OpenSimulator Interoperability with DRDC Simulation Tools

Compatibility Study

Mark Swartz
Evan Harris
CAE Inc. - Integrated Enterprise Solutions

Prepared By:
CAE Inc. - Integrated Enterprise Solutions
1135 Innovation Drive
Ottawa ON K2K 3G7

Contractor's Document Number: 5457-001 Version 01
Contract Project Manager: Damon Gamble, 613-247-0342
PWGSC Contract Number: W7707-135643
CSA: Mark G Hazen, Lead Maritime Command Team Support Group, 902-426-3100 x176

Contract Report
DRDC-RDDC-2014-C222
September 2014

Principal Author

Original signed by Mark Swartz

Mark Swartz

M&S Consultant, CAE IES

Approved by

Original signed by Damon Gamble

Damon Gamble

Project Manager, CAE IES

Approved for release by

Original signed by Leon Cheng

Leon Cheng

Chair, Document Review Panel

Abstract

This compatibility study examines potential interoperability between OpenSimulator, a free, server-based 3D virtual world simulator, and a set of DND/CF simulation tools. Interoperability is assessed in four areas: simulation operation, terrain models, 3D models, and human factors tools. Simulation technologies assessed for compatibility are the DIS and HLA simulation protocols, the Unity game engine, the VBS2 serious game, SmartMarine 3D and SmartPlant 3D CAD applications, the HumanCAD anthropomorphic articulation simulator, and the IPME task modelling and crew simulator. OpenSimulator does not use DIS or HLA, so it is not currently able to directly interface with any simulator that uses these industry standards. Terrain data compatibility is limited by the RAW format used by OpenSimulator. Unity is able to provide the required height-map portion of the terrain while VBS2 cannot produce a compatible terrain component directly. The set of DND/CF tools that use model-based assets can, in most cases, be converted to the COLLADA format which OpenSimulator is compatible with. HumanCAD also has the ability to export model formats that can be converted to COLLADA. IPME can communicate over TCP/IP connections and so an interface to OpenSimulator could be developed. Overall, OpenSimulator is compatible with the set of DND/CF tools examined in terms of its ability to re-use 3D model content originally developed for these tools. Some compatibility exists with respect to re-use of terrain data. But there is marginal to no compatibility with respect to operating OpenSimulator using the tools studied.

This page intentionally left blank.

Executive summary

OpenSimulator Interoperability with DRDC Simulation Tools: Interoperability with DRDC Simulation Tools

Mark Swartz; Dr. Evan Harris; September 2014.

Introduction: This report was written as part of the Virtual World (vWorld) Capability Development Support project by CAE Integrated Enterprise Solutions (IES) on behalf of Defence Research and Development Canada (DRDC). CAE investigated the compatibility of OpenSimulator in terms of its interoperability with other DND/CF simulation technologies.

Interoperability was assessed in four areas: simulation operation, terrain models, 3D models, and human factors tools. Additionally, four DRDC use cases were used to provide context within which OpenSimulator compatibility would be applied. These use cases are the ability to (1) import previously developed content, (2) export content created in OpenSimulator, (3) integrate an OpenSimulator simulation into a larger simulation, and (4) enable distributed simulation environments.

OpenSimulator is an open source server application that can be used to create a 3D distributed virtual world that can be accessed by multiple users from disparate geographical regions for collaboration and interaction activities. The OpenSimulator virtual world is accessed via a client application known as a "viewer". The DND/CF simulation technologies that were investigated included the DIS and HLA distributed simulation protocols, Unity game engine, VBS2 combat simulator, SmartMarine 3D and SmartPlant 3D CAD applications, and the human factors tools HumanCAD anthropometric modelling environment, and IPME discrete event simulator.

Results: Regarding the interoperability of simulation operation by using DND/CF simulation tools to operate, control, or perform a simulation using OpenSimulator, there exists effectively no direct compatibility. OpenSimulator does not currently support DIS or HLA protocols for distributed simulation, although it may be possible to develop this capability. While several efforts to construct a Unity-based viewer have been made, no Unity-based clients are available. The use of VBS2's API to interface with OpenSimulator was discussed, but it was concluded that there is likely no advantage to developing a VBS2 viewer to OpenSimulator. However, given that OpenSimulator enables communication between multiple users via Linden Labs protocol (using HTTP and UDP), interoperability with DND/CF tools is not required in order to execute a distributed simulation compatible with DRDC use case (4). A number of freely available compatible client applications (such as the Firestorm viewer) can be used to connect to OpenSimulator over the Internet.

The investigation of the interoperability of terrain models developed by external applications with OpenSimulator revealed that terrain models developed in Unity were compatible while VBS2 terrain models could not be directly converted by VBS2 to a compatible format.

It was found that OpenSimulator is directly compatible with 3D model files as long as they are in COLLADA format. For models that are not in this format, this report demonstrates how interoperability can be achieved by using a range of free and commercial tools for conversion.

These tools include Blender, AutoCAD, 3DS Max and SketchUp. Thus, DRDC use case (1) was demonstrated as feasible. However, regarding DRDC use case (2) for exporting content from OpenSimulator, it was determined to be infeasible without significant development effort since OpenSimulator cannot export models out into formats that can be used by the set of DND/CF simulation tools.

The interoperability between OpenSimulator and human factors tools HumanCAD and IPME proposed that model re-use of the human anthropometric mannequins could be enabled via Blender-compatible formats where the motion kinetics could potentially be incorporated and the new model subsequently exported from Blender to the OpenSimulator compatible format. OpenSimulator could potentially be integrated with IPME through the use of Region modules, making OpenSimulator a part of a larger simulation as per DRDC use case (3). However, without further investigation and experimentation, the technical feasibility of this approach is uncertain.

Significance: This report demonstrates where compatibility exists, in terms of the interoperability of OpenSimulator with several DND/CF tools exists. It also highlights where compatibility is absent, or where it is uncertain or unknown, and identifies areas for further study and investigation. The compatibility checklist it presents provides a useful tool to guide DRDC in the assessment of OpenSimulator compatibility with additional tools not reviewed in this report.

Future plans: This report presents some detailed analysis of OpenSimulator compatibility and often demonstrates the level of interoperability through step by step examples. However, where scope constraints required, a higher level of assessment was provided, leading in several instances to potential areas where further and more detailed investigation and experimentation would be required to definitively conclude the level of compatibility. Depending on DRDC priorities, these areas could provide avenues for future investigation or even development of technology to enable compatibility where currently there is none. Specifically, further investigation could be applied to the development of a DIS or HLA interface for OpenSimulator, development of content export capability from OpenSimulator, and demonstration of model and/or terrain export from VBS2 to OpenSimulator.

Table of contents

List of figures

List of tables

Acknowledgements

Special thanks to following individuals for their advice and input into this study:

Christopher Cooper, CAE IES Canada

Matthew Keown, Allen Vanguard

Tab Lamoureux, CAE IES Canada

Mike Lepard, CAE IES Canada

Tom Miller, CAE IES Canada

Chris Walters, CAE Professional Services Australia

This page intentionally left blank.

1 Introduction

This report is written as part of the Virtual World (vWorld) Capability Development Support project; contract number W7707-135643/001/HAL. The contractor, CAE Integrated Enterprise Solutions (IES), on behalf Defence Research and Development Canada (DRDC), has investigated how OpenSimulator can be used to provide an interactive 3D virtual environment to facilitate collaborative activities between disparate DRDC team members.

The potential DRDC use cases of virtual world simulation technology provide opportunities for cost savings through re-use of virtual world assets and content previously developed and by providing the capability to distribute simulation activities across computer systems and between disparate personnel. These use cases can be categorized as follows:

1. Import of content: re-use of content developed for one simulation application in another. This would require the ability to import to OpenSimulator virtual content developed by external model development tools, possibly for other simulation applications.
    - Example: Conduct an After Action Review in OpenSimulator using a ship model reused from a navigation trainer simulation.
2. Export of content: re-use of content developed for a specific vWorld instance/application into another instance/application. This might require the ability to transfer content between two different OpenSimulator applications or projects, or it may require export of content developed in OpenSimulator to external software applications (other simulators or CAD environments).
    - Example: Re-use of compartment layouts developed as part of the design process within OpenSimulator to develop operation procedures.
3. Simulation integration: re-use of a virtual world developed for one application by integrating it into another, perhaps larger, simulation.
    - Example: Integrate a virtual world developed in OpenSimulator into a broader distributed simulation framework (perhaps using DIS/HLA) or add external stimulation engagement models by interfacing with IPME to run a discrete event simulation.
4. Distributed simulation: provide the capability for distributed human engagement in exercises and experiments.
    - Example: Render a virtual ship layout or operations centre where disparate personnel can interact and execute objectives to complete a common set of tasks.

With the above use cases providing the context in which OpenSimulator could be employed, CAE IES performed a compatibility study between OpenSimulator and several DND/CF simulation technologies. Compatibility was assessed based on the level of interoperability between OpenSimulator and these technologies. This report presents the results of the compatibility study.

1.1 Overview of OpenSimulator vWorld Technology

OpenSimulator is an open source server application that can be used to create 3D virtual worlds that can be accessed, potentially, from any point in the world using a variety of clients. OpenSimulator is written in C# and can operate on Windows using the .NET Framework or on

Unix-like machines using the Mono Framework. The source code is released under a BSD License.

The OpenSimulator server is divided into two components: (1) backend data services consisting of user accounts, login service, assets, and inventory; and (2) the simulator server which can host numerous virtual environments called *regions*. Two modes are available: *standalone* mode and *grid* mode. In standalone mode, these two components are combined into a single OpenSimulator process. In grid mode, the two components are separated, placing the backend services into a ROBUST process (Redesigned OpenSimulator Basic Universal Server Technology) and the simulation servers into OpenSimulator processes. This allows scaling of services to support user growth since the ROBUST services and simulation servers can run on completely separate, dedicated machines if required. Multiple instances of the OpenSimulator process can be run on different machines in order to distribute processing load. For practical applications consisting of many simultaneous users, most implementations of OpenSimulator will operate in grid mode.

The term *grid* is derived from the nature of the 3D virtual world created by the OpenSimulator server. The world is comprised of a virtual grid of *regions* located by a unique pair of (x, y)-coordinates. Each region is 256×256 m^2 and is hosted by a simulation server. One simulation server can host multiple regions. Multiple simulation servers can be connected forming grids (potentially) as large as $65{,}536^2$ regions. The simulation servers may also be distributed over a large geographic area, providing the ability to form private network-based grids over a LAN or WAN, or public grids comprised of simulation servers that host regions worldwide and which can be accessed from the internet. From an avatar's perspective, however, these regions are all part of the same grid or virtual environment, and therefore, two adjacent regions in the virtual world hosted by two different servers in the real world that are separated by hundreds of kilometers can be easily traversed by simply walking the avatar across the region boundary. Grid-to-grid connections can also be established via a *hypergrid* which is comprised of a network of grids that are linked via portals which avatars can use to teleport to alternate grids.

