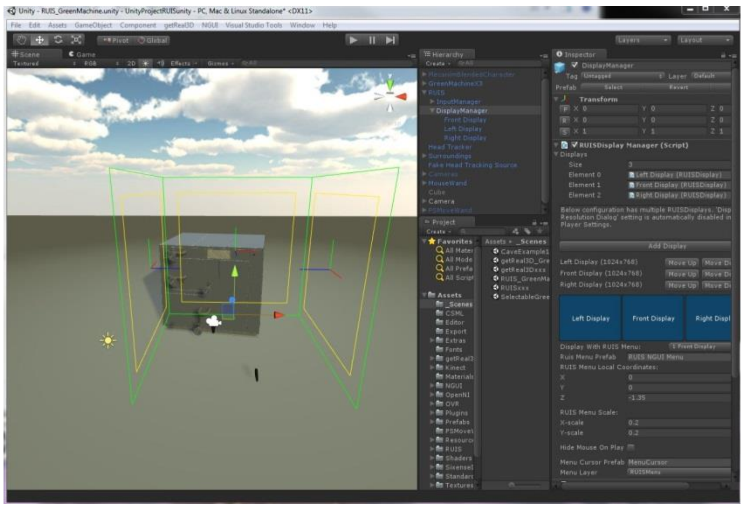


Figure 7: RUIS DisplayManager interface.

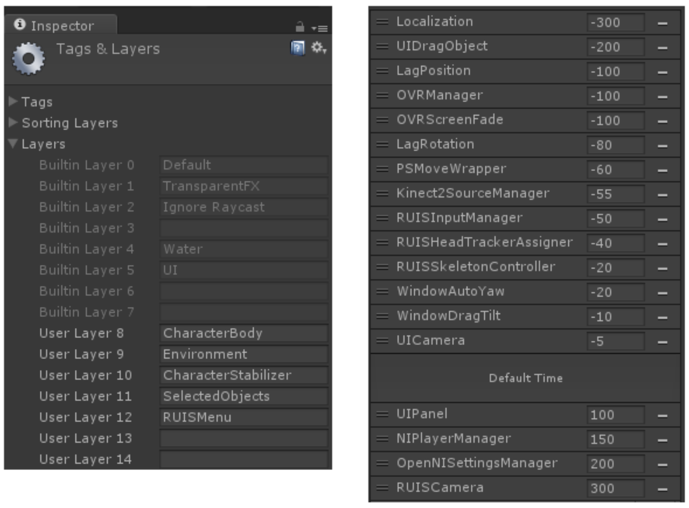
Head tracking can be achieved by selecting the desired tracker in Unity. The options are Kinect (1,2), Oculus Rift DK2, PS Move, Razer Hydra, or Input Transform. Keystone correction can be applied to projection walls while the scene is running. To access, click ESC-key to bring up the RUIS menu, and then click Display Management and drag the viewport corners. The user must temporarily disable the Head Tracker or mouse clicks will not respond when attempting to apply keystone correction.

5.3 Interaction

For interacting with the scene, RUIS offers several different wand prefabs including mouse, PSMove, RazerHydra, and Skeleton(Kinect). To interact with an object, the object must have a collider and a RUISSelectable script attached to it. Objects can be picked up and moved anywhere in the scene with the wand controller. The objects will highlight when the wand is positioned for interaction and once the main trigger button is pressed the objects will be picked up and held onto until the trigger is released. SkeletonWand is different from the other wands as it uses the selection gestures hold and fist. RUIS offers many other interaction scripts for wand interaction including ball and hinge joints. The newest version of RUIS offers avatar joint filtering and a fist gesture that can be used to grab and manipulate objects using the Kinect v2.

5.4 Using RUIS in Unity

A project folder with all RUIS scripts, example scenes, layers, etc. is available for download on the blog.ruisystem.net website. To get the scene running with the RUIS system, either create a package and import it into RUIS, or create a package in RUIS and import it into the scene and add the needed layers and script execution order. *Figure 8* shows the layers that need to be added to the Unity scene and the script execution order that will need to be set up.



