Air Traffic Controller
Cyber Attack Evaluation Serious Game
(ACES)

ACES Software Design Document

Spring Semester 2014
OR/SYST 699 Capstone Project
George Mason University
Fairfax, Virginia

THIS PAGE INTENTIONALLY LEFT BLANK

SIGNATURE PAGE

Submitted by	______________________ Doran Cavett, SOER Team MSSE Candidate	Date:	__________
Submitted by	______________________ Imran Shah, SEOR Team MSOR Candidate	Date:	__________
Submitted by	______________________ Wilbert C. Fontan, P.E., SEOR Team MSSE Candidate	Date:	__________
Concurred by	______________________ Paulo CG Costa, PhD Sponsor, GMU C4I Center	Date:	__________
Concurred by	______________________ Christopher Ondrus Sponsor, GMU Simulation & Game Institute	Date:	__________
Approved by	______________________ Kathryn Blackmond Laskey, PhD Professor, OR/SYST 699	Date:	__________

Executive Summary

The security and economic prosperity of a nation depend on critical infrastructure that is increasingly at risk from a variety of hazards, including cyber-attacks. Increasing connectivity and automation of critical infrastructure components and processes results in vulnerability to attacks on the cyber-infrastructure supporting our automated processes. This creates an opening for hostile actors to disrupt society through cyber-attacks. Progressively more, the cyber domain is seen as a new dimension of warfare — one that is especially open to lightweight, agile actors who do not require resources for major hardware investments and can operate from remote locations to disrupt our critical infrastructure.

The security and resilience of these assets, systems, networks, and functions — whether physical or cyber — requires a partnership that involves individuals and communities, businesses and non-profits, schools and universities, and governments at all levels, as well as a clear understanding of the risks we face. Only together, they can build a better understanding of the potential mission impacts of hostile cyber operations, better processes for planning for and rapidly responding to cyber threats, and better ways to assess both the impact of cyber operations and the effectiveness of their responses.

Serious games provide a means to evaluate cyber-attacks against critical infrastructure without the need for large investments in real world test scenarios and harm or loss of life. The George Mason University (GMU) Systems Engineering / Operations Research (SE/OR) candidates' team will guide a cadre of GMU undergraduate Simulation and Gaming (SGI) design students in designing and developing a serious game based on the contents of this document.

The Air Traffic Controller Cyber-attack Evaluation Serious (ACES) game will simulate cyber-attacks onto the Air Traffic Management (ATM) system used to conduct offshore helicopter operations in support of oil production off the Rio de Janeiro coast of Brazil. ACES will provide a venue for training Air Traffic Controllers (ATC) and understanding the impacts of cyber-attacks on ATM infrastructure and operations which will in turn help them identify and prepare effective mitigating actions.

This Software Design Document (SDD) is a standalone document that describes the architecture of ACES. The document will cover how to integrate Commercial Off the Shelf (COTS) software components that make up the ACES game beginning with the integration of VR-Forces simulation tool and the Unity game engine.

Table of Contents

Table of Figures

List of Tables

1. Purpose

1.1. General Description

The ACES game project is aimed at addressing the Air Traffic Management (ATM) risks encountered by air traffic controllers when challenged by a cyber-attack. It began as one of several potential GMU SE/OR capstone course research projects offered, aimed at putting learned OR/SE skills into practice.

Dr. Paulo Costa, GMU Associate Professor and a faculty member of GMU's C4I Center, provided the initial description of the operational need to be addressed. The operational needs established by his presentation, "Simulation-based Evaluation of the Impact of Cyber Actions on the Operational C2 Domain", set the foundation for the ACES game project.

The ACES game Software Design Document (SDD) is a standalone document that describes the architecture of the ACES game along with how to integrate ACES COTS. This document will be updated as ACES development continues in future semesters with a focus on integration of COTS software required to increase the realism of the game.

1.2. Mission

Unlike traditional games, serious games have an explicit and carefully thought-out educational purpose and are not intended primarily for amusement. Serious games can also be used to gain insights into the simulated operations and develop future planning based on what was learned ACES will provide a simulation of a real world situation and shall offer new experiences, insights, and knowledge to ATCs and observers, transforming learning into a more-engaging and dynamic process. Gameplay elements such as scoring and winning or losing are included to gauge a participant's progress with regards to established learning goals or objectives.

The operational concept described in this document is focused on cyber-attacks on helicopter operations in support of Maritime Oil Fields off the coast of Brazil. These offshore flights are often conducted at low altitudes and at distances beyond the range of any available mainland radar.

As a result, the safe and effective management of these offshore helicopter operations is then provided through the Automatic Dependent Surveillance-Broadcast (ADS-B) system.

ADS-B consists of two services ADS-B In and ADS-B Out. ADS-B allows for aircrafts and ground stations to receive ADS-B messages, air traffic messages, weather and terrain messages, ADS-B Out allows for aircraft to broadcast their identity, position, altitude, and speed to other aircraft and ground control stations. Aircrafts obtain their location from Global Positioning System (GPS) and broadcast ADS-B Out messages to other aircraft and relay or ground stations.

ADS-B communication is unencrypted and unauthenticated; anyone can listen to it and decode the transmissions from aircraft in real time. It does not make use of data level authentication of data from aircrafts; only checksums are used to verify integrity of a submitted message. ADS-B communications can be attacked through interception of messages, jamming of transmission, and injection of messages. A general description of these types of threats is shown below:

Type:	**Interception Attack**
Name:	Aircraft Reconnaissance
Description:	Intercepts and decodes ADS-B transmissions.
Purpose:	Target specific aircraft, gain knowledge about movement of assets and build an air order of battle, often the first step of a more insidious attack.
Target:	Aircraft
Technique:	Interception of ADS-B OUT signals
Difficulty:	Low

Type:	**Jamming Attack**
Name:	Ground Station Flood Denial
Description:	Disrupts the 1090MHz frequency at the ground station
Purpose:	Blocks all ADS-B signals intended for the ground station. Impact is localized to a small area determined by the range and proximity of the jamming signal to the ground station.
Target:	Aircraft and Air Traffic Controllers
Technique:	Jamming signal capable of disrupting the 1090MHz frequency range or GPS frequency
Difficulty:	Low

Type:	**Jamming Attack**
Name:	Aircraft Flood Denial
Description:	Disrupts the 1090MHz frequency for an aircraft
Purpose:	Blocks all ADS-B signals intended for an aircraft. Most significant impact involving this attack stems from gaining close proximity to an airport and affecting landing or taxi operations.
Target:	Aircraft
Technique:	Jamming signal capable of disrupting 1090MHz
Difficulty:	Medium

Type:	**Injection Attack**
Name:	Ground Station Target Ghost Inject
Description:	Injects an ADS-B signal into a ground station
Purpose:	Cause illegitimate (i.e., ghost) aircraft to appear on the ground controller's console.
Target:	Ground Station
Technique:	Inject message that conforms to ADS-B message protocol and mirrors legitimate traffic.
Difficulty:	Medium-High

Type:	**Injection Attack**
Name:	Aircraft Target Ghost Inject
Description:	Injects an ADS-B signal into an aircraft
Purpose:	Cause illegitimate (i.e., ghost) aircraft to appear on an aircraft's console.
Target:	Aircraft
Technique:	Inject message that conforms to ADS-B message protocol and mirrors legitimate traffic
Difficulty:	Medium-High

Type:	**Injection Attack**

Name:	Ground Station Multiple Ghost Inject
Description:	Injects ADS-B signals into a ground station
Purpose:	Overwhelm the surveillance system and create mass confusion for the ground controller
Target:	Ground Station
Technique:	Inject multiple messages that conform to ADS-B message protocol and mirrors legitimate traffic
Difficulty:	Medium-High

Current ABS-B vulnerabilities and their possible exploitation are of interest to a wider audience due to mandatory use of ADS-B in the United States by 2020 and in Europe by 2017. ADS-B is already in use in parts of North America, Europe, China, and Australia.