1.2 Compatibility Assessment Areas

In this report, the compatibility of OpenSimulator with several other simulation technologies used by DND/CF is investigated. The term *compatibility* is defined as the level of interoperability, or how well OpenSimulator works with these other simulation technologies. Specifically, this report examines interoperability with respect to four areas: simulation operation, terrain data, 3D models, and human factors tools.

Any limitations imposed by OpenSimulator as a result of how it has been designed to operate or work within these areas will directly affect the level of interoperability that can be achieved with alternate simulation technology. It is therefore important to understand how OpenSimulator has been designed to operate and interface with users, how it incorporates terrain models, and what its capabilities are with regard to modelling other content and virtual assets. These facets are discussed in the three following paragraphs.

Simulation operation:

1. OpenSimulator is a client-server application that uses HTTP- and UDP-based messages to communicate between ROBUST services and the various simulation

servers, between ROBUST services and the client, and directly between the simulation servers and the client. OpenSimulator requires a database to store information persistent both within particular regions (terrain configuration, terrain assets, buildings, etc.) and also grid-wide across regions (user account information, user assets, avatar configuration). In standalone mode, SQLite is the default database. In grid mode, MySQL is currently the only fully supported database, although most OpenSimulator features are also supported by Microsoft SQL Server. The clients that connect to the simulator are called *viewers*. Viewers allow the user to see the virtual world, move within it, and alter it by making changes to the terrain and environment, virtual objects (structures, vegetation, etc.), and avatar appearance.

Terrain models:

2. OpenSimulator terrains are modelled using a RAW format which consists of 13-layers of information including a height-map, provided as a gray-scale image, in the first layer. Terrain models are tiled to provide coverage for a single 256 × 256 m^2 region or can be created to cover multiple adjacent regions. Individual textures are applied separately via the viewer, allowing the owner of a region to customize its terrain with multiple textures that take effect depending on the elevation of a specific point. Terrain data is stored in the database.

3D models and meshes:

3. All in-world objects, or assets, are based on some type of 3D model or mesh. These include the avatar, its shape, clothes, and accessories; the virtual objects such as buildings, structures, and vehicles; and terrain culture such as trees and plants. OpenSimulator supports mesh objects and 3D models in COLLADA format; however, since not all viewers support mesh, the ability to import and/or view these models depends on the client used to connect to the grid. This report recommends the use of the Firestorm viewer since it supports both import and view of mesh objects (see Table 2 in section 4). Most viewers also provide the ability to construct simple primitive objects ("prims") directly in-world, such as cubes and spheres, and include a limited set of prim re-shaping operations to customize them. Importing sculpted prims ("sculpties"), which are prims whose shapes are determined by their texture, is also possible.

1.3 DND/CF Simulation Technology

This report describes the result of investigating the compatibility of OpenSimulator with the simulation technologies used by DND/CF. These simulation technologies are discussed in the following sections.

1.3.1 HLA & DIS

High-Level Architecture (HLA), or IEEE Standard 1516, and Distributed Interactive Simulation (DIS), or IEEE Standard 1278, are standards that facilitate interoperability of distributed simulations regardless of computing platform. Each standard defines an architecture (HLA) or communication protocol (DIS) for the transfer of data and synchronization of actions across

distributed platforms connected to a single real-time simulation environment. HLA and DIS are widely used within Defence for the purpose of conducting war-gaming, real-time strategy, and training simulations.

1.3.2 Unity

Unity is a cross-platform game engine and integrated development environment targeting 3D interactive content. Unity supports desktop computers, web plugins, video game consoles and mobile devices. Potential points of compatibility between Unity and OpenSimulator include: a Unity-based desktop computer OpenSimulator viewer; a Unity-based web-plugin OpenSimulator viewer; the sharing or converting of regions developed in one platform to the other; and the sharing or converting of 3D assets developed in one platform to the other. Unity version 4.1 was used for the purposes of this compatibility study.

1.3.3 VBS2

Virtual Battlespace 2 (VBS2) is used to create a virtual world focused primarily on land combat situations. DND is in possession of an enterprise license for VBS2 from Bohemia Interactive, which provides broad use of the application suite across the entire department, including contractors executing work for DND. Notwithstanding that VBS2 is primarily focused on land combat situations, VBS2 does come with an extensive development kit and the ability to modify and extend the capability inherent in the product. As such there is always potential for engineers and developers to design and implement extensions to VBS2 that would meet the needs of the vWorld efforts.

1.3.4 SmartMarine 3D and SmartPlant 3D

Intergraph's SmartMarine 3D for ship design, and SmartPlant 3D for plant design, are two CAD programs used or intended to be used by DND. The 3D CAD models created by these programs could potentially be imported into the OpenSimulator vWorld. To this end, this report investigates the interoperability of the 3D formats used by these software programs with OpenSimulator.

1.3.5 HumanCAD

HumanCAD allows a 3-D model of a virtual environment (e.g., an operations room on a ship) to be developed in order facilitate both visualization and the performance of anthropometric assessments. A series of mannequins to represent the effective physical range of the anticipated user population (e.g., 5th percentile female to 95th percentile male) are available within HumanCAD. These mannequins can be employed within the virtual environment to assess a series of reach, vision and clearance tests, corresponding to the task-based scenario in order to evaluate a proposed layout. These tests could be used to determine if the anticipated user population could perform the defined tasks that are under consideration. Demonstrating compatibility between HumanCAD and OpenSimulator could allow anthropometric mannequins to be employed within open source virtual worlds for the purposes of conducting HF-based assessments.

1.3.6 IPME

Integrated Performance Modeling Environment (IPME) is a task network simulation environment created to model human work. IPME's underlying architecture is a Discrete-Event Simulation (DES). A DES is the simulation of the changes of states in a system. A DES is thus either a process-oriented, event-oriented, or activity-oriented simulation tool. Tasks are described in IPME according to who performs them and the load the task imposes in terms of visual effort, auditory effort, cognitive effort and psychomotor effort. These dimensions can be affected by environmental conditions, and can interfere with each other. In addition to the description of task demands imposed on operators in a workflow, an operator's workload capabilities represent the amount of time a resource (split between the broad categories of input, central, and output resources) is used during a task. Further, tasks themselves can be given priority weightings, leading to tasks being interrupted or shed altogether at times of significant demand. IPME has been shown to be an effective method of representing human performance for the purposes of system evaluation and design. Task network models developed in IPME could be used to stimulate avatars within OpenSimulator virtual worlds with the added benefit of being able to assess human performance through IPME's workload algorithms.

1.3.7 Supplementary Software Tools

Additional software is referred to throughout this report that has been found to provide supplementary support for the creation of content and resources compatible with OpenSimulator. These include programs such as Blender, GIMP, AutoCAD, 3DS MAX, and SketchUp. They are discussed in context within the relevant sections of this report.

1.4 Report Organization

The remainder of this report breaks down the compatibility study into four main compatibility assessments between OpenSimulator and each relevant DND/CF simulation tool as follows:

1. Interoperability of Simulator Operation
2. Interoperability of Terrain Models
3. Interoperability of 3D Models
4. Interoperability with Human Factors Tools

Each assessment examines communication protocols, and/or file formats that would allow DND/CF personnel to leverage existing vWorld infrastructure (GUIs, 3D models, environments, terrain) created using familiar simulation tools, and incorporate them into OpenSimulator, all in the context of the DRDC use cases. The final section presents the conclusions of the compatibility study.

2 Interoperability of Simulation Operation

The interoperability between OpenSimulator and the set of DND/CF simulation tools with regards to operating, controlling, and/or performing a simulation is discussed in this section. Specifically, this section discusses the possibility of communication between simulations run within the different simulation environments, and also identifies if certain DND/CF tools could be used to build platforms for direct operation of OpenSimulator. This discussion begins with an overview OpenSimulator communication protocol.

2.1 OpenSimulator Clients

OpenSimulator's server-client architecture provides a distributed simulation architecture that connects multiple users located at different geographic locations to a common virtual environment where they can interact. As such, OpenSimulator enables DRDC use case (4), that of providing a distributed simulation, by allowing disparate DRDC personnel to connect from anywhere with internet access.

OpenSimulator uses the Linden Lab viewer protocol: a combination of HTTP and UDP communication protocols. Communication occurs initially between the client (known as the *viewer*) and ROBUST services when a user first logs in to the vWorld grid. ROBUST informs the simulation (region server) to expect the arrival of the user's avatar, while simultaneously providing the address of the simulation back to the client. The client and simulation then communicate directly to handle in-region tasks (object updates, avatar position updates, etc.) while grid-wide tasks (map services, teleports, etc.) are handled via communication between ROBUST, the client and simulator simultaneously [1].

A number of viewers have been developed which are compatible with OpenSimulator and Linden Labs Second Life virtual world. A viewer typically allows the user to roam the virtual environment via walking, running, or flying. Also, depending on the permissions set by the region owner(s), the viewer allows the user to modify the terrain, add, edit, or remove virtual objects that populate the world, as well as customize the avatar. Table 1 lists a subset of available viewers. A comprehensive list is available from the OpenSimulator website [2]. The selection of a particular viewer over another will depend on the capabilities of each respective viewer. In section 4, the Firestorm viewer is recommended in part based on its ability to import and/or view mesh objects. Note that some viewers are no longer being actively developed. In some cases, activity by its developers has been transferred to another viewer which is intended to replace the original one. If this is a case with a viewer listed in the table below, it is noted in the active column by indicating if the viewer has been replaced and what viewer has replaced it.

Table 1: OpenSimulator Viewers

Viewer	Developer	Website	Active
Cool VL	Henri Beauchamp	http://sldev.free.fr/	Yes
Firestorm	The Phoenix Firestorm Project, Inc	http://www.firestormviewer.org/	Yes - replaced Phoenix
Hippo	Metropolis	http://www.hypergrid.org/metropolis/wiki/en/	Yes
Imprudence	Imprudence	http://wiki.kokuaviewer.org/wiki/Main_Page	No - replaced by Kokua
Kokua	Imprudence	http://wiki.kokuaviewer.org/wiki/Main_Page	Yes - replaced Imprudence
Phoenix	The Phoenix Firestorm Project, Inc	http://www.firestormviewer.org/	No - replaced by Firestorm
Singularity	Singularity	https://sites.google.com/site/singularityviewer/	Yes
Teapot	ArminW	https://bitbucket.org/ArminW/teapot/wiki/Home	No

2.2 HLA & DIS Compliance

While OpenSimulator is a real-time distributed simulation, it was not originally designed to work with either the HLA or DIS standards. This means it cannot presently be used to interface with DND/CF simulation tools that can operate with these standards, such as VBS2, "out-of-the-box". While it is technically possible to develop an HLA or DIS interface in some form to OpenSimulator, it would require a significant software development effort. DND is aware that the U.S. Navy has developed an HLA/DIS gateway to OpenSimulator allowing them to use OpenSimulator as a federation viewer. This demonstrates that OpenSimulator could be operated over a DIS/HLA network. At the time of writing, neither detailed information on how the gateway works, nor access to the gateway technology was available.