Builtin Layer 0	Default
Builtin Layer 1	TransparentFX
Builtin Layer 2	Ignore Raycast
Builtin Layer 3	
Builtin Layer 4	Water
Builtin Layer 5	UI
Builtin Layer 6	
Builtin Layer 7	
User Layer 8	CharacterBody
User Layer 9	Environment
User Layer 10	CharacterStabilizer
User Layer 11	SelectedObjects
User Layer 12	RUISMenu
User Layer 13	
User Layer 14	

Localization	-300
UIDragObject	-200
LagPosition	-100
OVRManager	-100
OVRScreenFade	-100
LagRotation	-80
PSMoveWrapper	-60
Kinect2SourceManager	-55
RUISInputManager	-50
RUISHeadTrackerAssigner	-40
RUISSkeletonController	-20
WindowAutoYaw	-20
WindowDragTilt	-10
UICamera	-5
Default Time	
UIPanel	100
NIPlayerManager	150
OpenNISettingsManager	200
RUISCamera	300

Figure 8: RUIS layers and script execution order[16]

The easiest and quickest way to use the RUIS toolkit is to import the package into the RUIS project folder. The following steps assume the user is using the RUIS project folder with the scene imported.

1) Import the desired package into RUIS project folder and drag into new scene
2) Drag the RUIS prefab into Hierarchy, which includes a display and input manager.
3) Drag the RUISCamera for each display into Hierarchy.
4) Drag the desired interaction wand prefab
    a. Wand options are Mouse, PSMove, RazerHydra, and Skeleton(Kinect).
5) Click on InputManager in the Hierarchy and set the desired input device.
    a. If using PSMove the IP and Port of the PS3 must also be set. The IP need to be set in the PSMoveWrapper as well.
6) Click on DisplayManager in the Hierarchy and configure Display properties such as number of displays and corresponding position and resolution.
7) Drag HeatTrackers prefab to the scene to set up head tracking
    a. Options are Kinect(1,2), OculusDK2, PSMove, RazerHydra, MobileRazerHead, and RotationOnly.
8) Add colliders and RUISSelectable script to objects to be manipulated.
9) Test scene in Unity.
    a. By pressing play button tracking and controller can be tested in Unity. This is helpful in configuring the controller position.
10) Build executable
11) Drag executable to cover CAVE walls.
    a. The built executable will have a vertical line showing the division between screens.

6. Other toolkits

6.1 VRUI VR toolkit

This toolkit aims to support many applications that run on a range of VR environments with several input devices such as Kinect and Wiimote and run in multiple displays such as CAVE and the Oculus Rift. The toolkit offers support for multiple Kinects to facilitate a 3D video stream to other existing VRUI VR applications[18]. Although VRUI seems promising, it is not a Unity tool and is Linux-based, therefore it was not used for analysis for this study.

6.2 Unity Indie VRPN Adapter (UIVA)

UIVA is socket-based middleware to adapt VRPN to the Windows version of Unity 3D. It works with the Indie version of Unity 3D and is open source under the same license of VRPN. UIVA supports Microsoft Kinect, Nintendo WiiMote, Nintendo Wii Fit balance board, Wireless-T BPack accelerometer, General mouse, SpacePoint Fusion sensor (version 1.01), PhaseSpace optical motion capture system (version 1.02) [19].

UIVA consists of a server side and a client side. The basic concept and data circulation can be shown in *Figure 10.* The client side is a DLL file residing in Unity3D and the server side contains a VRPN client to talk to the VRPN server.

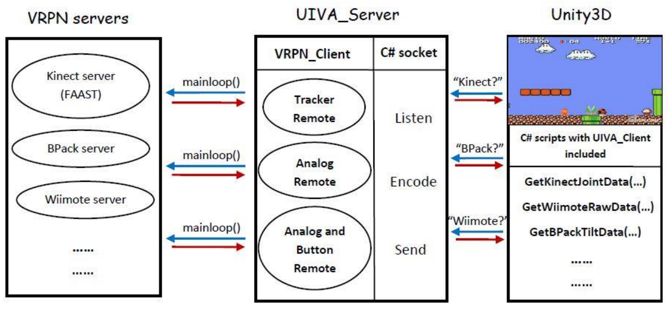


Figure 9: UIVA data circulation [20].

To pair devices to the machine the following steps are needed.