1.3. Operations

The operational setting for the ACES game is Brazil's Campos Basin where over 30 oil fields managed by large corporations such as, Petrobras, Esso, and Shell, are located. The Campos Basin region accounts for over 1 million barrels a day of petroleum production (80% of Brazil's petroleum production). Oil development operations in the Campos Basin include heavy helicopter traffic between the continent and oceanic fields during daytime, with an average of 50 minutes per flight.

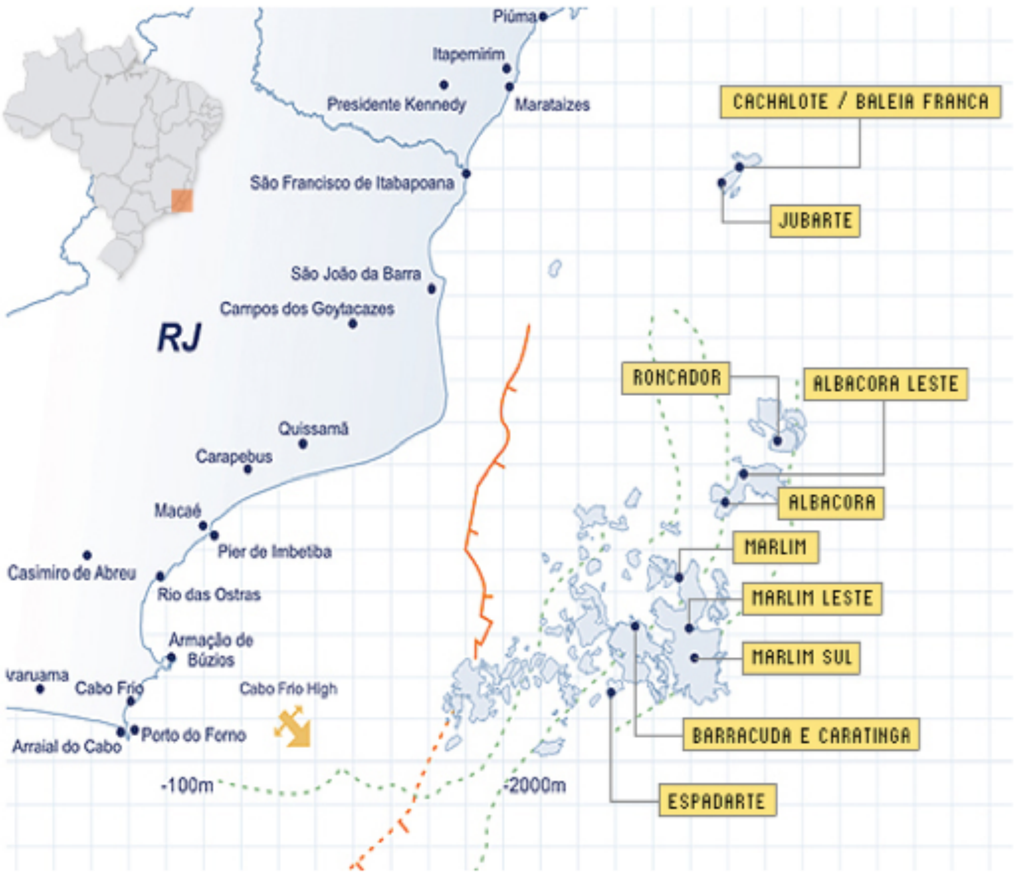


Figure 1- Overview of Campos Basin Oil Operations

Helicopter flights are conducted at low altitudes and oil platforms are located more than 60 nautical miles from the region's main airport, Macaé. As a result, helicopter operations cannot be monitored from the Macaé airport, the region's main airport, which only supports air traffic within a 45 nautical mile radius and 9500 foot and above altitude. ATM for these offshore helicopter operations is then provided through the ADS-B system.

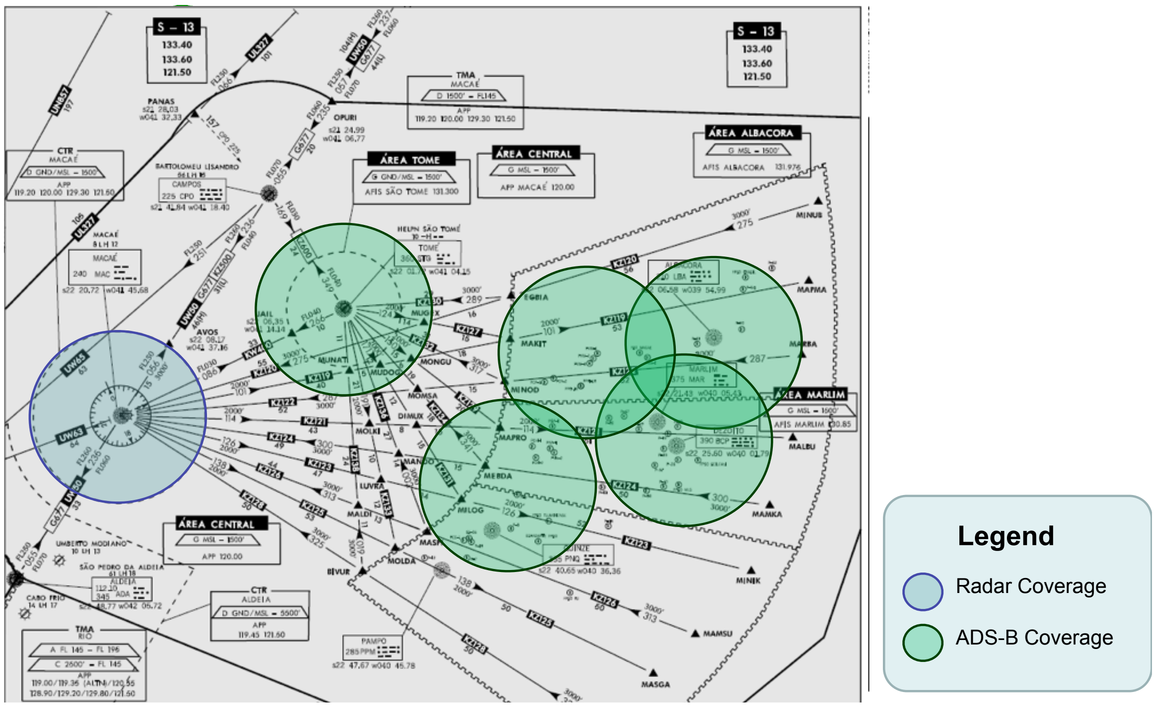


Figure 2 - Campos Basin Radar & ADS-B Coverage

Disruption to the Campos Basin helicopter operations will negatively impact and may even halt production at the oceanic fields. Safe and continuous operation of helicopters supporting offshore oil production is critical to meet production capabilities and protect against loss of life or assets.

ATC reliance on ADS-B may allow for hostile individuals to disrupt helicopter operations through various cyber-attacks. To better understand the potential mission impacts of cyber threats and to allow for the development of improved operational and risk management processes, ACES will simulate the real-time scenario, cyber-attacks, and their effects. As expected, ATC operating (or about to operate) in an ADS-B environment is the target player audience for the ACES game.

2. High Level Entities

ACES functionality is organized into five major subsystems as depicted below:

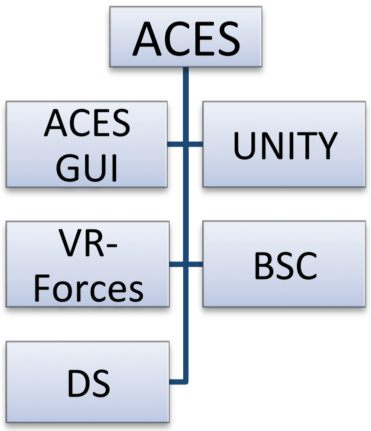


Figure 3 – ACES High Level Entities

2.1. ACES GUI

The ACES GUI Subsystem is responsible for providing a means for users to interact with the serious game. Every aspect of the game will need a GUI in order for a user to progress or influence the gameplay. The Simulation and Gaming Institute partners as well as the C4I Center at GMU are the major stakeholders of this subsystem.