2.3 Client Development Using Unity

An area where DND/CF tools could be used to directly interface with the operation of OpenSimulator vWorlds is the development of a client (OpenSimulator viewer) application using Unity. Note that this may not be a desirable or practical application of DND/CF resources because it is likely to be costly and there are many free viewers currently available for download.

As discussed above, a variety of third-party viewers can be used to interface with OpenSimulator. A complaint against these viewers, as discussed by one of the founders of OpenSimulator, Adam Frisby, is their inability to scale to large volumes of users (e.g. rendering more than 60 avatars in a single region, which is unlikely to be an issue for DND applications) and their dated graphics rendering capability compared to state-of-the-art game rendering engines [3]. Since Unity does not suffer from these shortcomings, some vWorld simulation providers are focusing on the Unity platform instead of OpenSimulator to develop virtual world environments and content, and then

deploy them to either the web as a browser plug-in or to private standalone applications. Such companies include Sine Wave Entertainment [4], ReactionGrid [5], Second Places [6], Rezzable [7], TPLD [8], and Hyperfair Inc. [9].

In early 2011, Rezzable produced a Unity-based web-browser viewer that interfaced with OpenSimulator directly [10]. Rezzable's viewer, shown in Figure 1, used a Rezzable module attached to an OpenSimulator region to package the content of the region and send it to a browser. A Unity scene was used to present the content and OpenSimulator was used as the "Massively Multiplayer Online" (MMO) engine. The region content received by the browser was the content present in the region at the time of the download. To see new content, users would have to refresh their browser. As with other viewers, visitors could interact with other avatars accessing the region.

However, it would seem that Rezzable has abandoned their Unity-based OpenSimulator viewer as no mention of it can currently be found on their website. Conversely, Unity remains one of their listed "preferred tools". This may be a result of Rezzable's plan to charge Grid operators a $50/region license fee in order to use the Rezzable module. This module needs to be attached to each region to publish OpenSimulator regions online, when one is not able to use the free OpenSimulator viewers [10].

Some challenges to developing an OpenSimulator client using Unity have been described by Rob Smart [11]. The article referenced mainly applies to web-browser based targets, but could potentially impact standalone client implementations. The primary issue involves incompatibility between the .NET/Mono library version used by OpenSimulator and the Mono version used by Unity. Library incompatibility issues can affect various aspects of simulator operation such as server communication using the "libomv" library. At the time the article was written (2009), OpenSimulator was at version 0.7.0 (.NET 3.5) and Unity was based on Mono 2.0, and the solution was to implement an earlier version of OpenSimulator to align library compatibility and establish server communication. However, at the time of writing this report, both OpenSimulator (version 0.7.5) and Unity (version 4), support .NET 4, and this particular issue is likely to have been resolved.

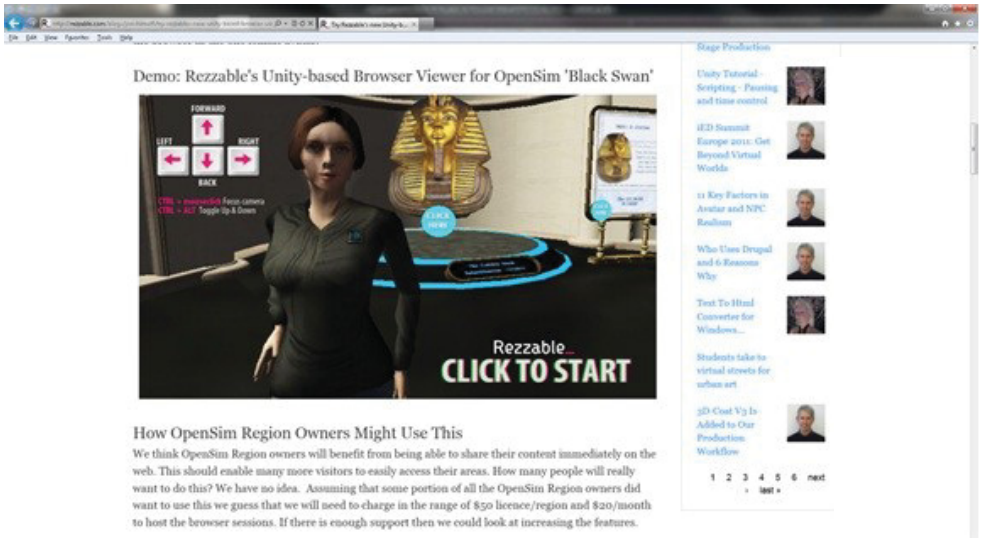

Figure 1: Rezzable's Browser-Based Viewer Used Unity to Present OpenSimulator Content (Source: [10]).

2.4 VBS2 API

The level of interoperability between OpenSimulator and VBS2 is extremely limited in terms of simulator operation due to the inherent differences of the two applications. OpenSimulator is a server-based application, coded in C#, running on the .NET framework, accessing a remote SQL database, connects disparate users via Linden Labs viewer protocol (over HTTP and UDP), and employs a simple height-map-based terrain. VBS2 operates on the Real Virtuality 3 game engine, incorporates constructive AI entities, features complex terrain, and can connect to distributed simulations using DIS or HLA.

The VBS2Fusion product provides an Application Programming Interface (API) to VBS2 that allows the state of objects within the VBS2 environment to be manipulated. This allows users to customize the VBS2 simulation by adding plug-ins to interface with, and gain control of, objects, entities, and other aspects of the virtual battlespace. This allows new features and capabilities to be added to the VBS2 simulated environment. Potentially, the API could be used to develop an interface to OpenSimulator, although it would likely be a significant development effort because OpenSimulator does not support a number of the features of a serious (war) game, including sensor and weapon effects.

2.5 Simulation Operation Summary

This section discussed the interoperability of DND/CF simulation tools with regard to operating, controlling, and/or performing a simulation using OpenSimulator. Overall, there exists little to no compatibility to operate or control OpenSimulator using any of DIS/HLA protocols, Unity, or VBS2 without exerting significant development effort.

OpenSimulator does not currently directly support DIS or HLA protocols for distributed simulation and therefore the capability to participate in distributed simulations exercises with DIS/HLA compliant simulations is currently non-existent. A DIS/HLA gateway developed by the U.S. Navy may allow some interoperability, although details are currently limited. Several efforts to construct a Unity-based client in order to overcome the shortcomings in graphics and scalability to large user numbers were discussed but currently, no Unity-based clients are available to use. Finally, the use of VBS2's API to interface with OpenSimulator was discussed, but it was concluded that there is likely no significant advantage to developing a VBS2 interface with OpenSimulator given the cost of developing such an interface.

However, given that OpenSimulator enables communication between multiple users via Linden Labs protocol (using HTTP and UDP), means that such interoperability is not required in order for DRDC to conduct a distributed simulation, DRDC use case (4). A number of freely available compatible client applications (such as the recommended Firestorm viewer) can be used to connect to OpenSimulator over the Internet.

3 Interoperability of Terrain Models

Terrain is modeled in OpenSimulator as a height-map that can be represented as a grey-scale image where black is "low" terrain and white is "high" terrain. The height-map is the first layer of a 13-layer (or channel) "RAW" format. The channels are labelled: height, factor, water, parcels, for sale, edit object, edit land, safe, flying, landmark, scripts, original height, and original factor. Each channel consists of 256 × 256 eight-bit pixels. This represents one 256 × 256 m^2 region in the simulator where each pixel in the terrain file represents a 1 m^2 area.

This section describes a step-by-step method for exporting a terrain model from Unity and importing it into OpenSimulator as well as briefly addressing terrain export from VBS2. Therefore, this section relates to DRDC use case (1) discussed in section 1, in which content developed for previous simulations by external applications is re-used by OpenSimulator.

3.1 Importing Terrain from Unity

Terrain models created in Unity can be exported in RAW format, and with some intermediate steps, imported into OpenSimulator. An example is used to illustrate the steps involved in this process.

Begin by creating a new terrain (*Terrain > Create Terrain*) in Unity or opening an existing terrain asset (drag a terrain model from the *Asset* window into the *Hierarchy*). Terrain created in Unity is shown in Figure 2.

For new terrain, before adding any terrain features it is possible to configure the RAW file to fit perfectly into a 256 × 256 m^2 OpenSimulator region. This is accomplished by selecting:

1. *Terrain > Set Resolution…*
2. Set the parameter "Heightmap Resolution" to 256 (Unity will add a +1 to the value) which corresponds to 256 × 256 m^2, as shown in Figure 3. Setting a higher resolution will result in a terrain map that will be larger than a single region.
3. If a multiple region terrain is desired, set the resolution to a value which is a multiple of 256. For example, a 2 by 2 set of regions spans 512 m^2 and the terrain created in Unity corresponds to a resolution of 512 (+1). As will be subsequently described, multi-region terrains can be cut up into single 256 × 256 m^2 region parcels and loaded into the simulator piece-wise.
4. The other parameters within the *Set Resolution* dialogue referring to terrain width, length, and height do not affect the height-map that is ultimately used by OpenSimulator.

For existing terrains, the resolution cannot be changed from the original as all features are reset when the height-map resolution is changed. Since the resolution corresponds to the square size of the region in OpenSimulator, checking the height-map resolution parameter will provide the size of the terrain as it will appear in OpenSimulator. If the existing terrain has a height-map resolution larger than 256, it will span more than one region and it will be required in a subsequent step to subdivide the height-map before it can be imported to OpenSimulator.

