VR-Forces 73 Easting 2020 Report

For
Distributed Simulation Project Final Report
May 5th, 2017

Prepared by
Kevin Foster
Phillip Showers
Rui Wang

Prepared for
Dr. Mikel D. Petty
CS 595, Modeling and Simulation II
Computer Science Department
University of Alabama in Huntsville

1. Introduction

1.1. Objective

Project VR-Forces "73 Easting 2020" is a Distributed Simulation of historical Battle of 73 Easting. It is based on the VR-Forces "73 Easting" project conducted by Charles Collins, and Ryan Preston in Spring of 2016 [1]. The purpose of this project was to develop a separate, independent model that works with VR Forces over network using the DIS (Distributed Interactive Simulation) protocol to calculate the impacts of a company of T-14 Armata tanks on the outcome of the battle. DIS is a UDP protocol for distributed simulation developed for the military over the course of the past 22 years. We were also tasked with evaluating the addition of M1A3 variant Abrams tanks on the U.S. forces side and if feasible, implementing them in VR Forces.

1.2. Tasks

1.2.1. Locate and reactivate VR-Forces, VR-Vantage, and existing 73 Easting scenario.

1.2.2. Confirm or expand scenario to "validatable" scope.

1.2.3. Execute 30 runs, analyze results, calibrate; repeat as needed.

1.2.4. Using the capabilities of VR-Forces, add the new U.S. M1A3 Abrams, with realistic movement, sensor, and weapon performance.

1.2.5. Develop a separate model that (1) executes concurrently with VR-Forces, (2) communicates with VR-Forces via the DIS protocol, simulates up to a company (10) of Russian of T-14 Armata tanks, with realistic movement, sensor, and weapon performance.

1.2.6. Test the new tanks in the 73 Easting scenario.

1.2.7. Execute 30 runs, analyze results, compare to historical battle.

1.3. The VR Forces 73 Easting Simulation Scenario

Collins and Preston simulated three U.S. tank Troops in the VR Forces Scenario that we used as the origin for our scenario. In their Scenario, Iron, Eagle, and Ghost Troops' actions were simulated in their actions against several segments of Republican Guard tank division. The main purpose of Collins and Preston's work was to faithfully simulate the outcomes of the battle at 73 Easting, thus many Troop scripting decisions were made in support of this that pushed the bounds of VR Forces' basic unit definitions. Because our scenario is more speculative in nature, we attempted to err on the side of realism, rather than reproducing the conditions encountered on the ground. For example, in the original battle, very few Iraqi tanks actually hit any U.S. forces with their main guns.

Additionally, in the rare case of a hit, the M1s armor proved more than sufficient to stopping the penetrator rounds fired by Iraqi forces. Collins and Preston simulated this behavior by introducing a probability of hit (pHit) coefficient that caused the Iraqi tanks to miss at least nine out of ten times. In contrast, we removed the coefficient and changed the pHit to a range based random draw. This results in Iraqi tanks form the future having a much better pHit. Another divergence in design that we took from the original scenario was regarding to the simulation of all three U.S. troops. Since we only gave the Iraqis a company of ten tanks from the future, we replaced the original grouping of eight T-72s encountered by Eagle Troop after their encounter with the Iraqi Scouts with a company of ten T-14s. This allowed the Iraqis to focus fire and defend the larger body of much weaker and more poorly manned T-72s to the east. It was then decided that since the other two divisions of Iraqi tanks did not have the benefit of T-14s, it made little sense to simulate their destruction. As a result, we only simulate the part of the battle between Eagle Troop and the corresponding Iraqi forces, however this includes both the Tawakalna Battalion and the Coil Reserve. If we had left the Ghost and Iron Troops in the scenario, it would have skewed the results with no benefit.

This image gives a context for the images below showing the location of the T-14 Company in the line of battle.