2.2. Unity

Unity is a COTS game development engine, fully integrated with a complete set of intuitive tools and rapid workflows to create interactive 3D and 2D content.

Unity will be used to enhance the visual aspects of ACES, such as terrain and building structures inside of the game. VR-Forces interfaces with Unity in order to accept 3-dimensional (3D) model updates to the Geographical Information System (GIS) data that comes preloaded with the tool.

Once built, these enhanced 3D models inside of Unity will be integrated to VR-Forces and mapped to object instances so that the visual aspects of the game are appealing to the user.

2.3. VR-Forces

The VR-Forces simulation subsystem was used in previous iterations of the project to perform an Operations analysis on the flight paths of the helicopters and the amount of throughput they could perform. For purposes of the serious game the intent was to utilize the previously performed work and enhance it by turning it into a serious game.

VR-Forces is a Computer Generated Forces (CGF) application and toolkit. It provides an application with a Graphical User Interface (GUI) that displays a simulated environment. Users can create simulations using pre-defined or custom entities. VR-Forces and the simulations generated can be combined with other software to develop applications and configured for access through a separate interface. VR-Forces consists of a GUI and a backend simulation engine. Each is a separate application that communicates with each other through the use of a network.

To build the ACES game prototype the SE/OR-SGI team shall leverage work previous completed in a joint effort between the GMU C4I Center and the Technological Institute of Aeronautics in Brazil. The code that will be reused in ACES is a C++ simulation of helicopter operations in the Campos Basin region developed by Dr. Alexandre Barreto in the Fall of 2013. The C++ simulation code is run as a simulation scenario in the MAK VR-Forces simulation tool.

2.4. Attack Generation Code

The Cyber-Attack Simulation Subsystem is designed to be extensible for the incorporation of new attack types beyond the work performed in the Spring 2013 semester. The main focus for our group was the Injection Attack but there are other attacks available for simulating and building into the game. The Cyber-Attack Simulation will interface with the Unity game environment as well as VR-Forces to simulate the attacks that it constructs and sends over for placement into the serious game.

2.5. Data Storage

The ACES Data Storage Subsystem consists of two components; the Database for storage and quick recall of user profile information and the Data Store that contains the functionality to write game save information to a client machine.

3. Low Level Design

3.1. Unity

3.1.1. Assets

Assets are used to build Unity projects. Examples of Unity assets include graphics, 3D models, sound files, and files used to create a game.

3.1.2. Scenes

A scene portrays an individual level or area of a game. Use of more than one scene allows for distribution of game loading time and unit testing of the game.

3.1.3. Game Objects

A GameObject is a type of asset used in a game scene. GameObjects contain a Transform component that gives the Unity engine details on the position, rotation, and scale of an object.

3.1.4. Components

A component can create behaviors, define appearances, and influence an aspect of an object's function. Scripts are used to build interactive elements of a game and are an example of components in Unity.

3.1.5. Scripts

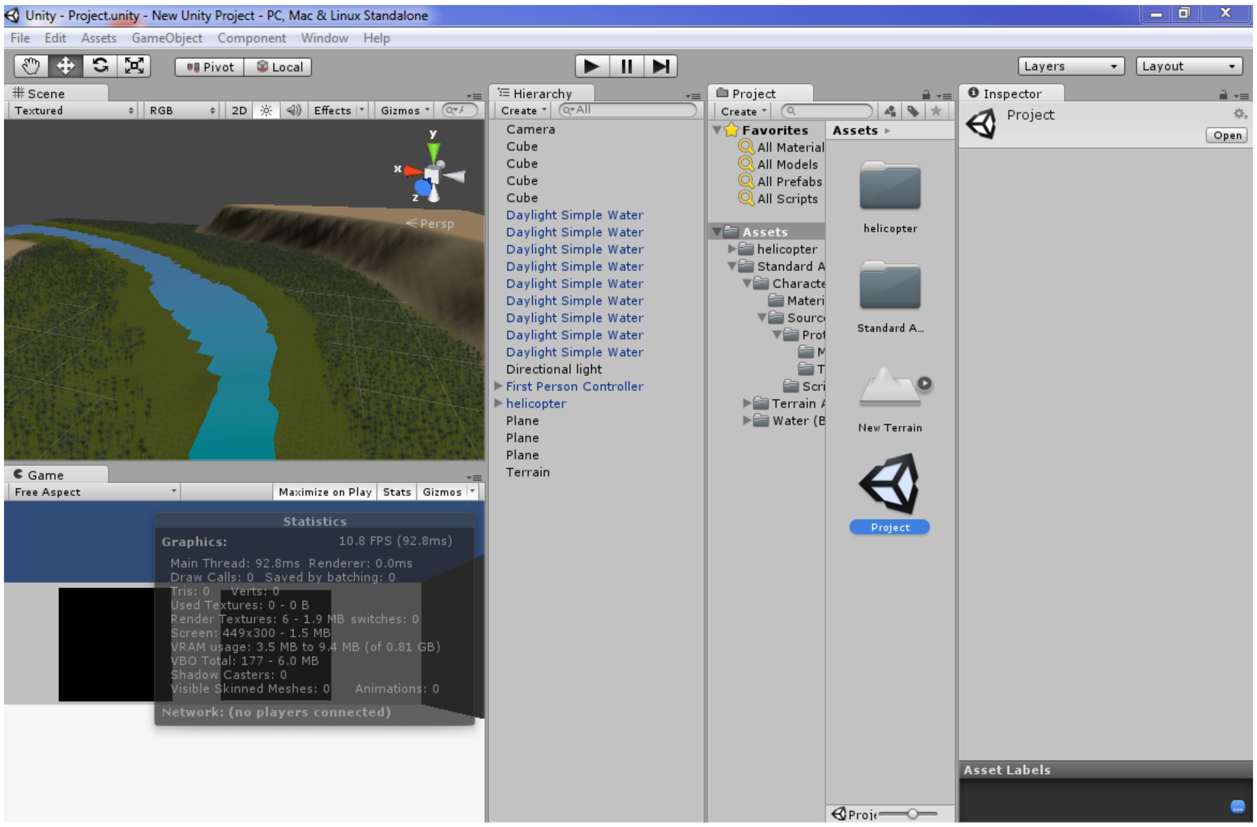
Scripts can be written in JavaScript, C#, or Boo. Unity provides a script editor for use and a Behavior class for developer use.

3.1.6. Prefabs

Prefabs are stored assets that can be reused in different parts of the game.

3.1.7. Interface

The Unity interface has a customizable layout. The Unity interface with an early demo of ACES is show in the figure below.



```
Statistics
Graphics:                    10.8 FPS (92.8ms)
  Main Thread: 92.8ms  Renderer: 0.0ms
  Draw Calls: 0  Saved by batching: 0
  Tris: 0     Verts: 0
  Used Textures: 0 - 0 B
  Render Textures: 6 - 1.9 MB  switches: 0
  Screen: 449x300 - 1.5 MB
  VRAM usage: 3.5 MB to 9.4 MB (of 0.81 GB)
  VBO Total: 177 - 6.0 MB
  Shadow Casters: 0
  Visible Skinned Meshes: 0      Animations: 0
Network: (no players connected)
```

Figure 4 – Unity Interface

3.2. VR-Forces

3.2.1. Applications

The GUI loads a VR-Forces scenario and terrain, displays the simulated entities, and executes the simulation. The back-end also loads the scenario and terrain, creates object, and executes the scripted plans. Scenario messages, simulation messages, and state updates are sent between the GUI and back-end via network communication.

3.2.2. Perspective

Simulations can be viewed either in two or three dimensions. Three dimensional views require additional computing resources to allow for rendering of the simulation.

3.2.3. Simulation Standards

VR-Forces can work with both the Distributed Interactive Simulation (DIS) and High Level Architecture (HLA) simulation standards.

3.2.4. Entities

Entities in VR-Forces can be assigned to complete one or more tasks that can be conducted independently of other entities or based on the actions of other entities.