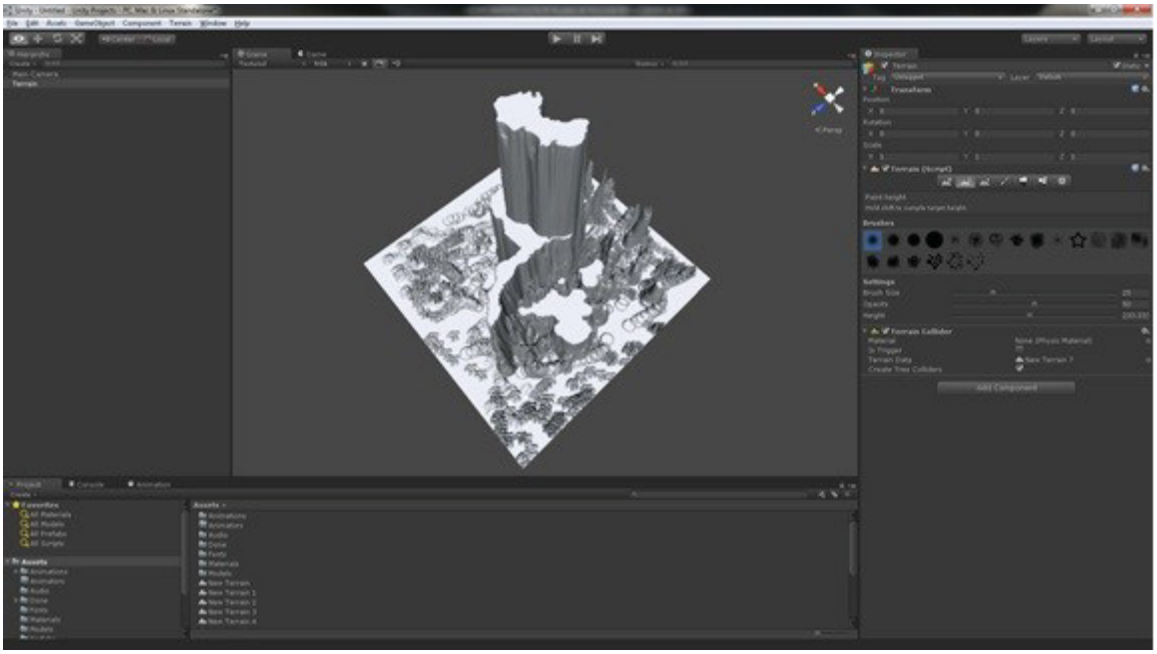

Figure 2: Creating Terrain in Unity.

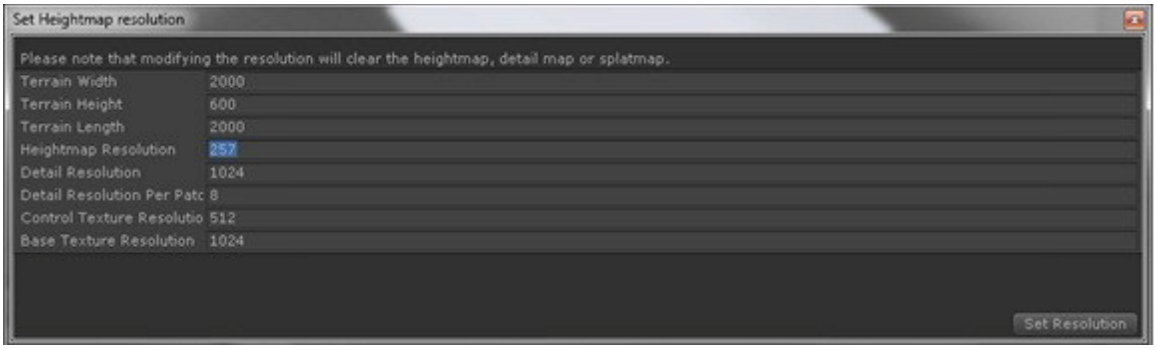

Figure 3: The Terrain > Set Resolution Dialogue.

Next, a RAW height-map file is generated by selecting *Terrain > Export Heightmap – Raw…*

1. The Export Heightmap dialogue will open as shown in Figure 4.
2. Change the parameter *Byte Order* to the respective OS platform.
3. If the height-map resolution was set to 257 in the previous step, then the *Width* and *Height* in the export dialogue will be 257 (an OpenSimulator region size). Otherwise, these values will correspond to the height-map resolution set previously for a larger size of terrain.

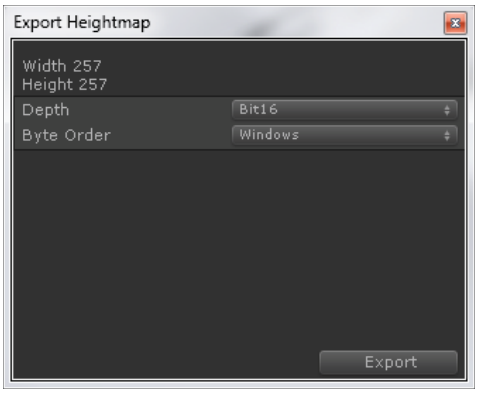

Figure 4: RAW Terrain Export Dialogue.

Unity exports the terrain into a single-channel RAW file containing only the height-map. Since OpenSimulator uses 13-channel RAW files (with the first layer containing the height-map), the Unity RAW file needs to be converted into the RAW format that OpenSimulator can read. One method of providing this conversion is by the following procedure.

First, the height-map data is converted to a Targa image file (.tga) via a terrain converter program that can be downloaded from [12]. This is accomplished by simply dragging and dropping the icon of the Unity RAW file over top of the icon of the converter, terrconv.exe. The Targa image file appears in the same directory and represents a grey scale height-map of the terrain created using Unity. This file can be opened directly by GIMP (the GNU Image Manipulation Program).

The next steps require two files that were provided by the OpenSimulator training program:

(1) *blank.raw*, a 13-channel empty RAW terrain file for OpenSimulator terrains, and

(2) *file_slraw.py*, a Python plug-in for GIMP that allows the 13-channel RAW files to be opened and edited using GIMP. This file must be copied to the following location: *C:\Users\username\.gimp-2.8\plug-ins*.

The steps are:

1. Open the Targa height-map image in GIMP.
2. Open *blank.raw* in GIMP.
3. Make the "Height" layer the visible layer then copy and paste the Targa image onto it.
4. Select *Layer* > *Anchor Layer*.
5. If the Targa image has a resolution of 257, it will fit perfectly into 256^2 pixel layer. Otherwise, if the resolution is larger, the Targa image will have to be subdivided into several 256^2-sized parcels and each piece copied into the "Height" layer of an independent *blank.raw* file. GIMP makes this process relatively easy since the whole Targa height-map can be selected and pasted into the "Height" layer, and then repositioned over the layer. Overlapping portions are automatically excluded.
6. Finally, select *File* > *Export…* and enter the name for the new RAW parcel created.

Note that, with a moderate amount of effort, an application could be developed to combine these steps into a single process, allowing Unity-produced RAW height-maps to be converted to OpenSimulator 13-channel RAW terrain files directly.

The final step is to import each RAW terrain file into an OpenSimulator region. The terrain can be loaded into an existing region (which replaces the existing terrain but leaves all assets such as trees, buildings, or other models intact including their positions within the region) or into a new region.

New regions are created from the OpenSim.exe command console using the command:

1. *create region <regionName> region.ini*.
2. If multiple regions are required, set the coordinates of each new region such that the terrain parcels will align to form the original terrain created in Unity.

To upload the new terrain:

1. Fly or teleport to the region.
2. From the Client/Viewer menu, select the Terrain tab located within the following menu structure depending on the viewer examples below:

   Hippo Viewer: *World > Region/Estate > Terrain*

   Firestorm Viewer: *World > Region Details > Terrain*
3. Click *Upload Raw Terrain…* and select the 13-channel RAW file created using GIMP. The *OpenSim.exe* console will display a lot of scrolling red text. This is normal, and after 20 to 30 seconds the new terrain should be drawn by the viewer. However, if the scrolling text stops with an I/O read error displayed within the console window, it means that the 13-channel RAW terrain file was formatted incorrectly.

Figure 5 shows the terrain created in Unity shown in Figure 2 being displayed in an OpenSimulator viewer after being uploaded into an OpenSimulator region.

Figure 5: Terrain Profile Created in Unity Uploaded to OpenSimulator.

3.2 Importing Terrain from VBS2

VBS2 includes the ability to generate terrain models using the Visitor 4 application [13]. Three primary sources of input data are required to construct the terrain as follows:

1. Height-map/Elevation Data: from Digital Terrain Elevation Data (DTED), Digital Elevation Model (DEM), GeoTIFF, ArcInfo ASCII, or XYZ raster data;
2. Imagery: from satellite images and aerial photography (GeoTIFF or PNG) to apply realistic texture and look to the terrain;
3. Vector Map (VMAP): terrain cultural features (roads, trees, buildings) from ESRI Shapefiles or other VMAP sources.

The height-map data consists of a matrix of terrain elevation values. While VBS2 provides the capability to export this data into an external ASCII file for archive/backup purposes, this format is fundamentally different than the gray-scale image used by OpenSimulator to represent terrain elevation. This means it cannot be used as is by OpenSimulator.

To generate the terrain file that will be used by VBS2, the height-map, imagery, and VMAP are overlaid within Visitor 4, and then exported to a "WRP format" file which is then packed into a compressed format that is read by VBS2 during simulation. This format is unofficially documented on the Bohemia Interactive web site as "internal undocumented structures". This means, while it would be possible to develop a convertor between the VBS2 file format and the OpenSimulator format, there is no guarantee that it would continue to work across releases of VBS2 as the internal undocumented structures may change.

Alternatively, by using third party GIS data processing applications (e.g. Global Mapper), it should be possible to take the height-map or elevation source data used by VBS2 and process it to

produce a gray-scale height-map required by OpenSimulator. Note that, this approach has not been verified during the course of this compatibility study.

3.3 Terrain Data Summary

This section discussed the interoperability of OpenSimulator with terrain models developed by alternate DND/CF simulation technologies, Unity and VBS2. The study revealed that terrain models developed in Unity were compatible with OpenSimulator: Unity terrain can be exported directly as a gray-scale height-map which forms the first layer in the 13-layer RAW terrain file used by OpenSimulator to model terrain. Step-by-step instructions were provided to illustrate the process. Conversely, VBS2, which constructs terrain models using actual terrain data (digital elevation data, imagery, and vector maps), does not export terrain to a gray-scale height-map directly, making it incompatible with OpenSimulator without the aid of additional software to convert the VBS2 terrain model. However, the height-map or elevation source data used by VBS2 should be able to be processed for use by OpenSimulator. Thus, DRDC use case (1) regarding the re-use of simulation content can be achieved with respect to terrain models by using Unity, but not VBS2 directly, unless additional processing is applied to the VBS2 terrain model or the source data for VBS2 is used.

4 Interoperability of 3D Models

An important aspect of interoperability is the potential for reuse of assets, in the form of mesh models, previously constructed for DND/CF's alternate simulation tools.This is related to DRDC use cases (1) and (2), identified in the introduction of this report, which involve the re-use or virtual content developed for use with other simulation applications (import), or transferred from one vWorld application to another (export).