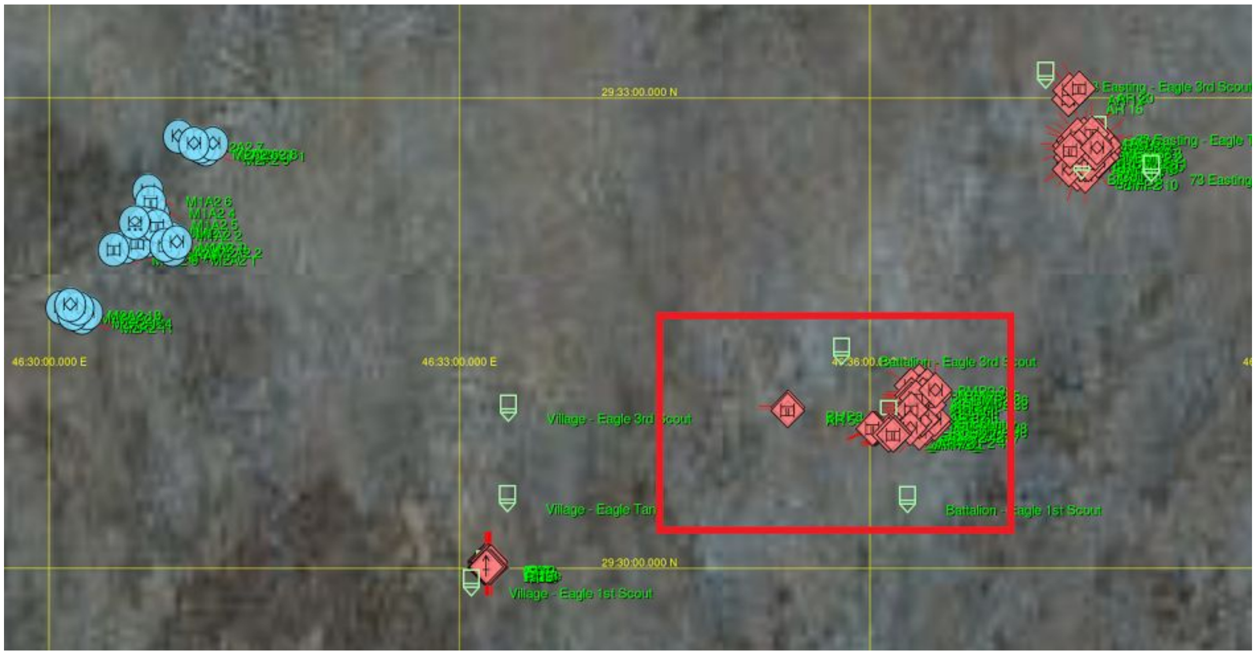

29:33:00.000 N
46:30:00.000 E
46:33:00.000 E
29:30:00.000 N
Village - Eagle 3rd Scout
Village - Eagle Tan
Village - Eagle 1st Scout
Battalion - Eagle 3rd Scout
Battalion - Eagle 1st Scout
73 Easting
M1A2 6
M1A2 4
M1A2 5

Figure 1 Exercise Screenshot (T-14 tanks are highlighted)

This image illustrates the location of the T-14 Company within the simulation. However it shows the T-14's at the end of a particularly hard battle the resulted in the loss of the Tawakalna Battalion.

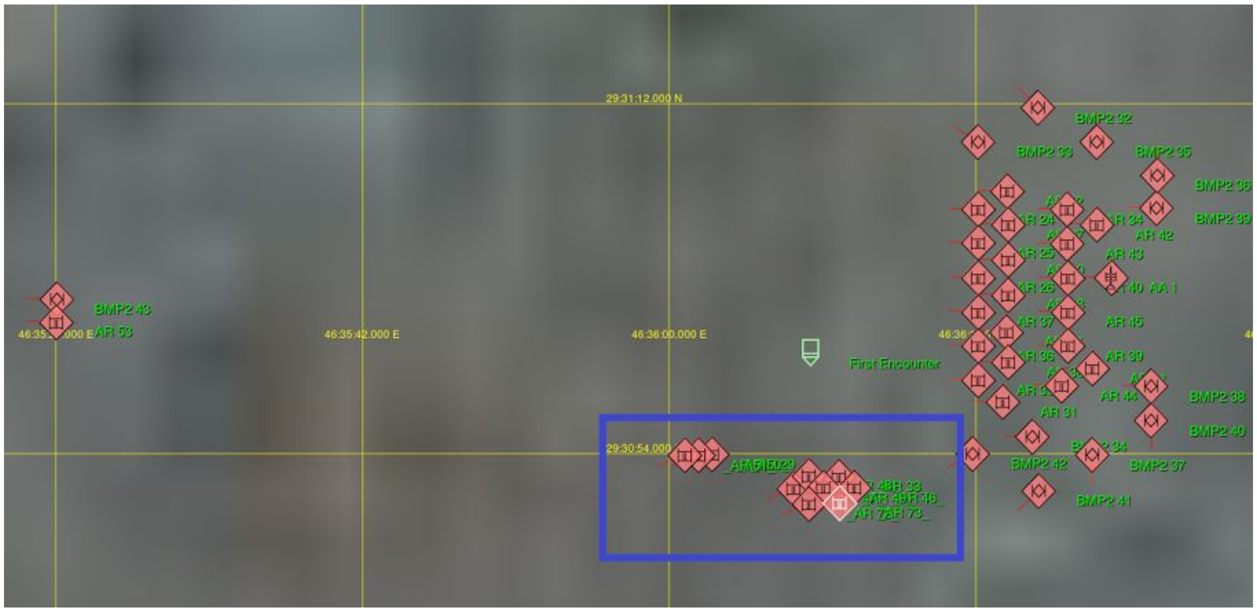


Figure 2 Exercise Screenshot (T-14 tanks are highlighted)

T-14 Company during engagement, here you can see red, and blue Fire PDUs being rendered by the display.