1) Start their VRPN servers
2) Setup UIVA_Server.cfg and run UIVA_Server.exe
3) Copy Uiva.dll to the Unity game Asset folder
4) Write C# scripts to request data in Update() [20]

This toolkit does not offer any prebuilt Unity package or project folder and cannot be quickly implemented into the Unity scene therefore was not analyzed in this study.

7. Comparisons and Analysis

7.1 CAVE display Comparison

Regarding configuration, MiddleVR has an external application where the viewports, nodes, cameras, trackers, etc. can be configured and the output is a configuration file that can be put in the Unity scene. With getReal3D, the configuration file is created by the support team with the user's specific VR system information. This process is timely and allows less freedom for configuring VR devices. RUIS toolkit has the quickest and fastest setup for a three-sided CAVE. However, unlike MiddleVR and getReal3D, RUIS does not allow top and bottom displays.

7.2 Interaction Comparison

In MiddleVr, to make an object interactable, a VRActor script and a collider must be added to it. The object can then be manipulated by grabbing with the wand, rotating it and setting it aside. Objects will return to their initial position after a set amount of time if the Manipulation Return Objects script is selected. This can be helpful in dissembling and reassembling complex objects such as the Green Machine shown in *Figure 11*. MiddleVR has several more built-in interaction options than the other toolkits such as the Gogo hand to reach far away objects.

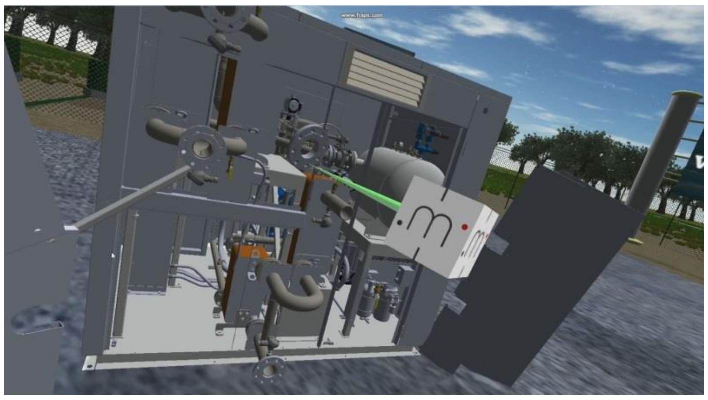


Figure 10: Dismantling the Green Machine with the MiddleVR toolkit in Unity.

In getReal3D, objects do not need any script to be manipulated. The objects need to have a rigidbody in addition to a collider attached to each object. The three given manipulation methods in the getReal3D toolkit are point, collide, and grab.

RUIS allows interaction device configuration and calibration during gameplay. It also allows the Oculus Rift, Kinect 2, and Razer Hydra or PS Move to be used simultaneously without adjusting any prior configuration files. To make an object intractable, add a collider and RUISselectable script to the object.

Table 3: Device, Interaction, and display comparison.

	Features	getReal3D	MiddleVR Academic Edition	RUIS
Common VR Game Devices	Kinect (v1,v2)	X	✓	✓
	Oculus (1,2)	X	✓	✓
	Leap Motion	X	✓	X
	Razer Hydra	X	✓	✓
	PS Move	X	X	✓
	WiiMote	✓	X	✓
Interaction	Navigation	✓	✓	✓
	Manipulation	✓	✓	✓
	Immersive Menus, GUI, Webview	X	✓	X
Displays	Viewports	Unlimited	Unlimited	3
	Stereo	Side-by-Side	Side-by-Side / Oculus	Side-by-Side, Top-and-bottom/ Oculus

The getReal3D toolkit is built on top of trackd, which supports a number of 3D trackers and controllers. Regarding common consumer VR devices like the ones shown in *Table 3*, getReal3D does not offer many 3D trackers and controllers support compared to MiddleVR and RUIS. For interaction, all toolkits offer some navigation and manipulation using a wand. MiddleVR is more extensive in giving different options on how to manipulate objects and offering an Immersive Menu. RUIS does offer a 2D menu system through the NGUI interface that is primarily for device selection and calibration. MiddleVR and GetReal3D offer clustered active stereo with OpenGL Quad-Buffer. For this, high end graphics cards are needed. In MiddleVR this is easily setup using the MiddleVR configuration tool and adding stereo cameras for each display. GetReal3D offers a getRealDisplayScreens script but no documentation on how to get active stereo working. With RUIS display manager, the application can run in any number of stereo displays when running in windowed mode, however, D3D11 Force Exclusive Mode must be disabled because that forces the application in full screen mode. With RUIS 3D displays side-by-side and top-and-bottom modes are supported.