OpenSimulator only supports the COLLADA (.dae) mesh format amongst common 3D model formats. COLLADA was designed as an interchange file format for 3D applications and has been standardised as ISO/PAS 17506. As a result, it is a format that is supported by most 3D modeling applications. Unity and VBS2 support a variety of mesh formats such as COLLADA plus those produced by a spectrum of 3D modeling applications including Maya, Cinema 4D, 3D Studio Max, Cheetah3D, Modo, Lightwave and Blender. Intergraph's SmartMarine 3D and SmartPlant 3D ship and plant design software support the PDS, PDMS, ACIS (.sat), MicroStation (.dgn), and AutoCAD (.dwg) formats, but not COLLADA directly [14]. An important limitation of OpenSimulator is that while it can import objects in COLLADA format, it does not provide a means to export individual objects back out for direct use with other tools. Like other simulation and gaming engines, such as Unity and VBS2, OpenSimulator is a tool that consumes 3D virtual content. While OpenSimulator can create specialized archive files (called OpenSimulator ARchive or "OAR" files and Inventory ARchive or "IAR" files) that effectively export vWorld 3D content and assets, these currently only work with other instances of OpenSimulator. The possibility of extending this capability is discussed in section 4.6.

While OpenSimulator supports the COLLADA format, the ability to import COLLADA models and, separately, view them, depends upon the viewer selected. The Hippo viewer neither supports the ability to upload nor view such models. Another viewer, Singularity, can view mesh models, but not upload them. Firestorm is one popular viewer that provides both capabilities and thus it is recommended that DRDC adopt this viewer. We note that Cool VL, Kokua, and Teapot also support both capabilities. Table 2 summarizes the mesh capabilities of the viewers that were tested for this study. In the following discussion, Firestorm will be used to demonstrate how a mesh model developed for Unity or VBS2 can easily be imported into OpenSimulator. To properly setup Firestorm to connect to a private OpenSimulator grid, refer to Annex A.

Table 2: Viewer Mesh Capability

Viewer	Mesh Upload	Mesh View
Cool VL	Yes	Yes
Firestorm	Yes	Yes
Hippo	No	No
*Imprudence	N/A	N/A

Viewer	Mesh Upload	Mesh View
Kokua	Yes	Yes
*Phoenix	N/A	N/A
Singularity	No	Yes
Teapot	Yes	Yes

*Discontinued viewer

4.1 Conversion of 3D Studio Max Mesh to COLLADA

The 3D modeling and animation software, 3D Studio Max (3DS Max) is a ubiquitous and common application for the development of 3D assets for gaming and simulation applications. While 3DS Max can export to COLLADA directly, the following example illustrates how free 3D modeling application Blender (version 2.65)[1] can be used to convert 3DS Max assets in .3ds format to COLLADA .dae format.

1. Open Blender.
2. Select *File > Import > 3D Studio (.3ds)*.
3. Choose the file and either double-click or click *Import 3DS* in the top right of the GUI to import the 3DS Max model.
4. Select *File > Export > Collada (Default)(.dae)*.
5. Chose the location and click *Export COLLADA* in the top right of the GUI to export the model to .dae format.

4.2 Conversion to COLLADA using SketchUp

The 3D modeling and animation software, SketchUp (owned by Google between 2006 and 2012, now by Trimble Navigation) is another application for the development of 3D assets. In the current version, SketchUp Pro 2013, it natively uses its own format for 3D models (.skp), but is able to import and export a number of other formats, including COLLADA (.dae), 3DS Max (.3ds) and AutoCad (.dwg, .dxf) formats.

The following steps illustrate how to import and export from SketchUp Pro 2013.

1. Open SketchUp.
2. Select *File > Import...*
3. Within the dialog, choose the appropriate *Files of type* if it is not the default.

[1] Note that Blender version 2.49b was used during the vWorld Training Course to demonstrate how to make and import sculpted mesh objects into OpenSimulator. This version is not recommended for mesh conversion due to the quality of its COLLADA implementation.

4. Choose file and either double-click or click *Open* to import the model.
5. Select *File* > *Export* > *3D Model...*
6. Within the dialog, set the *Export type* to *COLLADA File (*.dae)* if it is not the default.
7. Chose the location, set the *File name* and click *Export* to export the model to .dae format.

4.3 Conversion of SmartMarine/Plant 3D to COLLADA

Intergraph's design software for ships and plants, SmartMarine 3D and SmartPlant 3D, respectively, provide support for the PDS, PDMS, ACIS (.sat), MicroStation (.dgn), and AutoCAD (.dwg) formats, but not COLLADA, the only format accepted by OpenSimulator. Since Intergraph's software cannot export to COLLADA directly, they must be converted by another program. Unfortunately, Blender does not support any of these formats and therefore it cannot be used to convert them to COLLADA for import into OpenSimulator as was the case with 3DS Max.

One possible workflow would import the models from either SmartMarine 3D or SmartPlant 3D into Autodesk's AutoCAD 2014, which supports .sat and .dgn models. These would then be exported to .fbx format. Then, 3DS Max would be used to import the .fbx file and export to .dae (COLLADA). Alternatively, a second workflow (again using AutoCAD 2014) would involve exporting the .sat/.dgn file to .dwg format. This file would then be imported into SketchUp and exported as .dae [15]. These workflows would maintain the model geometry, but some model information (materials, textures) may be lost.

Neither approach was verified during the compatibility study. We recommend that models produced by SmartMarine 3D or SmartPlant 3D be obtained and the two workflows tested to ascertain the efficacy of the approach and the quality of the result.

Import COLLADA to Unity

Unity supports a variety of 3D model files including .fbx, .3ds, .dxf, .obj, and .dae files. The following steps can be used to import a COLLADA model into Unity as shown in Figure 6.

1. Open Unity.
2. Select *Assets* > *Import New Asset…*
3. Select the COLLADA model file and click *Import*.
4. The COLLADA model will appear listed in the Assets window.
5. Drag the model into the *Hierarchy* window and it will appear in 3D in the *Scene* window.

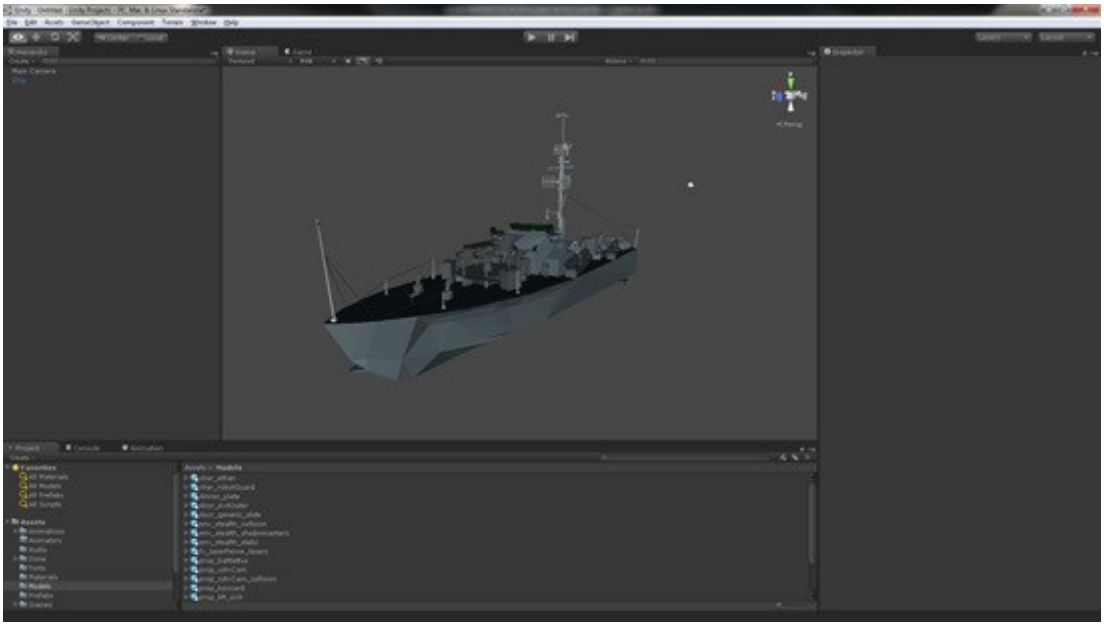

Figure 6: COLLADA Model Import into Unity (Source: [27]).

4.4 Import COLLADA to OpenSimulator

OpenSimulator supports COLLADA mesh files. Thus, any COLLADA-based mesh asset used in Unity or VBS2 can also be used by OpenSimulator. To import a mesh model in COLLADA format into OpenSimulator, a viewer supporting mesh uploads such as Firestorm must be used. Refer to Annex A for information on installing this viewer.

The following procedure will upload a COLLADA mesh into OpenSimulator, such as the one shown in Figure 7. From the Firestorm viewer:

1. Select *Build* > *Upload* > *Mesh Model*…
2. Select the COLLADA (.dae) file and click *Open*.
3. The *Upload Model* dialogue appears. Several options are available for selecting the level of detail (number of triangles and vertices), applying physics to the model, and previewing the model. Under the *Level of Detail* tab select either High, Medium, Low, or Lowest labels. These can be previewed in the preview pane on the right by selecting the corresponding level from the drop-down menu. The preview can be zoomed in by using the mouse scroll wheel or by left-clicking and dragging vertically upwards. The preview can also be rotated by dragging the mouse horizontally while left-clicking on the image.
4. Click the Calculate weights and fees button. This feature calculates the computational cost of the model on the simulation. The calculation may take several minutes to complete. Once it does, the button will change to *Upload*. Alternatively, a lower level of detail could be selected and the cost recalculated.

5. Click the Upload button to upload the model to the user's inventory. The inventory dialogue will automatically open once the upload is complete showing the model in the Objects category.
6. The model can be placed into the OpenSimulator environment by dragging it from the inventory into the 3D virtual world.
7. Right-clicking on the model and selecting Edit will launch the editing tool.

Once the model has been imported, its size in each of the X, Y and Z directions can be changed but not its shape: direct editing of the mesh itself is not possible. Several different models can be grouped together so that they move as a single model by holding "Ctrl", selecting each model, and then (in the Firestorm viewer) going to Build > Link.

Figure 7: COLLADA Model Import into OpenSimulator

4.5 Limitations on Exporting Models for OpenSimulator

As Unity and VBS2 are both consumers of 3D model assets and not producers of these assets, neither application is capable of directly exporting model content out in a form that can be easily reused by third-party applications such as OpenSimulator. For example, in Unity, assets can be imported in a variety of formats. But assets can only be exported to a Unity-friendly package for import to alternate instances of Unity itself. On export, models are broken down into their disparate parts, categorized into a multitude of files, and then compressed into gzip format (obscured by Unity with a .unity extension).