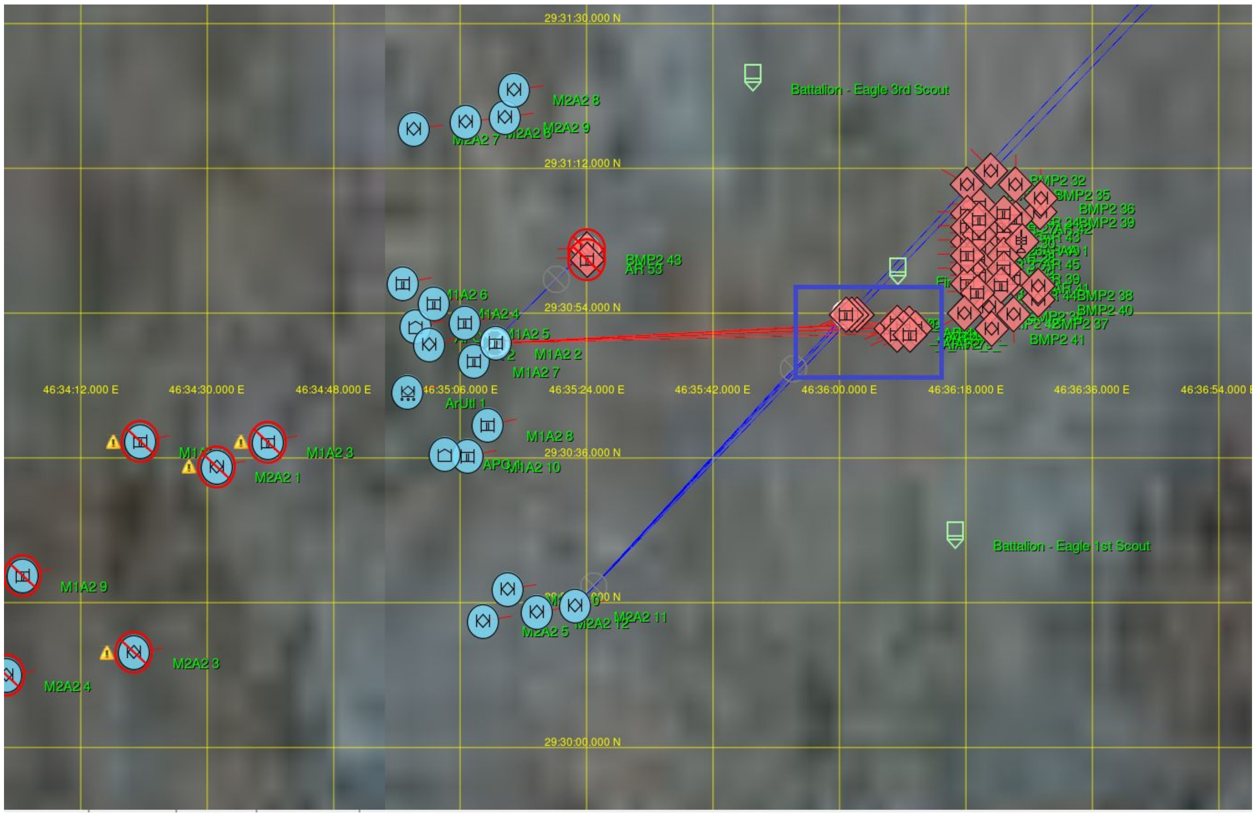


Figure 3 Exercise Screenshot (T-14 tanks are highlighted)

This is a screen capture of the aftermath of an engagement with the T-14 Company.