3.2.5. Interface

The figure below shows the VR-Forces interface while a simulation of helicopter operations in support of oil development in the Campos Basin region is loaded.

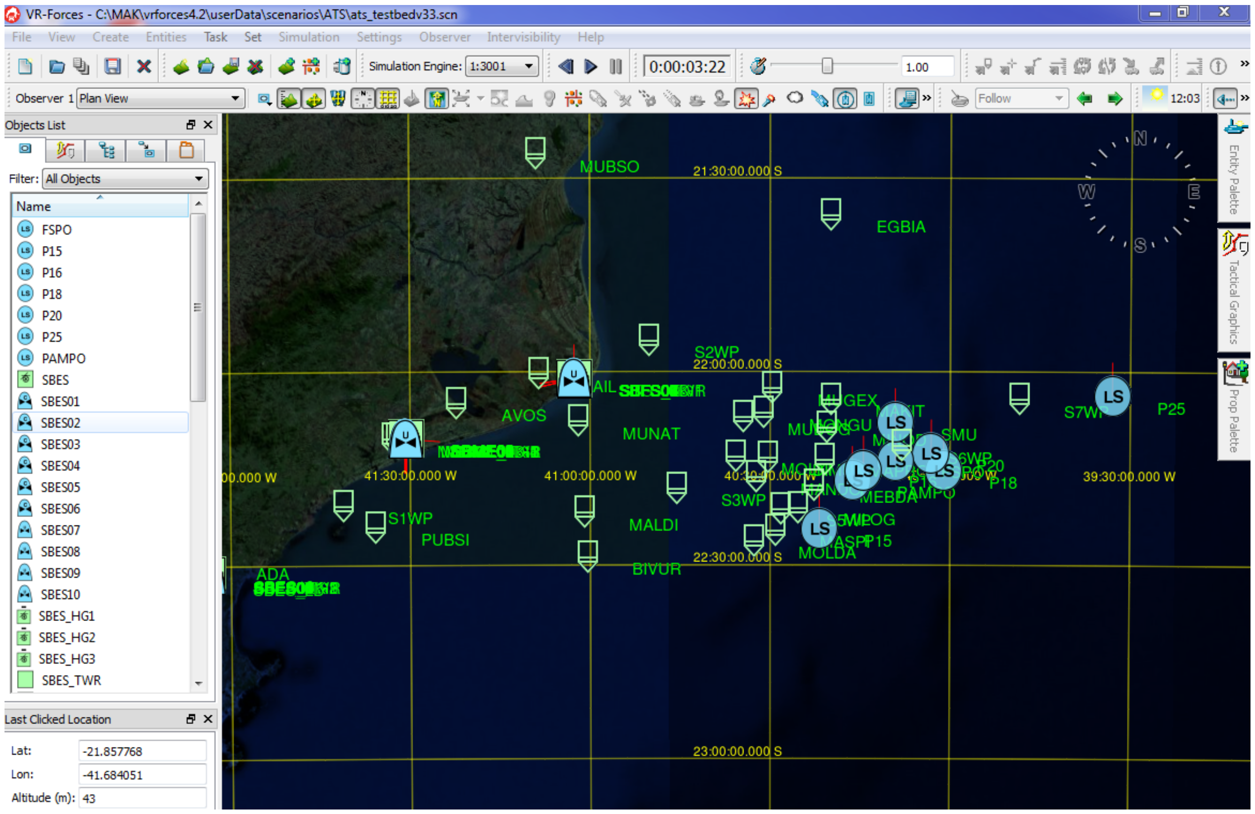


Figure 5 - VR-Forces Interface

3.3. VR-Link for Unity

VR-Link implements the HLA and the DIS protocol. VR-Link for Unity is used to communicate between VR-Forces and Unity.

3.3.1. Layers

VR-Link for Unity is composed of the GameLinkCS and GameLink layers. GameLink is a C++ library that adds Unity coordinate conversions and management of entities on top of the VR-Link toolkit. GameLinkCS is a set of C# bindings that allows Unity to communicate with VR-Link and GameLink using languages such as C# or JavaScript.

3.3.2. GameLinkCS and GameLink

GameLinkCS layer passes messages between Unity and VR- Link:

- From VR-Forces: GameLinkCS receives messages from VR-Link and stores the state in Unity game objects. Objects are moved based on dead reckoning, smoothing, and ground clamping provided by VR-Link.

- From Unity: Unity objects can be published to VR-Link. VR-Link is then able to publish the object to the DIS/HLA simulation network.

GameLink and GameLinkCS are depicted in the figure below:

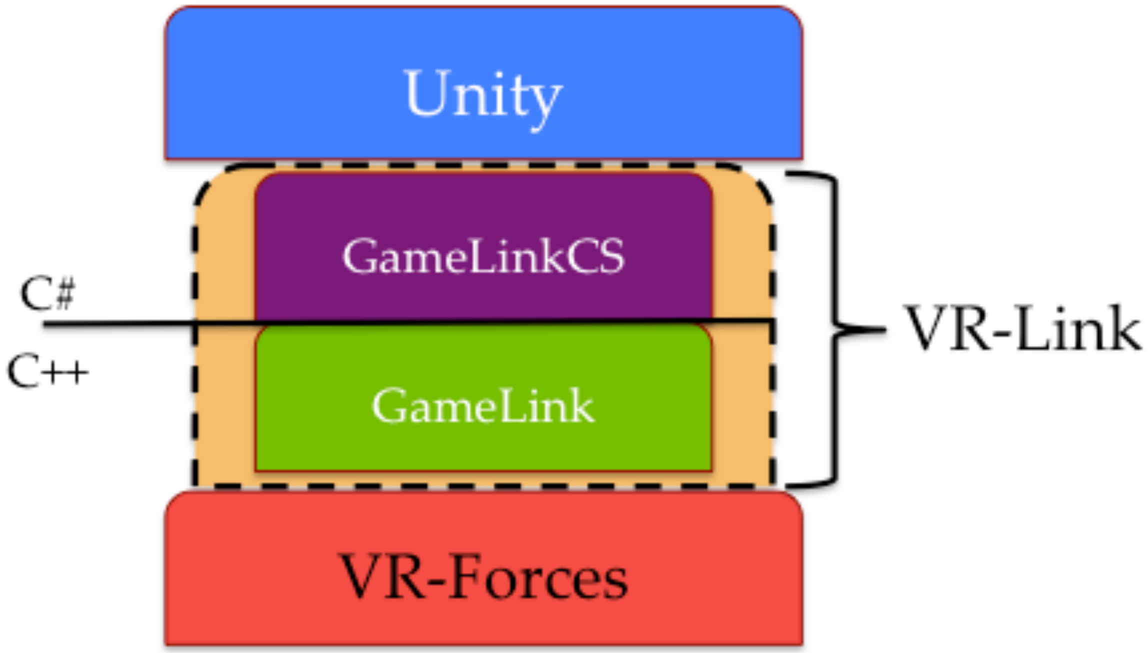


Figure 6 - VR-Link GameLinkCS

3.4. Interactions

The figure below shows how the ACES GUI, Unity, MAK VR-Forces, Attack Generation code, and the data storage interact and share data.

The ACES GUI displays GIS data mapped to 3D entities and GIS data to the end user's display. Account data is also displayed via the GUI. Users input commands to manipulate the game (i.e. detect and respond to cyber attacks) and login/account data.

The Unity game receives GIS data from MAK VR-Forces simulation and user account data from the data storage. GIS data mapped to 3D entities is then displayed to end users via Unity.

MAK VR-Forces is responsible for conducting simulation of helicopter operations. Cyber attacks against ADS-B communication are fed into the simulation from the attack generation code and scripts stored in the data storage. MAK VR-Forces stores and retrieves simulation data by means of the data storage. GIS data is sent to Unity for eventual display to end users. End user commands are able to modify execution of the simulation.