VBS2 content can be packaged and exported out to a "Packed Bohemia Object" (PBO) file. These files can contain any VBS2 object that is represented within the Mission Editor folder tree such as missions, campaigns, add-ons, animations, or 3D models. Within the PBO file, the models are represented as P3D files. The PBO export function is designed to work only with

other instances of VBS2 (which can directly import the PBO file) and it does not directly facilitate exporting out 3D assets for use with OpenSimulator. However, Bohemia Interactive does produce a P3D-to-FBX converter and the resulting FBX files can either be imported to AutoCAD and 3DS Max [16]. By using 3DS Max, the model could then be exported either directly to COLLADA format, or to the 3DS format that Blender, as per section 4.1, can convert to the required COLLADA format. Due to legal limitations on the content provided with VBS2, it should be verified before exporting any 3D model out of VBS2 that the user or organization has the legal right to do so.

4.6 Exporting Models Out of OpenSimulator

Similar to Unity and VBS2, OpenSimulator is a consumer of 3D model assets that are built using other tools. While most OpenSimulator viewers provide the ability to create, edit, and upload objects inside the virtual world there is currently no mechanism available to easily export those objects out of OpenSimulator for use with other DND/CF simulation technologies.

4.6.1 Archive Files

OpenSimulator does provide the ability to save archived copies of whole regions (OAR-files) and inventory items (IAR-files), but these are designed to work with other instances of OpenSimulator and, to our knowledge, no free or commercial tools have been developed that would allow the information stored in these files to be converted to other formats, such as COLLADA, that could be imported into other programs.

A tool could be developed that would read in the 3D object data contained in an OpenSimulator archive file, since the object data within these archives is stored in XML format, and use it to construct the 3D model formats required by other applications. The OAR file contains a sub-folder with XML files that correspond to each object within the vWorld. For groups of "linked" objects, as described in section 4.4, the OAR file combines the objects into one XML file. Therefore, an export tool could work on individual as well as "linked" groups of objects.

4.6.2 OpenSimulator Database

Details of the OpenSimulator regions, inventories, and other details, are also stored within the OpenSimulator database. The database schema is partially documented on the OpenSimulator web site [26], although the structure is considered "alpha" and may not be stable between OpenSimulator releases.

A tool could be developed that would read the 3D object data contained in the OpenSimulator database and use it to construct the 3D model formats required by other applications. Similar to an export tool developed to read the archive files, such a tool could work on individual as well as "linked" groups of objects.

4.6.3 Third Parties

Further investigation and experimentation into the feasibility of these proposed tools is required, but the ability to export OpenSimulator model data to other formats has been demonstrated previously by third parties who developed their own proprietary methods. For example, Tip-o-de-an Technologies provides a conversion service that transforms OpenSimulator objects, including entire regions, into a COLLADA mesh that can then be imported into Unity [17]. Additionally, DRDC is aware of software developed by the U.S. Navy that exports assets or models created in OpenSimulator into a format that can be imported into Unity. At the time of writing, no references to this work were available.

4.7 3D Models Summary

This section discussed that OpenSimulator is directly compatible only with 3D model files in the COLLADA format and showed how interoperability can be achieved using a range of free and commercial tools for conversion, notably Blender, AutoCAD, SketchUp and 3DS Max, as well as source and target applications including SmartMarine 3D/SmartPlant 3D, Unity, and VBS2. Where conversion to COLLADA is indirect, involving several intermediate tools, it is likely that if the original model is CAD-based and includes material information, then this information will be lost and only the geometric model component will be preserved. This is not an issue since OpenSimulator does not interpret any other information other than the model geometry.

It was discussed that OpenSimulator, as a virtual world simulator, was not designed to produce 3D model content for use with other applications, but was designed to consume content for building virtual worlds. It therefore includes features to import models and textures, and internal tools to build virtual environments but not to export them back out in a format compatible with other external tools. It was proposed that model export functionality could be added with considerable development effort, by using the geometric model information contained within OpenSimulator archive files or the database.

Based on the discussion presented above, the DRDC use case of content re-use via import of existing model assets can be achieved. However, as far as model re-use for content developed within OpenSimulator, no course for direct export to external applications is available.

5 Interoperability with Human Factors Tools

Human Factors is the study of how devices and equipment work with the human body and the brain's cognitive abilities. It incorporates the disciplines of engineering and psychology, anthropometry, biomechanics, and industrial design [18]. Within an OpenSimulator virtual world, the avatar represents the user. It may or may not look like the user depending on how the avatar appearance has been configured. It may not even appear human although most avatars tend to be humanoid. Thus, emphasis tends to be on avatar personalization rather than on optimizing fidelity to the human form factor.

This section discusses possible areas of compatibility between OpenSimulator and two Human Factors software applications: HumanCAD and IPME. It will address DRDC use cases (1), importing existing content, and (3), integrating OpenSimulator as part of a larger simulation.

5.1 The OpenSimulator Avatar

5.1.1 Default Appearance and Customization

The default avatar loaded by running OpenSimulator for the first time is called "Ruth": a generic female avatar with default body shape configuration and clothing options selected as shown in Figure 8. Assuming that the Firestorm viewer is used, these settings can be changed by right-clicking the on the avatar and selecting *Appearance* > *Edit Outfit*. Three tabs present the options "Clothing", "Attachments", and "Body Parts".

Clicking on the "Body Parts" tab reveals a list that displays the avatar's default eyes, hair, shape, and skin. Hovering the mouse over any of these will illuminate the item and show a wrench icon which can be clicked to edit the default settings. Alternatively, the edit feature can be opened by right-clicking the item and selecting *Edit*. Right-clicking also reveals options to *Replace* the existing body part with something else, or *Create* a new one.

Editing the *Shape* body part involves adjusting various sliders that control the shape parameters for each of the categories: Body, Head, Eyes, Ears, Nose, Mouth, Chin, Torso, and Legs. The *Eyes*, *Hair*, and *Skin* body parts, in addition to providing various configuration parameters, also allow a texture to be uploaded which can effectively alter the look of the avatar as much as changing the parameters settings. Textures must be uploaded in advance using either *Avatar* > *Upload* > *Image* or *Build* > *Upload* > *Image*. New textures are added to the user's inventory under the *Textures* category.

Changing the skin texture has a pronounced effect on the avatar appearance. Three textures, one each for the head, upper body, and lower body, are required. As shown in Figure 8, new skin textures were uploaded to the user inventory, and then a new skin was created to which the skin textures were applied. These skin textures are shown in Figure 9. Alternatively, instead of creating a new skin, the default Ruth skin texture could be replaced with the new one. In order for new user avatars to automatically be configured with the new default, the textures and shape

settings have to reside within the centralized ROBUST database rather than an individual user's inventory.

Similarly, under the *Clothing* tab, different types of clothing can be applied to the avatar, and reconfigured, including applying new textures to simulate more complex clothing patterns than the default colours provided. Figure 10 shows the clothing textures used to customize the avatar in Figure 8.

Figure 8: Default "Ruth" Avatar (left) and Customized Avatar (right).

Figure 9: Skin Textures Applied to Head, Upper, and Lower Body (Source:[19]).

Figure 10: Clothing Textures Used to Configure a New Shirt and Pair of Pants (Source:[19]).

5.1.2 Avatar Model

OpenSimulator avatar models consist of the mesh components that form a "bone structure" and a set of animations that simulation the body kinematics of walking, running, flying, sitting, and other possible movements. Two applications are primarily used to construct the avatar models and their corresponding animations:

1. Blender
2. Make Human

A biped Blender model is shown in Figure 11. This model, a set of tutorials, and a listing of best practices can be found on the OpenSimulator website [20].

These tools could be used to customize the base avatar mesh model in order to increase the fidelity of the anthropomorphic features. Alternatively, it is possible to use specialized Human Factors tools to create a mesh model of a humanoid in various poses of interest and import them into either:

(a) OpenSimulator directly, but the imported model would simply be geometric with no animation capabilities, or

(b) Blender first, to construct, piece-wise, the avatar components modelled by the Human Factors tool, and adding in the other avatar components such as animations before importing the avatar model into OpenSimulator.

Figure 11: Blender Biped Avatar Model (Source: [20]).

5.2 HumanCAD Compatibility

The ability to reuse the HumanCAD anthropometric mannequins within the OpenSimulator environment would provide a natural avenue for compatibility and would apply directly to DRDC use case (1). HumanCAD 2 comes standard with a variety of supported model formats including DXF, FBX, OBJ, 3DS, and others. With the optional CADExchange module, models in IGES, STEP, and STL can be imported and exported (DWF can also be exported) [21]. However, since OpenSimulator supports only the COLLADA format, models exported from HumanCAD could not be directly imported without an intermediate conversion step. Further, the exported mannequin models would be geometric only: the ability to pose a mannequin would be lost during the export process. The free 3D modelling software Blender can be used to solve the problem of format conversion, as can a number of commercial software tools, and potentially Blender could also be applied to the problem of restoring body articulation, although this was not tested during this compatibility study.

Blender can import at least three of the model formats exported by HumanCAD. These include:

1. OBJ,
2. 3DS, and
3. STL.

The first two model formats come standard with HumanCAD while the last model format is enabled with the optional CADExchange module. Blender can then export the model in COLLADA format which can be imported into OpenSimulator using a mesh-compatible viewer such as Firestorm. However, as previously mentioned, the model will be geometric only, and thus an exported mannequin model will lose all body articulation.

As discussed in the previous section, the OpenSimulator community has made available a Blender-based (.blend format) avatar model that models not only the geometry of the avatar (its "bone" structure), but the body kinematics that simulate body articulation and motion [20]. This model could provide a base avatar upon which the exported HumanCAD mannequin model could built upon to form a hybrid avatar compatible with OpenSimulator. The hybrid model would be

constructed by replacing the base bone structure of the blender avatar with the exported HumanCAD mannequin model. The modelled avatar kinematics would then be incorporated, if not inherited, from the base model and subsequently tuned as required to provide body articulation within the human ranges of motion. The hybrid model could then be used in OpenSimulator as an avatar that could be posed and controlled for the purpose of conducting HF-based assessments.

In addition, any model-based infrastructure used in HumanCAD could be reused in the OpenSimulator virtual world. Similar to the mannequin models, modelled objects such as equipment and furniture can be exported from HumanCAD into one of the three Blender-compatible formats, imported into Blender, and then exported to a COLLADA model. Finally, the COLLADA model would be imported by OpenSimulator and positioned in the virtual world.

5.3 IPME Compatibility

Alion MA&D's Integrated Performance Modelling Environment (IPME) has the ability to link to other simulation environments in real-time through TCP/IP sockets or the HLA protocol [22]. As a result, there is the potential for task network models developed in IPME to interface with entities in OpenSimulator with some development effort. This would facilitate DRDC use case (3). For example, DRDC Toronto has experience in utilizing IPME simulation input-output protocols to engage with external simulators such as the Fire and Smoke Simulation Model (FSSIM). This was used in DRDC Toronto's study of crewing effectiveness and automation to simulate a ship crew's effectiveness in extinguishing a fire [23].