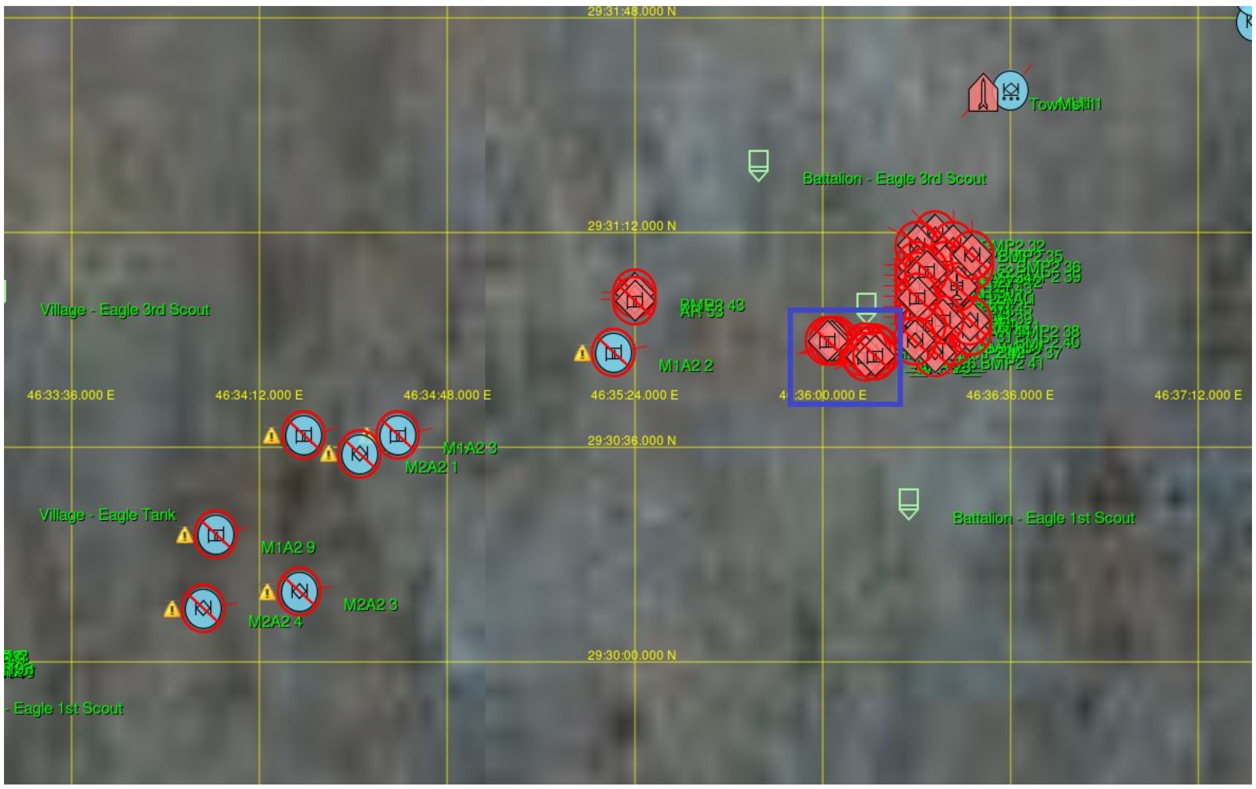


Figure 5 Exercise Screenshot (T-14 tanks are highlighted)

2. Background Information

2.1. Existing 73 Easting Scenario

VR-Forces "73 Easting" project conducted by Charles Collins, and Ryan Preston in 2016. The simulation follows Eagle Troop as they moved eastward to engage elements of the Tawakalna Division of the Iraqi Republican Guard. Per their report:

> As the 2nd ACR approached 68 Easting around 1525, Eagle Troop spotted a village complex, captured a small group of Iraqi soldiers, and received small arms fire from the complex's vicinity. Eagle Troop returned fire with 25mm cannon fire and a volley of 120mm rounds, silencing the enemy fire [2, pg. 133-136]. Suspecting more enemy positions to the east, Eagle Troop transitioned into a wedge formation in which tanks were on point and continued to advance.

> Approaching 73 Easting at 1618, Eagle Troop swung south and then north, encountering a battalion-sized unit of Iraqi armor. The Iraqi unit, mainly composed of T-72s and BMPs, was defending in depth on the reverse slope of a dune that Eagle had just crested. The Iraqi unit had been taken by surprise and was slow to respond, and the few shots fired by Iraqi vehicles missed. Eagle Troop annihilated the battalion and then began firing on a group of T-72s and BMPs which were left in reserve roughly 2 kilometers northeast of the battalion. These Iraqi forces were also annihilated. After these engagements, Eagle Troop halted on a rise and set up defensive positions near 73 Easting.

Our use of their scenario excluded Ghost and Iron Troops because they do not impact the course of the engagement with respect to the added T-14 tank company.

2.2. T-14 Armata Tank

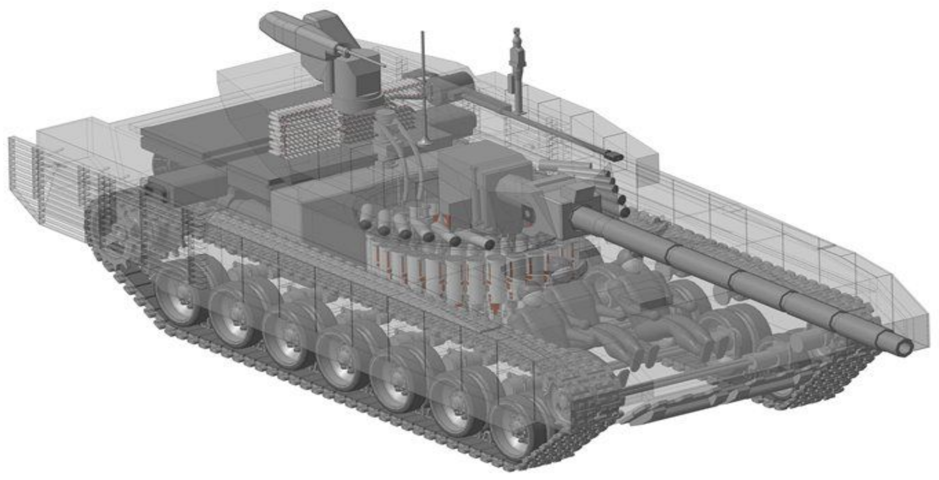


Figure 1 T-14 Armata[5]

We hypothesize that two main features of the T-14 would improve the prospects of the Iraqi forces. First, the advanced fire control system should bring the probability of hit of the T-14 into a range that was at least compatible with the M1A2. Related to this the turret in the T-14 is a powered turret, like that of the M-1, so that moving the guns to the a target will be much faster than a T-72 that is turned with hand cranks. Second, the targeting sensors on the T-14 are 0.9 meters higher than the M-1 family of vehicles, so they should be able to observe and engage M-1.

3. Model Architecture and Design

3.1. Architecture Description

This simulation model utilized multithreading strategy to solve the concurrency issue of sending Pdus and receiving Pdus at proper interval or whenever remote Pdus arrives. It used OpenDIS library to follow DIS protocol. We are also use VR Forces terrain database runs a VR Forces backend as terrain server.