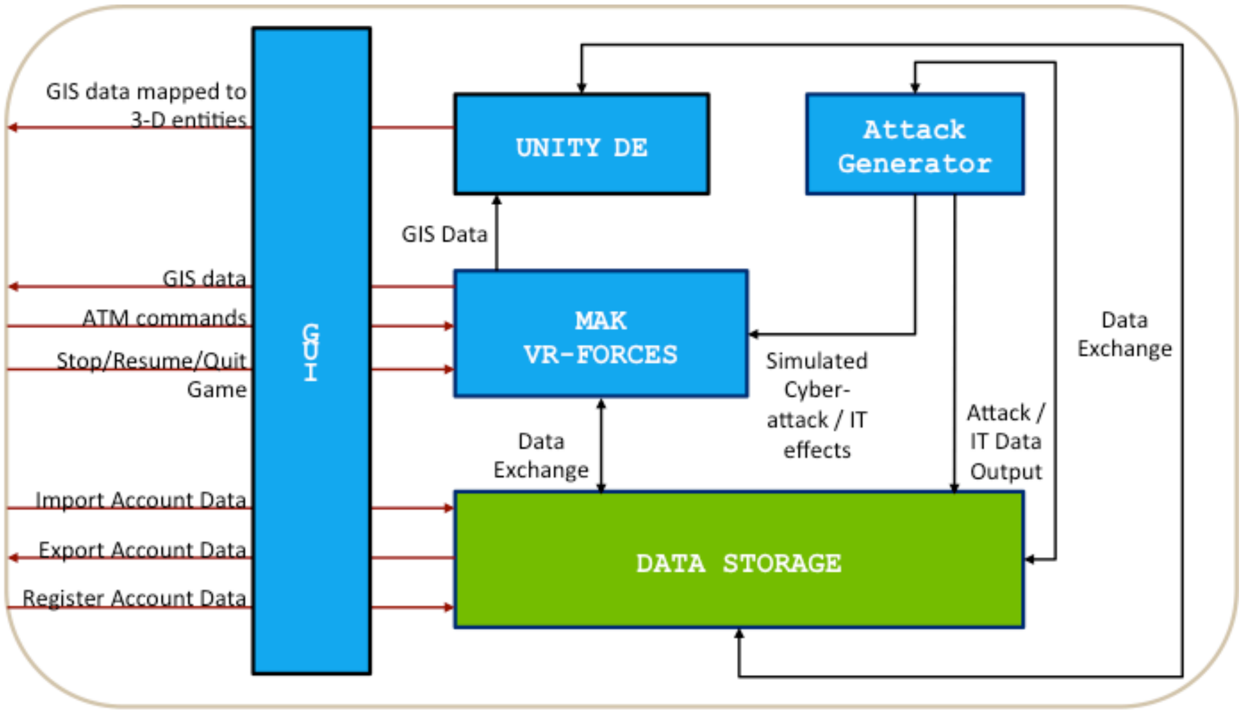


Figure 7 – ACES Interactions

4. Benefits, Assumptions, Risks, Issues

4.1. Benefits

4.1.1. Reuse

The proposed design leverages existing simulation code of helicopter operations developed for Dr. Barreto's PhD thesis and allows for building upon the code for enhancements.

4.1.2. Accuracy

VR-Forces is a powerful tool that allows for simulation of helicopter flights that will accurately model interactions with assets and impact to operations. The use of VR-Forces enhances the accuracy of the game scoring and analysis of impact.

4.1.3. Realism

The use of Unity for game design and 3D gameplay provides a realistic environment that allows a player to learn in a setting that mimics the ATC experience.

4.1.4. Training

Through use of data storage the player will be able to continue to play and improve score and learning over time.

4.2. Assumptions

4.2.1. Licensing

Unity and VR Forces licenses will be available for future development of ACES.

4.2.2. Domain Knowledge

ACES game users responsible for completing the programmable script that allows for of cyber-attacks have appropriate knowledge to configure realistic ADS-B cyber attacks.

4.3. Risks

4.3.1. Support

Continued support for all COTS tools from the vendors will be available especially for VR-Link, which is required to allow Unity to interact with VR-Forces. Support is also required for development of the Attack Generation Code.

4.3.2. Skillset

Development of ACES is dependent on a team with skillsets in the following tools and languages:

- VR-Forces: Familiarity with the VR-Forces tool, configuring and manipulating simulations
- Unity: Familiarity with designing and manipulating a game in Unity
- C#/C++/Programming Languages: Programming skills are required for maintenance and update of attack generation code and manipulation of behaviors in Unity beyond built in actions

4.3.3. Attack Generation Code

Attacks generated must be realistic to provide for meaningful training and analysis of impact to critical infrastructure and helicopter operations.

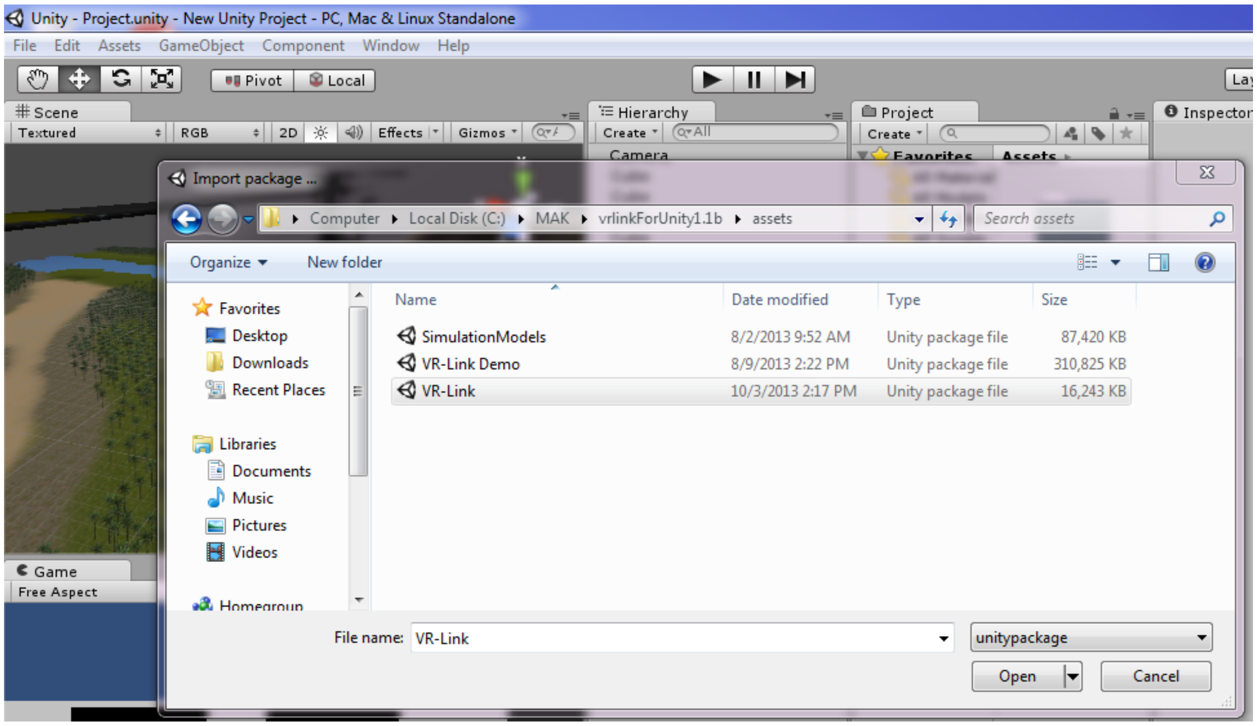
4.4. Issues

4.4.1. Integration of Unity and VR-Forces

VR-Forces entities are not capable of direct manipulation from Unity. The steps required to integrate VR-Forces and Unity are described below along with explanation of a workaround for manipulation of VR-Forces entities from Unity.

4.4.2. Import VR-Link Package

Importing the VR-Link package into Unity will add the VR-Link directory to the Unity Assets folder. The VR-Link folder contains C# scripts for interfacing with the C++ layer as well as the classes for managing the simulated entity behaviors and terrain referencing. It will also add the VR-Link DLLs and gamelink subfolder to the Plugins directory. These are the C++ libraries that are used. The figure below shows how to import the VR-Link package.