In the crew effectiveness simulation, FSSIM modelled the spread of fire and smoke within a specific environment (such as a ship), where different subsystems, such as ventilation systems, automatic fire suppression systems, doors, and valves were also modelled. IPME modelled the crew performance via a network of tasks that must be performed including the time to detect the fire, make assessments, as well as time it takes to take appropriate actions to contain the fire such as turning off ventilation systems and closing bulkheads, doors, and valves. These tasks are coupled with a stochastic model.

A "Model Bridge" application was developed in Java to allow IPME to poll the outputs from FSSIM. Communication was not performed in parallel with the execution of the two simulators due to an FSSIM limitation. Instead, FSSIM was run in advance under a variety of initial conditions to create a set of deterministic outputs that the Model Bridge could poll based on the current success of the simulated IPME crew. The simulation continued until the fire was completely suppressed and extinguished.

Applying a similar approach to OpenSimulator is probably not appropriate since the purpose of IPME is to simulate a crew of individuals whereas the purpose of OpenSimulator is to allow groups of real people to interact via their avatars. In such a scenario, crew simulation is unnecessary given the fact that a real crew could be used instead. Regardless, OpenSimulator does provide a way to provide access to its virtual world environment by other simulation technologies via region modules.

Region modules have access to all of OpenSimulator's core functions. This allows modules to be developed that interface with a variety of events such as logging, chat messages, user logins, texture transfers, and positioning of object prims [24]. To illustrate how to implement a region module, the OpenSimulator website provides a "Hello World" example which writes the phrase in large block letters (primitive objects) floating within the region and moves them every few seconds [25]. This could provide external simulation tools a way to interface with OpenSimulator or for "bridge" applications to provide an interface similar to how the Model Bridge application facilitated communication between FSSIM and IPME. Development would be required to create this interface.

5.4 HF Tools Summary

This section discussed the interoperability between OpenSimulator and human factors tools HumanCAD and IPME. In terms of the DRDC use case scenarios, model re-use of the human anthropometric mannequins could be enabled by exporting to a Blender-compatible format where the motion kinetics could potentially be incorporated and the new model subsequently exported from Blender to the OpenSimulator compatible format. OpenSimulator could potentially be integrated with IPME through the use of Region modules, making OpenSimulator a part of a larger simulation (DRDC use case (3)). However, without further investigation, the feasibility of this proposed approach is uncertain.

6 Compatibility Checklist

The following checklist can be used to determine various aspects of OpenSimulator compatibility with third-party tools. The checklist summarizes all the features and functions that are relevant when determining the compatibility and were used in the preparation of this report. Note that a simulation tool can be compatible in one aspect and incompatible in another aspect simultaneously.

Table 3: OpenSimulator Compatibility Checklist for Technology X.

Question	**Yes?**	**If Yes:**
Simulation Operation		
Does X use the Linden Labs (Second Life) Protocol?		Directly compatible
Does X have a well-documented external communication format (e.g., HTTP, other forms of TCP or UDP)?		Development of interface technically possible but may require significant effort
Does X have an API?		Development of interface technically possible but may require significant effort
Otherwise		Not compatible
Viewer Features		
Does X support COLLADA Mesh viewing?		Directly compatible
Does X support COLLADA Mesh loading into OpenSimulator?		Directly compatible
Terrain Generation		
Does X generate the Linden Labs 13-layer RAW terrain format?		Directly compatible
Does X generate / export height map data in an image format that can be read by GIMP?		Indirectly compatible
Does X generate / export height map data in a format that another tool (e.g., terrconv.exe) can convert to an image format that can be read by GIMP?		Indirectly compatible

Does X generate / export height map data in an officially documented format?		Development of a convertor is possible; effort required is likely to be moderate
Does X generate / export height map data in an unofficially documented format?		Development of a convertor is possible but is not guaranteed to work across releases of X; effort required is likely to be moderate
Otherwise		Not compatible
3D Model Generation		
Does X generate / export files in Linden Labs sculpted primitive format?		Directly compatible
Does X generate / export files in the COLLADA format?		Directly compatible
Does X import OpenSimulator archive files (OAR / IAR)?		Directly compatible
Does X generate / export files in a format that another tool (e.g., Blender, 3DS Max, SketchUp) can read and then export in the COLLADA format?		Indirectly compatible
Can the files imported by X be converted to COLLADA format by another tool (e.g., Blender, 3DS Max, SketchUp) and therefore also be imported to OpenSimulator?		Indirectly compatible
Does X generate / export files in an officially documented format?		Development of a convertor is possible but effort may be significant
Does X generate / export files in an unofficially documented format?		Development of a convertor is possible but is not guaranteed to work across releases of X; effort may be significant
Otherwise		Not compatible

Table 4 contains the compatibility checklist for each of the major technologies considered in this report. Within it: ✓ means "yes", × means "no", and - means "not applicable".

Table 4: OpenSimulator Compatibility Checklist for DIS, HLA, Unity, VBS2, HumanCAD and IPME.

Question	**Yes?**						**If Yes:**
	DIS	**HLA**[2]	**Unity**	**VBS2**	**HumanCAD**	**IPME**	
Simulation Operation							
Does X use the Linden Labs (Second Life) Protocol?	×	×	×	×	×	×	Directly compatible
Does X have a well-documented external communication format (e.g., HTTP, other forms of TCP or UDP)?	✓	✓	×	×	×	✓	Development of interface technically possible but may require significant effort
Does X have an API?	×	✓	✓	✓	×	✓	Development of interface technically possible but may require significant effort
Otherwise	-	-	-	-	✓	-	Not compatible
Viewer Features							
Does X support COLLADA Mesh viewing?	-	-	✓	✓	-	-	Directly compatible
Does X support COLLADA Mesh loading into OpenSimulator?	-	-	×	×	-	-	Directly compatible
Terrain Generation							

[2] For specific HLA-compatible products and implementations

Question	Yes?						If Yes:
	DIS	HLA[2]	Unity	VBS2	HumanCAD	IPME	
Does X generate the Linden Labs 13-layer RAW terrain format?	-	-	×	×	-	-	Directly compatible
Does X generate / export height map data in an image format that can be read by GIMP?	-	-	×	×	-	-	Indirectly compatible
Does X generate / export height map data in a format that another tool (e.g., terrconv.exe) can convert to an image format that can be read by GIMP?	-	-	✓	×	-	-	Indirectly compatible
Does X generate / export height map data in an officially documented format?	-	-	✓	×	-	-	Development of a convertor is possible; effort required is likely to be moderate
Does X generate / export height map data in an unofficially documented format?	-	-	×	✓	-	-	Development of a convertor is possible but is not guaranteed to work across releases of X; effort required is likely to be moderate
Otherwise	-	-	-	-	-	-	Not compatible
3D Model Generation							
Does X generate / export files in Linden Labs sculpted primitive format?	-	-	-	-	×	-	Directly compatible
Does X generate / export files in	-	-	-	-	×	-	Directly compatible

Question	Yes?						If Yes:
	DIS	HLA[2]	Unity	VBS2	HumanCAD	IPME	
the COLLADA format?							
Does X import OpenSimulator archive files (OAR / IAR)?	-	-	×	×	×	-	Directly compatible
Does X generate / export files in a format that another tool (e.g., Blender, 3DS Max, SketchUp) can read and then export in the COLLADA format?	-	-	×	×	✓	-	Indirectly compatible
Can the files imported by X be converted to COLLADA format by another tool (e.g., Blender, 3DS Max, SketchUp) and therefore also be imported to OpenSimulator?	-	-	✓	✓	✓	-	Indirectly compatible
Does X generate / export files in an officially documented format?	-	-	-	-	✓	-	Development of a convertor is possible but effort may be significant
Does X generate / export files in an unofficially documented format?	-	-	-	-	-	-	Development of a convertor is possible but is not guaranteed to work across releases of X; effort may be significant
Otherwise	-	-	-	-	-	-	Not compatible

Table 5 contains an abbreviated compatibility checklist for the OpenSimulator viewers considered in this report. Within it: ✓ means "yes", × means "no", and - means "not applicable".

Table 5: Abbreviated OpenSimulator Compatibility Checklist for OpenSimulator viewers.

Question	**Yes?**								**If Yes:**
	Cool VL	**Firestorm**	**Hippo**	**Imprudence**	**Kokua**	**Phoenix**	**Singularity**	**Teapot**	
Viewer Features									
Does X support COLLADA Mesh viewing?	✓	✓	×	×	✓	×	✓	✓	Directly compatible
Does X support COLLADA Mesh loading into OpenSimulator?	✓	✓	×	×	✓	×	×	✓	Directly compatible

OpenSimulator provides a 3D persistent virtual world in which humans may interact through their avatars. As such, it serves a different purpose from each of the other technologies considered in this report which have different use cases and may all be considered to be complementary. Interoperability is likely to be most useful for the sharing of Terrain and 3D Models rather than direct connection of the technologies, which would be likely to involve significant development effort.

7 Conclusion

This compatibility study explored the interoperability of OpenSimulator with a set of DND/CF simulation technologies and standards which include DIS, HLA, Unity, VBS2, SmartMarine 3D/SmartPlant 3D, and human factors applications HumanCAD and IPME. Interoperability was investigated over four main categories: simulator operation, terrain models, 3D model compatibility, and human factors tools. It also examined compatibility in the context of four DRDC use cases which included (1) import of existing vWorld content, (2) export of OpenSimulator content, (3) integration of OpenSimulator into a larger simulation scenario, and (4) enabling of distributed simulation exercises.

The study of the interoperability of simulator operation first revealed that OpenSimulator has the ability to launch distributed simulations via HTTP and UDP protocols, and therefore the DRDC use case (4), requiring the ability to perform distributed simulations, could be achieved without any additional tools or development. As far as interoperability with other DRDC technologies, the study found that Unity could potentially be used to develop a client/viewer for OpenSimulator. While limited success has been shown from using Unity directly as an interface with OpenSimulator, the potential to leverage Unity's advanced graphical capabilities and ability to scale to large numbers of users provides some motivation to explore this development path further if this capability is desired. The capability to develop plug-ins to interface with VBS2 via its API infrastructure could provide some compatibility in terms of simulator operation, although it is unclear what utility an OpenSimulator-to-VBS2 simulation operation plug-in would have given that the two tools have different purposes.