MiddleVR offers several predefined-configurations for Oculus, NVIS, Sony and Vuzix HMDs. MiddleVR also has predefined configurations for passive stereo and 3D TVs. No user configuration interface for displays is given with the getReal3D toolkit. To get the proper display configuration the specifications for the user's system must be sent to Mechdyne and they will test and send the desired configuration file.

Taking into account multiple devices in the same system leaves a lot of room for comparison. RUIS supports the Oculus Rift, Kinect 2, and Razer-Hydra in the same coordinate system. MiddleVR only has set configurations for two devices such as the Oculus and Razer Hydra or Oculus and LeapMotion. MiddleVR does support the Kinect 2 but has no predefined configuration for using it with any other devices, such as the Oculus and Razer-Hydra. The trackd software that is used by getReal3D offers configuration files for each of the supported devices shown in *Table 1,* however there are no configuration files for using multiple devices in the same coordinate system.

RUIS multi-display setups have future plans on enabling Oculus Rift with the multi-displays when Oculus SDKs that support this are available. This will allow multiple users to view the scene while one user wears the Oculus Rift. Currently the Oculus Rift can be used for head tracking only while using multi-display setups. However RUIS did recently have added support for Unity 5 and it is unclear if they have enabled support for this.

Table 4 shows a comparison of the toolkits regarding supported OS and Unity versions, licenses and associated costs, as well as support offered.

Table 4: Software and license comparison

Features	getReal3D	MiddleVR Academic Edition	RUIS
Toolkit version	3.0.1	1.6.0.f4(Unity 4.2-4.6),1.6.1b3(Unity 5)	1.06 (Unity 4.6), 1.07 (Unity 5)
Platform	WinXP 32/64, Win 7 32,64	Win XP, Vista, 7, 8, 32/64	Win XP/Vista/7/8/ OSX
Unity version	4.3 - 4.5	4.2 and above, 5-beta	4.6, 5
Unity license	Pro	Pro for Oculus, active stereo, cluster	Pro for Oculus in 4.6, Indie for 5
Other software	DirectX11, trackd	DirectX11, MS Visual Studio 2012	OpenNI for Kinect v1, Moveme for PS Move
License type	demo / paid	Free/30 day trial/ paid	LGPL Version 3
Cost	>$3000	>$3000	Free
Support	Professional	Professional, Forum	Forum

RUIS is the most versatile when it comes to operating systems with support for WIN 8 and OSX. With no Windows 8 support and only Unity support up to version 4.5, getReal3D is the least up-to-date toolkit with respect to platform versions. A demo license is offered with getReal3D which is used for testing for this report. Currently for any change in the system a new configuration file must be sent by the getReal3D support.
For an educator one of the biggest challenges in creating a VR system is cost [21]. It is unclear what the cost of getReal3D will be but it seems to be competitive with other academic licenses. MiddleVR offers a free version and a 30 day trial of HMD, Academic, or Professional versions. RUIS is distributed under the LGPL Version 3 license. The most attractive part about RUIS when compared to the other toolkits is that it is free.
Regarding support most requests made to MiddleVR are answered within a few hours. Sometimes they are answered the next day but this is probably due to the time difference as the headquarters is in Paris, France. Professional support is available for getReal3D with most email responses within a day or two. RUIS has an active forum where questions can be submitted and are commonly answered the same day.

8. Rubric Comparison

In order to qualitatively compare the toolkits, a rubric system was created based on a software evaluation rubric presented in The 21st-Century Classroom: Teaching and Learning with Technology [22]. *Table 5* shows a rubric system concerning the documentation and support provided for each toolkit. This involves analyzing the documentation provided, technical support, help options, tutorials and examples given. These attributes can help gauge the initial learning curve required to use the software and support available to achieve proficiency. This analysis was made using demo and trial versions of the software, and is a record of this studies initial experience.

Table 5: Toolkit documentation and support evaluation rubric system.