Unity - Project.unity - New Unity Project - PC, Mac & Linux Standalone
File Edit Assets GameObject Component Window Help
Pivot Local
Scene Hierarchy Project Inspector
Textured RGB 2D Effects Gizmos
Create
Game
Free Aspect
Import package ...
Computer ▸ Local Disk (C:) ▸ MAK ▸ vrlinkForUnity1.1b ▸ assets
Search assets
Organize New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Homegroup
Open
Cancel

Name	Date modified	Type	Size
SimulationModels	8/2/2013 9:52 AM	Unity package file	87,420 KB
VR-Link Demo	8/9/2013 2:22 PM	Unity package file	310,825 KB
VR-Link	10/3/2013 2:17 PM	Unity package file	16,243 KB

File name: VR-Link

unitypackage

Figure 8 – Importing VR-Link Package

4.4.3. Model Mapping

Following import of the VR-Link package the Model Mapper (may appear as EntityMapper in Unity) asset is available for creation. The Model Mapper object is used to map DIS enumerations to a Unity Prefab object. A discovery for an entity type will create a new object in the game based on the template. The figure below shows how to create a ModelMapper/EntityMapper.

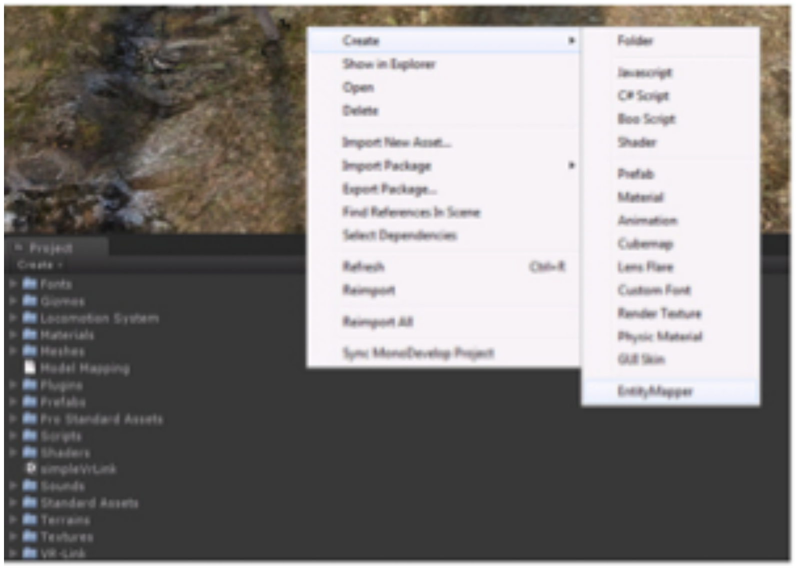
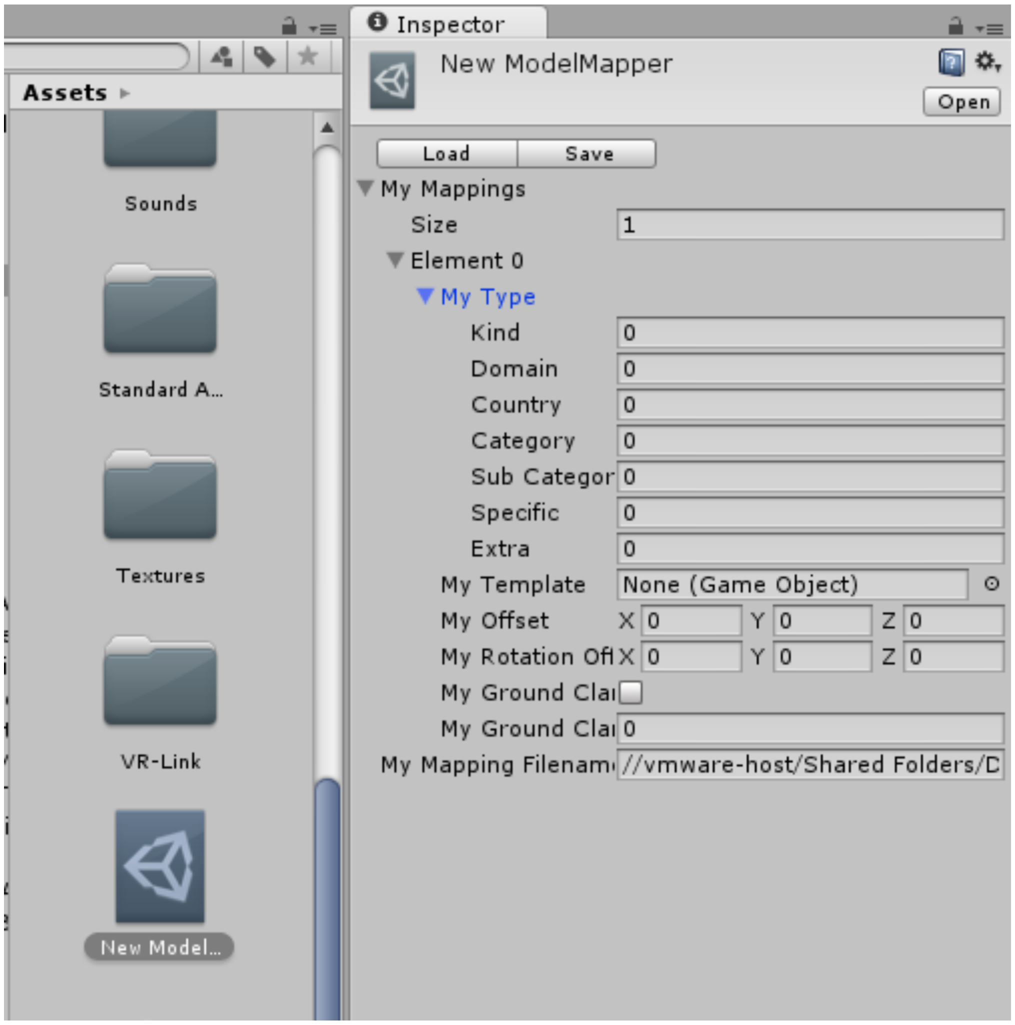


Figure 9 - Creating Unity ModelMapper

Details of the ModelMapper are shown in the figure below. The size field determines the number of entities mapped in the object as shown in the figure below.



New ModelMapper

My Mappings
Size 1
Element 0
My Type
Kind 0
Domain 0
Country 0
Category 0
Sub Categor 0
Specific 0
Extra 0
My Template None (Game Object)
My Offset X 0 Y 0 Z 0
My Rotation Off X 0 Y 0 Z 0
My Ground Cla ☐
My Ground Cla 0
My Mapping Filenam //vmware-host/Shared Folders/D

Figure 10 - Mapping VR-Forces Entity in Unity

4.4.4. Simulation Manager

Simulation manager provides the C# Application Programming Interface (API) to VR-Link libraries that allow for connections from Unity to a simulation and handles message dispatch and the creation of entities. A prefab Simulation Manager game object is provided by VR-Link for Unity. The figure below shows how to add the simulation manager behavior.