3.1.1. T14 class

The T14 class is where we define the T-14 Armata tank within the simulation. The tank class includes data for building Entity State PDUs, including reporting entity id, location, velocity, orientation and the appearance of the tank to indicate its status. This class also performs target discovery, along with Terrain Server queries. Builds Fire PDUs to shoot at targets the calculated simulated rate. It also performs probability of hit (pHit) for targets that it shoots at and probability of kill (pKill) calculations to calculate the tank object's survival when it receives Detonation PDUs. The T-14 class also maintains a list of received Fire and Detonation PDUs that it regularly processes to react to incoming fire.

3.1.2. DataRepository class

This class maintains the remote entity "database" (hash table), the local entity "database", a dead reckoning "database". It is also the hub for communication for the simulation. When a PDU is received by the EsPduRecevier, it calls on the Data Repository to decide how where the PDU is processed. It is the key class that keep data consistency.

3.1.3. Simulation class

This is the class instantiates T-14 tank instances and denote all objects to initial their behaviors and update status. This is also where the main Update loop runs, calling the individual tank updates. A single update loop traversal consists of every tank instance having its specific update method called.

3.1.4. T14Sender class

This class uses a multicast socket to broadcast PDUs as it is called in a specific T14's update method.

3.1.5. EsPduRecevier class

This class runs in its own thread which receives PDUs and passes the data into the DataRepository class for dissemination to the T14 instances.

3.1.6. TerrainServerInterface class

This is the class is responsible for maintaining the TCP connection to the Terrain Server and querying data from the terrain database. It has two queries, an Elevation query, and a Visibility query. The Elevation query is used by the Simulation class at the T14's initialization step to correctly locate each T14 instance in the sim. The Visibility query is used to determine if a particular point on the map is occluded by the terrain from an origin point. This is used by the T14 class to determine realistic target selection. All data is passed via the WGS 84 map coordinate system as that is the standard coordinate system used in VR Forces and transmitted in the DIS data.