	Evaluation Criteria							
Software Feature	***1 Poor***	***2 Below Average***	***3 Average***	***4 Above Average***	***5 Excellent***	**getReal3D**	**MiddleVR**	**RUIS**
Document-ation	Document-ation is excessively technical and/or difficult to follow	Document-ation is generally understand-able but not very user-friendly	Document-ation is user-friendly and reasonably easy to follow	Clear document-ation that is logical and easy to follow	Very clear, user-friendly document-ation that leaves no questions	3	4	3
Technical support	No support available	Online forum support only or emails with long response time	Online forum and emails answered in a reasonable amount of time	Online forum and emails quickly answered	Online forum, email and 24-7 real time chat or phone support.	2	4	2

Help features	Few or no help features available	Help limited to a Help or Read-Me file	User guide, forum	User guide, forum, trouble-shooting guide	Extensive user guide, reference material, trouble-shooting guide, support site, forum	2	5	3
Tutorials	No tutorials provided	One tutorial provided with no documentation	Limited tutorial and/or documents provided	Several tutorials provided	Extensive tutorials provided	1	4	1
Examples	No example provided	One example provided with no document-ation	Limited example and/or documents provided	Several examples covering some applica-tions	Extensive examples provided covering many applica-tions	2	3	5
					Total	10	20	14

RUIS provides a 22 page readme file with requirements, known issues, and installation procedures, and has a website with posts for updates in the toolkit and a forum for troubleshooting. getReal3D comes with an 11 page user guide and a 26 page developer guide outlining the basics from installation to scripting. The developer guide includes a section on enabling VR capabilities such as interaction and navigation. The guide also provides information on scripts and prefabs given with the tool. MiddleVR includes an extensive 246 page user guide with tutorials, concepts, configurations, scripting examples and much more. The guide is written for every level of Unity and programming proficiency from beginner to expert. There are sections on advanced programming, haptics, stereoscopy, VRPN server, and a troubleshooting section.

Technical support is available through an online forum for RUIS and MiddleVR but not getReal3D. No other support is available for RUIS, but getReal3D offers professional technical services 24 hrs a day via Pivot with a Mechdyne service contract. MiddleVR offers paid professional support services as well as forum and email for technical assistance. There is also a knowledge base with a troubleshooting guide along with tips and tricks. MiddleVR offers video tutorials on a wide range of topics and documented tutorials with step-by-step logic to get the user quickly started on a working VR application. RUIS is the only toolkit that provided several example scenes inside of Unity to quickly test several VR devices in the RUIS testbed scenes. MiddleVR did provide an example Shadow executable but no project folder that can be edited inside of Unity. MiddleVR significantly outperformed the other toolkits regarding the documentation and support. Along with having extensive help documentation and video tutorials, emails were answered promptly.

A further analysis of the toolkit comparison is shown in *Table 6* using another rubric system that was created to analyze the toolkits use in VR applications involving CAVE display and interaction. Performance, flexibility and ease of use are figures of merit used for comparison, which have been stated as three primary requirements for a VR development system [5].

Table 6: Toolkit VR application figures of merit rubric system.

	Evaluation Criteria							
Software Feature	***1 Poor***	***2 Below Average***	***3 Average***	***4 Above Average***	***5 Excellent***	**getReal3D**	**MiddleVR**	**RUIS**
Perform-ance & Reliability	Not able to run application or always crashes	Slow, crashes frequently; other technical issues	Performs ok, crashes sporadically, some technical issues	Loads and performs quite fast; minor technical issues	Loads and performs quickly; reliable	2	5	4
CAVE display Flexibility	Very restricting requirements and limited hardware	Limited configure-ations and few supported VR devices	Set number of configure-ations and some support VR devices	Several configure-ation options and many supported VR devices	Quickly adapts to several supported devices and configure-ations	2	4	3
Inter-action Flexibility	Very restricting require-ments and limited VR devices	Limited configure-ations and few supported VR devices	Set number of configure-ations and some support VR devices	Several configure-ation options and many supported VR devices	Quickly adapts to several supported devices and configure-ations	2	4	5
Ease of use	Steep learning curve and support needed.	Extensive time needed to gain proficiency	Some time needed to gain proficiency	Software is easy to follow with minimal learning curve	Software is intuitive with no learning curve required	4	3	3
VR Applica-tions	Lacks some of the commonly included basic features	Only basic features are included	Basic features and additional features are included	Includes most of the features desired	Comprehen-sive features included	2	5	4
					Total	12	21	19