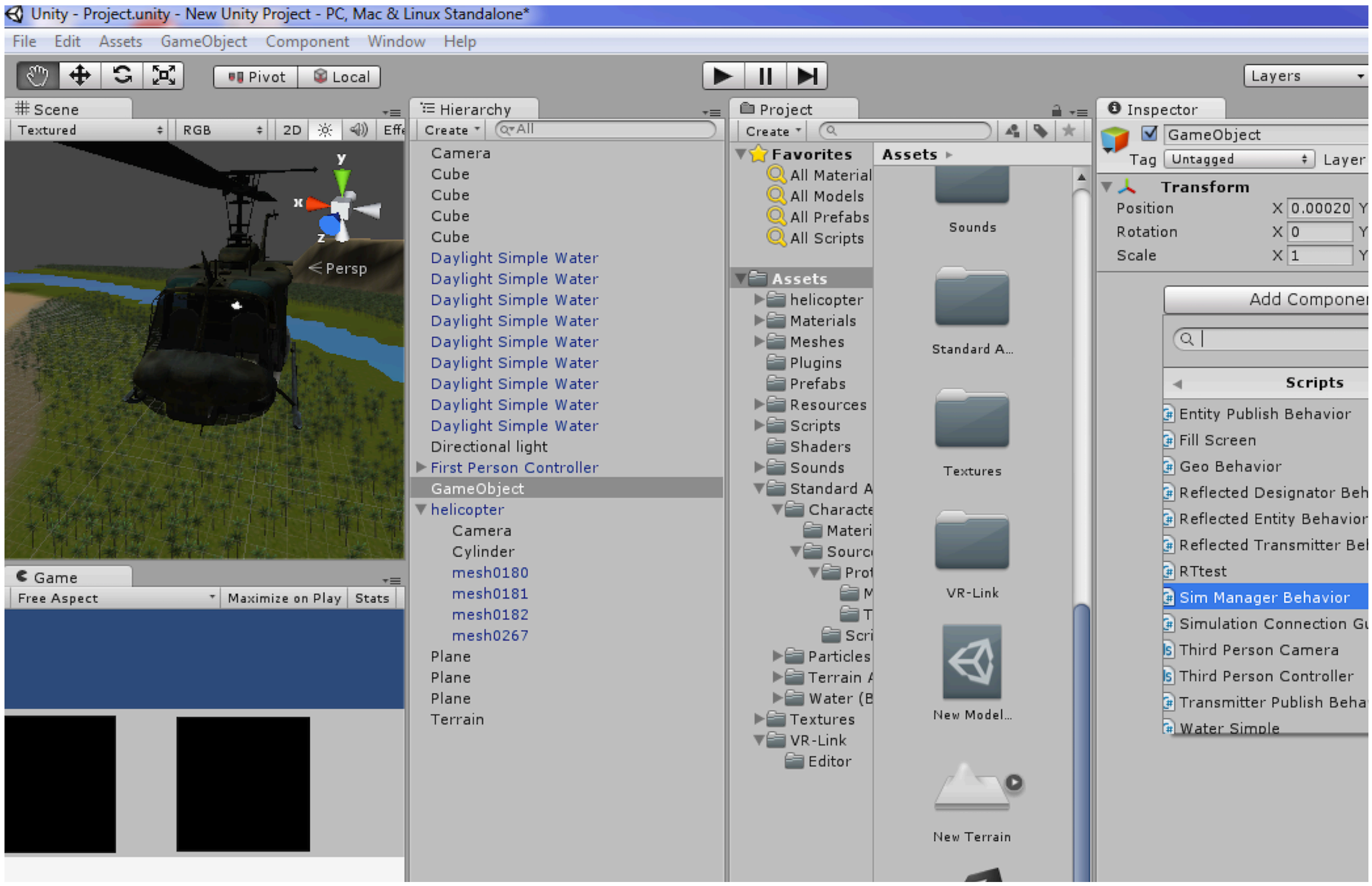
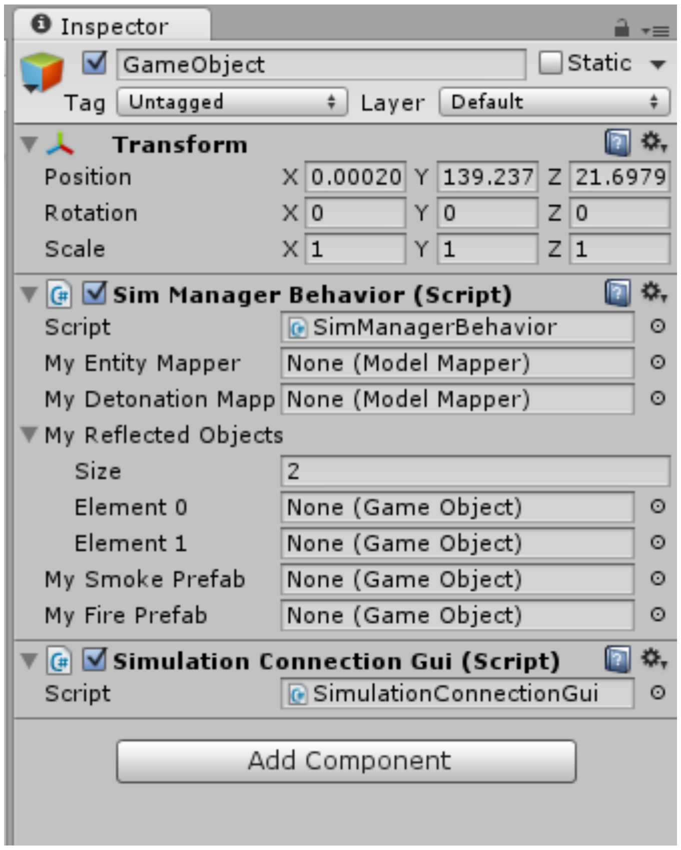


Figure 11 - Simulation Manager Behavior

Create a Simulation Manager by first creating a Unity empty Game Object. Add the “Sim Manager Behavior” script as shown in the figure above.

Model Mappers for entities and detonations are inputs to the Simulation Manager. These Model Mappers need to be dragged and dropped into the reflected objects section of the Sim Manager Behavior.


Inspector

GameObject — Static

Tag: Untagged — Layer: Default

Transform

Position	X	0.00020	Y	139.237	Z	21.6979
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Sim Manager Behavior (Script)

Script	SimManagerBehavior
My Entity Mapper	None (Model Mapper)
My Detonation Mapp	None (Model Mapper)
My Reflected Objects	
Size	2
Element 0	None (Game Object)
Element 1	None (Game Object)
My Smoke Prefab	None (Game Object)
My Fire Prefab	None (Game Object)

Simulation Connection Gui (Script)

Script	SimulationConnectionGui

Figure 12 - Input Model Mapper Reflected Entities

You will also need to add the "Simulation Connection Gui" script as shown in the figure above to allow for connections to the simulation from a GUI when the game starts.

4.4.5. Connections

Sending an InitConnection message through the Simulation Manager creates a simulation connection. There is also a public Connect() function available through the Simulation Manager to make connections. The Connect() function allows for DIS and HLA connections as long as the connection parameters are correctly set. Only one active connection is allowed at a time. Switching between DIS and HLA is possible without terminating the game. The shutdown() function disconnects from the simulation. The Game Object created to hold the Simulation Manager behavior can also be destroyed to achieve the same effect.