Investigation of terrain model interoperability found that Unity's terrain creation tools can be used to export a terrain height-map which can then be combined with OpenSimulator's 13-channel RAW format using the GIMP image manipulation software and a third-party conversion tool. An example demonstrated the process where a terrain model built in Unity was imported into OpenSimulator. Thus, DRDC use case (1) regarding the re-use of existing simulation content could be realized with regards to terrain models produced by Unity. Conversely, it was determined that while VBS2 has advanced terrain generation capabilities, these capabilities do not extend to exporting those models for use in a directly compatible format with OpenSimulator.

The investigation into the interoperability of 3D mesh models discussed how a spectrum of 3D modelling and animation software could be used to generate the assets used to populate the virtual world environments created by Unity, VBS2, SmartMarine 3D, and SmartPlant 3D. These mesh models include avatars, buildings, vehicles, and other objects. It was shown how these models could be converted to COLLADA format, and then imported into OpenSimulator using the Firestorm viewer. Full interoperability exists for 3D models between OpenSimulator and these alternate simulation tools due to the universality of the COLLADA format. Therefore, the DRDC use case (1) was extended beyond the re-use of terrain models to all virtual content models in general. Regarding DRDC use case (2), content export, it was discussed that OpenSimulator, like Unity and VBS2, does not intrinsically include the ability to export out models to COLLADA format (or any other 3D format) regardless of whether these models were originally imported into, or created within, OpenSimulator. It also does not have the ability to edit the shape of the imported COLLADA model, save for specifying the length, width, and height dimensions.

However, virtual content can be transferred from one OpenSimulator instance to another via archive OAR files.

The Human Factors tools HumanCAD and IPME were also evaluated for compatibility at a high level. In another example of the model import and DRDC re-use cases, it was discussed that HumanCAD uses articulated mannequin models for anthropomorphic visualization that can be exported to a standard set of external model formats. It was discussed how Blender can be used to convert those models to COLLADA format which can then be imported into OpenSimulator. The loss of the articulated component of the mannequin can potentially be restored by using the avatar model for OpenSimulator that is freely available in Blender to combine the HumanCAD model components with the Blender avatar kinematic animation model.

IPME compatibility was demonstrated through a DRDC case study where a "Model Bridge" application was built to facilitate communication between IPME and FSSIM. This example illustrates how IPME could potentially interface with OpenSimulator. OpenSimulator's region modules also provide a level of customization that allows external applications to interface with the simulator's base functions. This could potentially allow the integration of OpenSimulator with a larger IPME discrete event simulation – an example of DRDC use case (3).

Finally, a compatibility checklist was presented which summarizes each area of compatibility that was examined in this study and could be applied by DRDC to assess alternate software technologies that were not assessed in this report for compatibility with OpenSimulator.

This report presents some detailed analysis of OpenSimulator compatibility and often demonstrates the level of interoperability through step by step examples. However, due to time and scope constraints, in some cases only a high level of assessment is provided, leading in several instances to potential areas where further and more detailed investigation and experimentation is required to definitively conclude the level of compatibility. Depending on DRDC priorities, these areas could provide avenues for future investigation or even development of technology to enable compatibility where currently there is none. For example, further investigation could be applied to the development of a DIS or HLA interface for OpenSimulator, development of content export capability from OpenSimulator, and demonstration of model and/or terrain export from VBS2 to OpenSimulator.

This page intentionally left blank.

References

[1] OpenSimulator.org. “Communication Protocols”. Internet: http://opensimulator.org/wiki/Communication_Protocols. 11 March 2013 [March 2013].

[2] OpenSimulator.org. “Connecting”. Internet: http://opensimulator.org/wiki/Connecting. 7 December 2012 [March 2013].

[3] Alexander Gladstone. “OpenSim founder goes for Unity”. Internet: http://www.hypergridbusiness.com/2012/02/opensim-founder-goes-for-unity. 7 February 2012 [March 2013].

[4] Sine Wave Entertainment. Internet: http://www.sinewavecompany.com/. 2012 [March 2013].

[5] ReactionGrid Inc. “Jibe”. Internet: http://reactiongrid.com/. 2012 [March 2013].

[6] Second Places. “Second Places Unifier”. Internet: http://www.secondplaces.net/opencms/opencms/virtualWorlds/unifier/. 2011 [March 2013].

[7] Jon Himoff. “Try Rezzable’s new Unity-based Browser Viewer for OpenSim”. Internet: http://rezzable.com/blogs/jon-himoff/try-rezzables-new-unity-based-browser-viewer-opensim. 2 February 2011 [March 2013].

[8] TPLD. “Virtual Events”. Internet: http://hostavirtualevent.com/. [March 2013].

[9] Hyperfair. Internet: http://www.hyperfair.com/. 2012 [March 2013].

[10] Jon Himoff. “Try Rezzable’s new Unity-based Browser Viewer for OpenSim”. Internet: http://rezzable.com/blogs/jon-himoff/try-rezzables-new-unity-based-browser-viewer-opensim. 2 February 2011 [April 2013].

[11] Rob Smart. “The Challenges of writing an OpenSim client in Unity3D”. Internet: http://robsmart.co.uk/2009/11/12/the-challenges-of-writing-an-opensim-client-in-unity3d/. 12 November 2009 [March 2013].

[12] Terrain Converter. Internet: http://dubaron.com/terrainconvert/. [March 2013].

[13] Bohemia Interactive Australia. “VBS2 Visitor 4 Terrain Editor’s Manual”, 1.30 Tools.

[14] Intergraph, “Intergraph® Releases New Version of SmartMarine® 3D Software for Offshore and Ship Design”. Internet: http://www.intergraph.com/assets/pressreleases/2010/05-05-2010.aspx. 5 May 2010 [July 2013].

[15] Autodesk, “Discussion Group: Is there a Collada (DAE) Export”. Internet: http://forums.autodesk.com/t5/Autodesk-Revit-Architecture/Is-there-a-Collada-DAE-Export/td-p/3306713. 25. January 2012 [July 2013].

[16] Bohemia Interactive Simulations, “P3D to FBX converter”. Internet: http://resources.bisimulations.com/wiki/P3D_to_FBX_converter#Summary. 2012 [July 2013].

[17] Tip-o-de-an. “Convert your OpenSim Island into COLLADA to run within Unity3D”. Internet: http://www.tipodean.com/converter/index.html. 2010 [March 2013].

[18] Wikipedia. “Human factors and ergonomics”. Internet: http://en.wikipedia.org/wiki/Human_factors_and_ergonomics. 25 April 2013 [April 2013].

[19] Jamie Wright. “The Ginger Punk Complete Make Avatar Update”. Internet: http://opensim-creations.com/2013/01/02/the-ginger-punk-complete-male-avatar-update/). 2 January 2013 [April 2013].

[20] OpenSimulator. “OpenSimulator Avatar”. Internet: http://opensimulator.org/wiki/OpenSimulator_Avatar. 4 March 2012 [April 2013].

[21] NEXGEN Ergonomics. “HumanCAD® 2”. Internet: http://www.nexgenergo.com/ergonomics/humancad2.html. [April 2013].

[22] MA&D, Alion Science and Technology. “Integrated Performance Modelling Environment (IPME)”. Internet: http://www.maad.com/index.pl/ipme. [April 2013].

[23] Curtis Coates, Rui Zhang, Chris Cooper. “Crewing Effectiveness Modeling Proof of Concept”. Esterline|CMC Electronics Inc. on behalf of Department of National Defence. Report Number: DRDC-TORONTO-CR-2008-079 . 31 March 2008.

[24] OpenSimulator. “IRegionModule”. Internet: http://opensimulator.org/wiki/IRegionModule. 4 March 2012 [April 2013].

[25] OpenSimulator. “Getting Started with Region Modules”. Internet: http://opensimulator.org/wiki/Getting_Started_with_Region_Modules. 4 March 2012 [April 2013].

[26] OpenSimulator. “Database:Documentation”. Internet: http://opensimulator.org/wiki/Database_Documentation. 29 May 2013 [September 2013].

[27] 3D Ship Model. Internet: http://archive3d.net/?a=download&id=2c6df1c9. [October 2013].

Annex A Firestorm Viewer Setup

The Firestorm viewer is one of the few available viewers that can both import and view mesh models. However, the configuration of the viewer to enable log-in to a private grid can potentially be more complicated due to a bug which prevents a private grid operator from adding their grid IP address to the grid list. The following procedure serves as a guide to manually add the grid operator's private grid IP address.

1. Download the Firestorm viewer from the following location (http://wiki.phoenixviewer.com/downloads#current_release_-_opensim_build). Note that there are several viewers are available: select the OpenSimulator build.
2. Install and run Firestorm (also launch ROBUST and OpenSim instances).
3. To login into a private grid:
    1. *Select Viewer* > *Preferences* from upper left corner of main GUI window
    2. Under the *Advanced* tab check the box next to "Allow login to other grids…"
    3. Select the *OpenSim* tab
    4. Attempt to enter your IP address, omitting the "http://" and click *Apply*. If this fails to add your IP address, skip to step 6 otherwise, click *Apply* and *OK*.
    5. Select the new grid from the *Log onto Grid* drop-down menu. Enter your credentials and login. The *Username* is your account First and Last name separated by a space.
    6. If the grid selector does not add your IP address after pressing *Apply*, you will have to manually edit the grid user XML file as Firestorm has mistakenly associated your IP address with another grid (most likely Metropolis/Hippo). This is a known bug. To fix it:
        a. Go to C:\Users*username*\AppData\Roaming\Firestorm\user_settings.
        b. Open grids.user.xml using a text editor.
        c. Locate your grid IP address in the list. This address has most likely been associated with a login URI for another grid (look for the string under *loginuri*) and therefore could not add it again.
        d. Change the string under *loginuri* from the incorrect http address to your IP address. Also change the string under *gridname* to the name of your grid.
        e. Save the XML file and re-launch Firestorm. Your grid should appear in the grid list. Select it and login as described above.

List of symbols/abbreviations/acronyms/initialisms

3DS Max	3D Studio Max
API	Application Programming Interface
DEM	Digital Elevation Model
DES	Discrete-Event Simulation
DIS	Distributed Interactive Simulation (IEEE 1278)
DND	Department of National Defence
DRDC	Defence Research & Development Canada
DRDKIM	Director Research and Development Knowledge and Information Management
DTED	Digital Terrain Elevation Data
HLA	High-Level Architecture (IEEE 1516)
HTTP	Hypertext Transfer Protocol
IAR	(OpenSimulator) Inventory Archive
IEEE	Institute of Electrical and Electronic Engineers
IES	Integrated Enterprise Solutions
IPME	Integrated Performance Modeling Environment
OAR	OpenSimulator Archive
R&D	Research & Development
ROBUST	Redesigned OpenSimulator Basic Universal Server Technology
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VBS2	Virtual Battlespace 2
VMAP	Vector Map
vWorld	Virtual World