The CAVE display performance varied with the toolkits as the way the display appeared on projectors differed. MiddleVR can either be executed in the configurator or the Unity built executable and will appear clustered on each projected display. The display was consistent and had no performance issues. There were issues with Unity crashing and objects not displaying correctly when using the getReal3D toolkit. With getReal3D, the display system is configured specifically for the user's current VR system taking into account the resolution, display configuration, and computer hardware. It is unclear what all the changes are made to the configuration file but it took three weeks to get the file and it was not consistently working and very unstable. When executing an RUIS built scene for CAVE display, the output is one stretched window that must be dragged to the desired location. The performance of interaction with MiddleVR far exceeded RUIS using the test bed scene with the Razer Hydra. It was much smoother and precise and allowed object manipulation while picking up, such as rotation and position return. It should be noted that the new version of RUIS with Unity 5 support, released April 23, 2015, offers more precise interaction than previous versions.

For CAVE flexibility, MiddleVR was the preferred system, with the display configurator giving the user unlimited options for screens, cameras, and tracking nodes. The cluster window allows server client configurations for connecting several computers. With getReal3D, several systems can be configured such as a 6-sided CAVE or CAVE 2. RUIS does not offer a bottom and top CAVE display.

The main focus of the getReal3D toolkit is not interaction and gaming, rather it is to display clustered active stereo in multiple different CAVE configurations. There are few wand configuration options and only the gamepad device is supported on the demo version. Therefore, getReal3D is given the lowest score for interaction flexibility compared to the other toolkits. However, it may have advantages in other scenarios.

For flexibility, RUIS offers several common interaction devices and the corresponding scripts for using them. RUIS also provides scenes with intractable devices such as hinges and levers with associated scripts. Several devices can easily be added together with tracking and CAVE display, thus giving a versatile interaction experience. MiddleVR offers a long list of interaction devices that can easily be added together with corresponding tracking nodes for head and hand. However MiddleVR does not offer much interaction or gesture recognition for the Kinect 2.

The getReal3D system had the smallest learning curve for importing into the Unity scene and executed in the RUIS configurator. The MiddleVR configurator is more complex but allows many more options than getReal3D. For some applications, RUIS is the easiest for quick CAVE display. However, RUIS distributes a project folder whereas getReal3D and MiddleVR distribute a Unity package that can be easily imported into the scene. To get the scene running with the RUIS system, the user can either create a package and import into RUIS or create a package in RUIS and import into the scene and add the needed layers and script execution order. Either way, it will take longer to implement. If the scene desired can easily be exported into a package and imported into the RUIS project folder, then this will not be an issue. CAVE active stereo was achieved using MiddleVR toolkit and documentation provided. Although this is a feature of the getReal3D and RUIS toolkits, due to lack of relevant help documentation, active stereo was not achieved for this study.

Regarding ease of use, MiddleVR is exceptional as one Unity-built executable can be adaptable to various combinations of interaction devices using the configurator. For RUIS, the interaction wands are given as prefabs with abilities of enabling and calibrating with an on screen menu while running the executable. This allows the user to test and calibrate how different devices work together without having rebuilt the scene or even open a different executable. However, each prefab must be brought into the scene and positioned correctly.

Along with navigations and manipulations, MiddleVr is the only toolkit providing immersive menus, immersive webview, and immersive custom GUIs using HTML5. This gives MiddleVR the clear advantage amongst the other toolkits concerning VR applications. RUIS offers several examples using multiple interaction devices together simultaneously. With the given prefabs the Oculus Rift, PS Move and Kinect v2 can be used together which opens up the door to many VR applications.

Overall MiddleVR only slightly outperformed RUIS in the toolkit applications figures of merit. getReal3D lagged behind the other toolkits with few configurable options and limited interaction abilities, but game and consumer devices are not the main purpose.