3.2. Architecture Diagram

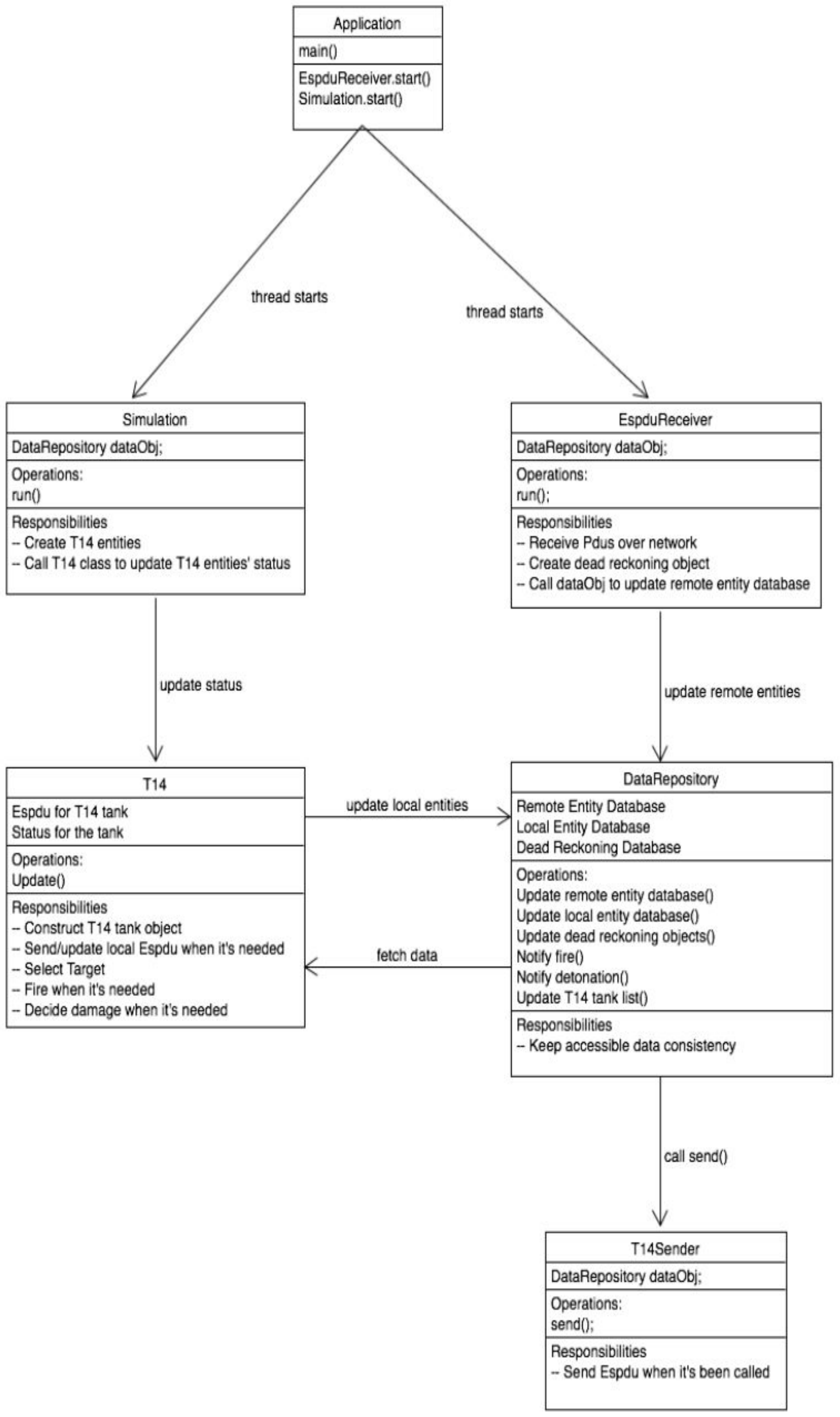


Figure 2 Architecture Diagram

4. Model Implementation

4.1. Application versions

- VR-Forces (4.4.2)
  This was the Simulation framework that ran the main scenario.
- Visual Studio 2013
  The Terrain Server was developed using MS Visual Studio 2013. Its use was mandated by
  the inclusion of necessary libraries provided with thanks by MAK.
- NetBeans
  Netbeans was used to build the modified OpenDIS library that our T14 Sim and the Data recorder used to build PDU objects for processing of data.
- Eclipse (Mars)
  The majority Java of development for the T14 Sim was done in Eclipse.
- Wireshark
  We used Wireshark to peruse the contents of 'live' PDUs sent by the VR Forces backends.

4.2. Terrain Server

Great Thanks go to MAK developers for their help with the development of this part of the project. Without their help, it is doubtful that we would be able to provide real terrain data, as shared by the VR Forces sim instances. Because of their assistance, the Terrain server wraps a duplicate of the VR Forces backend, thus allowing simulation initialization data to be sent to it as a part of the Simulation initialization. This approach mandates the use of a third VR Forces license, however MAK was more than generous with such requests in support of this study.

4.3. Development Timeline

February 14, 2017	Finished distributed simulation lectures.
March 12, 2017	Recreated VR-Forces "73 Easting" scenario by Charles Collins, and Ryan Preston. Started to implement the external simulation node with T-14 Armata.
April 2, 2017	Finalized design architecture. Successfully utilized OpenDIS Java library passing Entity State Pdus between Java program and VR-Forces.
April 8, 2017	Set up terrain server. Working on queries in terrain database. The external simulation node can create Tank object, sending and receiving Entity Pdus between old 73 Easting scenario in VR Forces. Implemented remote entity hash table and local entity hash table.
April 15, 2017	Working on integration of terrain database with our simulation node. Working on event inside of simulation like detecting enemy vehicles, firing, detonating. Utilized openDIS implemented dead reckoning.
April 22, 2017	Terrain server and terrain database integrated in our simulation system and working properly with queries during the simulation.
April 30, 2017	Finalized our simulation node with terrain server inside of VR Forces 73 Easting scenario with all desired functionalities.

5. Simulation Configuration and Results

Below are a summary of specific data regarding the scenario setup and the results of the runs.

5.1. T14 Tanks Placement

Table1 Iraqi T14 Armata Tank Placement

Index	**Entity Id**	**Position**		
		Latitude	**Longitude**	**Altitude**
303	AR 29	29.515003	46.600697	249
304	AR 33	29.514684	46.602762	249
305	AR 46	29.514522	46.603010	249
306	AR 47	29.514521	46.602031	249
307	AR 48	29.514702	46.602278	249
308	AR 49	29.514522	46.602515	249
309	AR 50	29.514999	46.600476	250
310	AR 51	29.514993	46.600259	250
311	AR 72	29.514299	46.602272	250
312	AR 73	29.514311	46.602773	250

5.2. Modifications to Simulated T14 Tank and VR Forces Tank

Table 2 T-72 125mm Gun Load and Unload Times

	Load Time (in seconds)	Unload Time (in seconds)
Initial Values	5	5
Final Values	15	15

Table 3 Probability of Hit Table for Tank 125mm Gun

Range(in meters)	T-72 pHit	T-14 pHit
1000	0.072	0.9
2000	0.032	0.8
3000	0.015	0.75
4000	0.007	0.7

Table 4 Probability of Kill Table for Tank Main Guns

Range(in meters)	Initial Probability of Kill	Final Probability of Kill
2000	0.8	0.8
3000	0.6	0.6
4000	0.3	0.3

We had to change the Pk table in VR Forces for both the American 120mm gun and the Russian 125mm guns to rectify the situation with the default VR Forces values where at some ranges the PK is greater for a h that initial Pk values are only effective when target distance are less than 1000 meters.