4.4.6. Objects

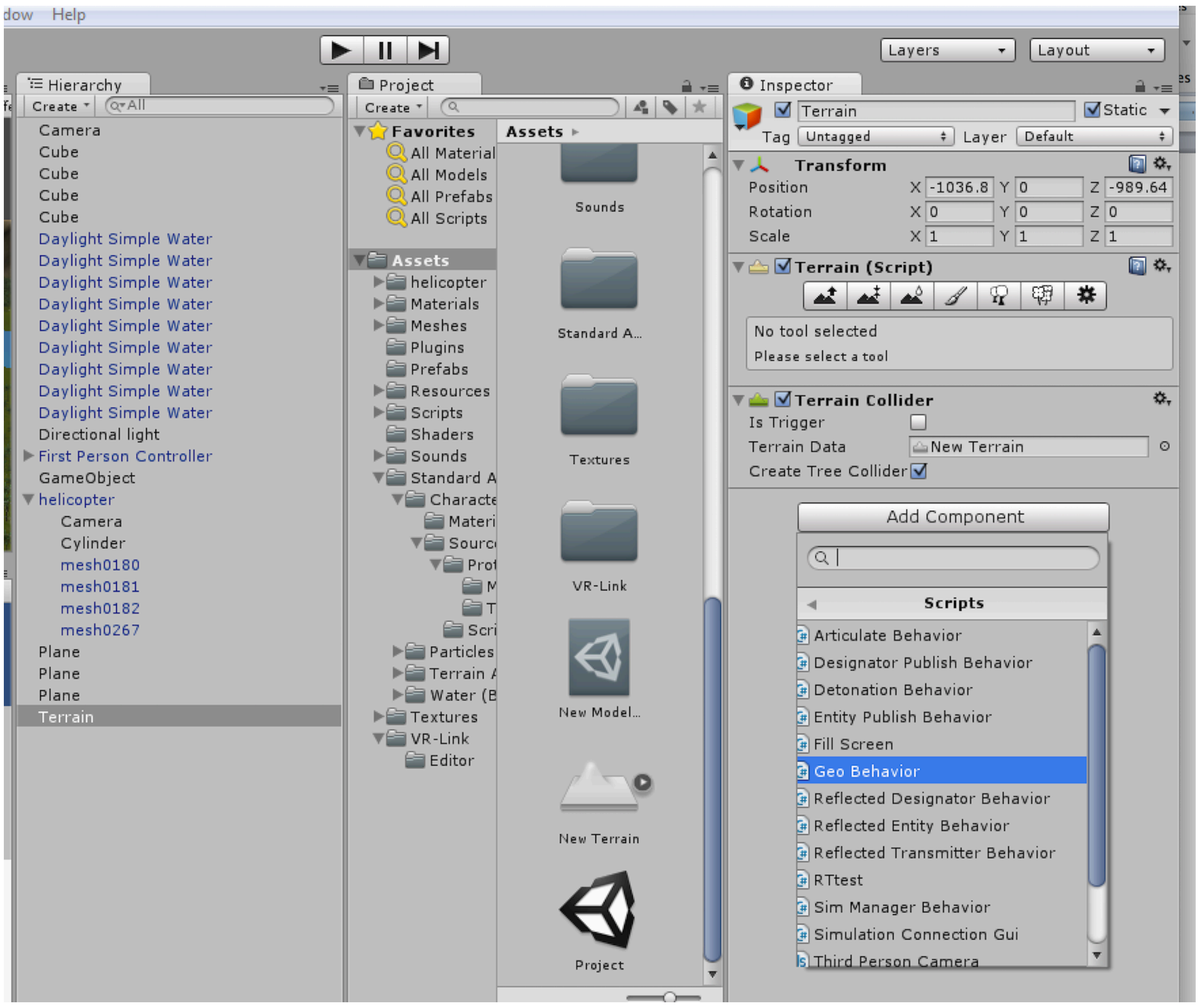
Behavior of the Simulation Manager is updated as the Unity game runs. Discovery of a new entity results in a lookup in the ModelMap for creation of a new GameObject according to the defined template. The GameObject is then configured to receive update messages from the VR-Forces mapped DIS/HLA entity.

4.4.7. Publishing

Attaching a Publish Behavior (script) to a Game Object allows for publishing Unity Game Objects as DIS/HLA entities.

4.4.8. Terrain

Attaching the Geo Behavior script is show in the figure below. The script allows for translation between the simulated and Unity coordinates.



Terrain ✓ Static
Tag Untagged Layer Default

Transform
Position X -1036.8 Y 0 Z -989.64
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

✓ Terrain (Script)
No tool selected
Please select a tool

✓ Terrain Collider
Is Trigger ☐
Terrain Data New Terrain
Create Tree Collider ✓

Figure 13 - Translate VR-Forces Terrain

The two coordinate options are TOPOGRAPHIC and PASSTHROUGH. TOPOGRAPHIC assumes Unity makes use of a projection of geographical terrain. PASSTHROUGH passes simulation coordinates to Unity without any transformation or conversion of coordinates.

4.4.9. Extensibility

The VR-Link for Unity C++ architecture allows for extension of capabilities through the use of plugins.

4.4.10. Interactions, Limitations

The table below captures the interactions required between VR-Forces and Unity. Limitations of VR-Link for Unity are explained along with potential workarounds.

VR-Forces and Unity Interaction				
Source	Destination	Data Exchanged	Desired Result	Feasibility
VR-Forces	Unity	Position of VR-Forces simulation entities	Display VR-Forces simulation entities in Unity game	Supported though VR-Link for Unity as described in sections 4.4.1 – 4.4.9 of this document
Unity	VR-Forces	Player interaction with VR-Forces simulation entities	Change in movement/operation of VR-Forces simulation entities	Unsupported directly. Workarounds discussed below the table
VR-Forces	Unity	Scoring: Landing of helicopters / Near accidents / Violation of helicopter operation rules (too high, too low, too close to others)	Provide data to allow for scoring of player	Captured purely in Unity and supported though VR-Link for Unity as described in sections 4.4.1 – 4.4.9 of this document

Table 1 - VR-Forces and Unity Interaction

4.4.11. Unity Player Interaction with VR-Forces Simulation Entities

4.4.11.1. VR-Forces Tasks

A task in VR-Forces can cause an entity to move to a location, patrol a route, follow an entity, take off / land, or fly to a location. VR-Forces allows for custom tasks to be written in the Lua scripting language.

4.4.11.2. Reactive Tasks in VR-Forces

A reactive task is only executed if a condition is met. The simulation is monitored and a reactive task is executed when a condition is fulfilled.

4.4.11.3. Suggested Approach

VR-Link for Unity doesn't directly allow for the manipulation of VR-Forces entities from Unity. However a task can be configured in VR-Forces that allows a VR-Forces entity to react to the behavior of another entity (the entity to which a VR-Forces entity reacts can be either in Unity of VR-Forces). In this configuration a VR-Forces entity, for example, can be configured to change directions of speed once a Unity entity enters an area or is within a certain distance of another entity. This indirect means of controlling a VR-Forces entity through Unity is suggested for implementation by future teams through a control panel interface for the ATC.

APPENDICES

A	Glossary of Terms (if necessary)
B	Acronym List
C	References

Appendix A: GLOSSARY OF TERMS

The glossary should define repetitive terms used throughout the document.

Appendix B: ACRONYM LIST

ACES	Air Traffic Controller Cyber Attack Evaluation Serious (Game)
ATC	Air Traffic Controller
ATM	Air Traffic Management
C4I	Command, Control, Communications, Computer, and Information
GMU	George Mason University
HW	Hardware
HELO	Helicopter
IA	Information Assurance
OA	Operational Assessment
OILPLAT	Oil Platform
OR	Operations Research
SE	Systems Engineering
SME	Subject Matter Expert
SOP	Standard Operating Procedure
T&E	Test & Evaluation

Appendix C: REFERENCES

“Simulation-based Evaluation of the Impact of Cyber Actions on the Operational C2 Domain”, Paulo C.G. Costa, Ph.D., Associate Professor, Department of Systems Engineering and Operations Research / C4I Center/ Center for Air Transportation Systems Research

“Automatic Dependent Surveillance-Broadcast (ADS-B) Out Performance Requirements To Support Air Traffic Control (ATC) Service”; OMB Approval of Information Collection, https://federalregister.gov/a/2010-19809

BRAZIL. ICA 100-12: Regras do Are Servicos de Trafego Aereo. Rio de Janeiro, Brazil, April 2009.

Cisco 2014 Annual Security Report

“Exploring Potential ADS-B Vulnerabilities in the FAA’s Nextgen Air Transportation System Graduate Research Project”, Air Force Institute of Technology, Donald L. McCallie, BS, MS Major, USAF, http://www.hsdl.org/?abstract&did=697737

http://www.radartutorial.euhttp://www.oig.dot.gov/sites/dot/files/ADS-B_Oct%202010.pdf
“Hackers + Airplanes No Good Can Come Of This”, Defcon 20, Brad “RenderMan” Haines, CISSP

Unity Game Development: Welcome to the 3D world http://www.packtpub.com/article/unity-game-development-welcome-to-3d-world Will Goldstone, September 2009

U.S. National Security Alliance, http://staysafeonline.org/

U.S. Multi-State Sharing and Analysis Center, http://msisac.cisecurity.org/

VR-Link for Unity Users Guide 2013 Revision VRU-1.1-1-130801

VR-Forces Users Guide 2013 Revision VRF-4.2-1-131022

VR-Forces Scenario Management Guide 2013 Revision VRF-4.2-18-131022