9. Conclusion

In this paper, a qualitative study has been conducted analyzing several toolkits available to bring VR applications to a Unity scene. By focusing on application development for a three-sided cave using wand interaction, three toolkits available were found to meet these criteria. The three toolkits analyzed all have some features and interaction options that are superior for a specific application. MiddleVR significantly outperformed the other toolkits regarding the documentation and support available. Although RUIS does offer some attractive features, MiddleVR was the most versatile toolkit analyzed regarding CAVE display and interaction. With the added bonus of immersive menus, webview, and custom GUIs, MiddleVR strongly outperforms the other toolkits in this study. getReal3D may be better suited than MiddleVR for some specific CAVE applications and training scenarios, such as CAVE 2, but due to lack of consumer device and interaction modules it scored lowest in this study. MiddleVR only slightly outperformed RUIS and with RUIS being free and highly versatile, it is promising for low budget applications.

10. References

[1] M. O. Onyesolu, I. Ezeani, and O. R. Okonkwo, "A Survey of Some Virtual Reality Tools and Resources," *Virtual Real. Environ.*, pp. 21–42, 2012.

[2] W. R. Sherman and A. B. Craig, *Understanding Virtual Reality*. San Francisco: Morgan Kaufmann Publishers, 2003.

[3] B. Shneiderman and C. Plaisant, *Designing the user interface: strategies for effective human-computer interaction*, vol. 215, no. 7. 2005.

[4] Unity Technologies, "The leading global game industry software," 2015. [Online]. Available: https://unity3d.com/public-relations. [Accessed: 25-Jun-2015].

[5] A. Bierbaum and C. Just, "Software tools for virtual reality application development," 1998.

[6] S. Bangay, "A Comparison of Virtual Reality Platforms," Grahamstown, 1994.

[7] J. Kjeldskov and J. Stage, "Interaction styles in tools for developing virtual environments," *Virtual Real.*, vol. 12, no. 3, pp. 137–150, 2008.

[8] M. Fiorentino, G. Monno, and a E. Uva, "Smart Tools For Virtual Reality Based Cad Related Work," *Assoc. Naz. Disegno di Macch.*, 2004.

[9] J. O. Kim, M. Kim, and K. H. Yoo, "Real-time hand gesture-based interaction with objects in 3D virtual environments," *Int. J. Multimed. Ubiquitous Eng.*, vol. 8, no. 6, pp. 339–348, 2013.

[10] A. Olwal and S. Feiner, "Unit — A Modular Framework for Interaction Technique Design , Development and Implementation .," Columbia University, 2002.

[11] Mechdyne Corporation, "getReal3D for Unity3D User Guide," Marshalltown, 2015.

[12] Mechdyne Corporation, "getReal3D for Unity3D Developer Guide," Marshalltown, 2015.

[13] A. Nishimoto, "Guide for running Unity in CAVE2," *GitHub*, 2015. [Online]. Available: https://github.com/arthurnishimoto/omicron-unity/wiki/Guide-for-running-Unity-in-CAVE2.

[14] Mechdyne Corporation, "Trackd User's Guide," Virginia Beach, 2011.

[15] S. Kuntz, "MiddleVR User Guide," Paris, 2015.

[16] T. Takala and M. Matveinen, "RUIS for Unity 1.07," Helsinki, 2015.

[17] T. Takala and R. Pugliese, "Reality-based User Interface System (RUIS)," *WordPress*, 2015. [Online]. Available: http://ruisystem.net/. [Accessed: 06-May-2015].

[18] O. Kreylos, "Vrui VR Toolkit," 2015. [Online]. Available: http://idav.ucdavis.edu/~okreylos/ResDev/Vrui/index.html.

[19] J. Wang and R. W. Lindeman, "Unity Indie VRPN Adapter (UIVA)," *Worcester Polytechnic Institute*, 2014. [Online]. Available: http://web.cs.wpi.edu/~gogo/hive/UIVA/. [Accessed: 09-May-2015].

[20] J. Wang, "Unity Indie VRPN Adapter," 2014.

[21] K. Hanson and B. E. Shelton, "Design and Development of Virtual Reality : Analysis of Challenges Faced by Educators," *Educ. Technol. Soc.*, vol. 11, pp. 118–131, 2008.

[22] J. Lever-Duffy, J. McDonald, and A. Mizell, *The 21st-Century Classroom: Teaching and Learning with Technology*. Addison-Wesley Longman Publishing Co., Inc., 2002.