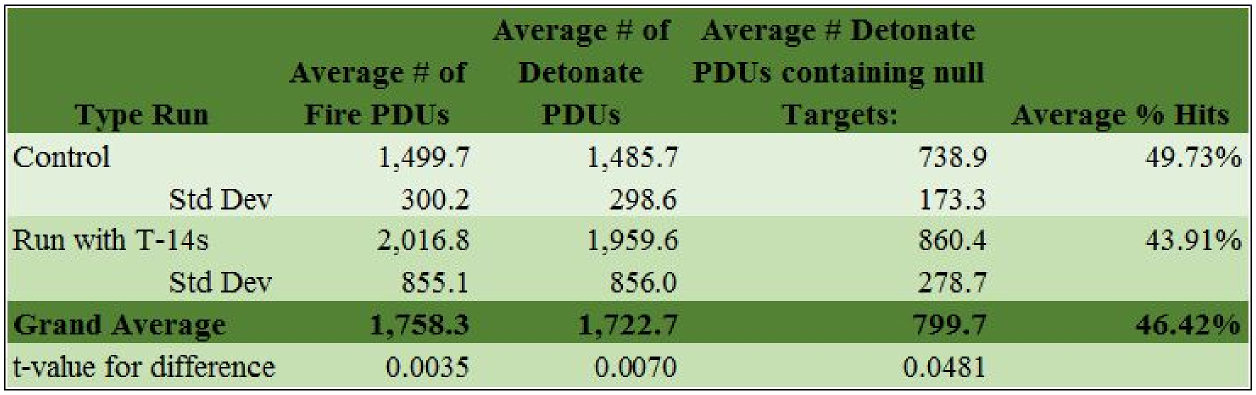
Table 4 Probability of Kill Table for Tank Gun

Range(in meters)	Initial Probability of Kill	Final Probability of Kill
1000	(catastrophic-kill -0.04 0.0 1.0)	0.95
2000	(catastrophic-kill -0.04 0.0 1.0)	0.90
3000	(catastrophic-kill -0.04 0.0 1.0)	0.80
4000	(catastrophic-kill -0.04 0.0 1.0)	0.70

5.3. Exercise Results

As we hypothesized the introduction of the T-14 significantly increased the performance of the Iraqi forces, although the fact that we only used 10 T-14s meant that the T-72 would still be take the brunt of the fighting. One effect that we would expect is the T-14 would increase the Iraqi volume of fire, because it has a superior fire control system that would allow it to identity and engage targets at a much greater range that the T-72. The fire control system would also allow for great accuracy. We ran 30 simulated runs using the scenario where the Iraqis had only T-72s as our control, then ran 30 trials where the Iraqi forces had 10 T-14 tanks. Table 5 below shows that the overall volume of fire increase by over ⅓ with the T-14s included, and the number of detonations hitting null targets--misses--decreased by nearly 6 percentage points.

Table 5 Total Difference in Volume of Fire and Accuracy



Type Run	Average # of Fire PDUs	Average # of Detonate PDUs	Average # Detonate PDUs containing null Targets:	Average % Hits
Control	1,499.7	1,485.7	738.9	49.73%
Std Dev	300.2	298.6	173.3	
Run with T-14s	2,016.8	1,959.6	860.4	43.91%
Std Dev	855.1	856.0	278.7	
Grand Average	**1,758.3**	**1,722.7**	**799.7**	**46.42%**
t-value for difference	0.0035	0.0070	0.0481	

A review of table 6 below shows that the introduction of the T-14 to the Iraqi forces accounts for the increase in the volume of fire and increase in accuracy. The difference in both categories is only statistically significant for the Iraqi tanks, indicating that just introducing 10 T-14 made a substantial difference in Iraqi capabilities.

Table 6 Difference in Volume of Fire and Accuracy by Country

Type Run	Average # USA Forces Fired	Average # USA Forces Hit	Average % Hits	Average # Iraqi Forces fired	Average # Iraqi Forces Hit	Average % Hits
Control	1,108.2	669.6	60.43%	342.1	55.3	16.15%
Std Dev	229.4	137.3		122.8	42.0	
Run with T-14s	1,197.9	785.3	65.55%	761.6	313.9	41.22%
Std Dev	827.0	581.8		89.4	48.4	
Grand Average	**1,153.1**	**727.5**		**551.9**	**184.6**	
t-value for difference	0.3031	0.2973		0.0000	0.0000	

Just adding 10 tanks significantly increased the losses on the US side, but ultimately the US forces still were able to destroy the Iraqi forces after the T-14s were taken out. In only 5 of 30 scenarios were the Iraqis able to stop Eagle Troop's advance. In only one of those battles were any of the T-14 able to survive. This is likely related to the fact that we had the T-14s deployed to the front of the Iraqi formation.

Table 7 Difference in Entities Destroyed by Country

Type Run	# of Blue Entities	Average of Blue Destroyed	Average % Destroyed	# of Red Entities	Average of Red Destroyed	Average % Destroyed
Control	26	13.3	51.15%	107	105.5	98.60%
Std Dev		8.6			5.8	
Run with T-14s	26	17.8	68.46%	107	94.9	88.69%
Std Dev		5.0			20.6	
Grand Average		**15.6**			**100.2**	
t-value for difference		0.0187			0.0108	

6. Challenges

6.1. Setting up VR Forces

Setting up and running the scenarios provided by Collins and Preston was relatively straightforward once the details were understood. We were provided with the simulation files for the scenarios they studied. We discovered that the VR Forces software starts in a more stable manner if the backend is started first, followed by the front end, rather than using the combined tool startup option. Additionally, it is of paramount importance to select unique application ids at the start time, otherwise the two backends will not communicate properly. Collins and Preston used 3001 and 3002 as their application ids, and we continued that for our use of their scenario.

6.2. OpenDIS Library and different version of DIS protocols

We used OpenDIS Java library[4]. The source code and tutorial of OpenDIS are not documented extensively for usage. We spend a good amount of time figuring out how to utilize the library. At the early stage of the project,we had a lot of troubles getting VR Forces understand our simulated tank. The very important set up is OpenDIS and VR Forces need to use the same version of DIS protocol(DIS 7.0).We also needed to run WireShark to capture DIS packets to learn how VR Forces builds Pdus. This enabled us to get more information about how to create our own Pdus in this model and learn how to set up all the parameters. OpenDIS uses a factory class to unmarshal the binary data into values within its class structure. Another important step in success with OpenDIS was writing a custom PDU Factory class capable of generating DIS 7 class objects and recompiling the openDIS jar file with it. This allowed us to start receiving DIS 7 messages from the VR Forces sim.

6.3. Terrain Server and Terrain Database

MAK offers a product that allows external applications to access their simulations' terrain data called InTerra. After lengthy discussions with their technical team, it was decided to abandon our attempt to use InTerra and instead build an application that wrapped the core code for the VR Forces backend itself. This allowed us to make queries directly to the terrain data downloaded by the backend at sim initialization time. A basic TCP/IP connection is made between the T14 Java Sim and our custom Terrain Server allowing queries of point-to-point visibility data and specific point elevation data for tank placement.

7. Potential Improvements

7.1. Updating Tank pKill table in VR Forces

When we checked VR Forces' pKill tables for different munition they are not very detailed in reading munition information to differentiate all the probability values. And for initial pKill values used in existing 73 Easting scenario, the equations are giving effective values when it was hit within 1000 meters.

7.2. Updating details of Detonation Pdu in VR Forces

In detonation Pdu there is a field: Battlefield support, which record details of the munition. We found in VR Forces they didn't implemented very strictly according to IEEE DIS protocol[6]. It could upgrade the "intelligence" of the modeled vehicles and increase the accuracy for the scenario if these details are implemented more comprehensively.

8. References

[1] VR Forces User Guide
[2] Charles Collins, Mikel D. Petty, and Ryan Preston: 73 Easting VR-Forces Model Simulation and Results, 2016
[3] Slides from CS 595 Modeling and Simulation II
[4] http://open-dis.sourceforge.net/Open-DIS.html
[5]http://sturgeonshouse.ipbhost.com/index.php?/topic/131-glorious-t-14-armata-pictures/&page=5
[6] J. Joseph Brann (May 5, 2003): Enumeration and Bit Encoded Values for Use with Protocols for Distributed Interactive Simulation Applications
[7]https://en.wikipedia.org/wiki/T-14_Armata

9. Code Appendix

9.1. Scenario Data Logger

This class stores data about the simulation execution, does a bit of statistics and logs everything to a file when the user ends the execution.

Logger.java

```
import java.io.*;
import java.net.*;
import java.nio.charset.Charset;
import java.util.*;

import edu.nps.moves.dis7.*;
import edu.nps.moves.disutil.Pdu7Factory;

/**
 * The user ends execution by typing enter in the console input.
 * @author Phil Showers
 */
public class Logger implements Runnable{

        public static final int MAX_PDU_SIZE = 8192;
        public static final int PORT = 3300;
        public static final String DEFAULT_MULTICAST_GROUP = "10.56.0.255";
        public static final int APPLICATION_ID = 999;
        public static final short USA_FORCE_ID = 1;
        public static final short IRAQI_FORCE_ID = 2; // 2 for Iraqis

        public static long FUNCTIONAL_APPEARANCE_TANK_F = 6291456; //FROZEN 600000 in HEX
        public static long FUNCTIONAL_APPEARANCE_TANK_N = 4194304; //NOT FROZEN 400000 in HEX
        public static long FUNCTIONAL_APPEARANCE_INFANTRY_F = 39911424;//FROZEN  0x02610000
        public static long FUNCTIONAL_APPEARANCE_INFANTRY_N = 37814272;//NOT FROZEN  0x02410000

        private boolean loop = true;
        private boolean exit = false;
        private MulticastSocket socket = null;

        class KillLog
        {
                String marking;
                EntityStatePdu original;
                EntityStatePdu firstKilled;
                ArrayList<FirePdu> firePDUs = new ArrayList<>();
                ArrayList<DetonationPdu> detPDUs = new ArrayList<>();
                boolean isKilled = false;

                public KillLog(EntityStatePdu esPdu) {
                        original = esPdu;
                        marking = new String(esPdu.getMarking().getCharacters(), Charset.forName("US-ASCII")).trim();
                }

                public void kill(EntityStatePdu esPdu) {
                        firstKilled = esPdu;
                        isKilled = true;
                }

                public void log(FirePdu fire) {
                        try{
                                firePDUs.add(fire);
                        }catch(Exception e)
                        {
                                e.printStackTrace(System.out);
                        }
                }

                public void log(DetonationPdu detonation) {
                        try{
```

```
                        detPDUs.add(detonation);
                }catch(Exception e)
                {
                        e.printStackTrace(System.out);
                }
        }

    }

    private long numES = 0;
    private long numDet = 0;
    private long numDetNothing = 0;
    private long numFire = 0;

    ArrayList<FirePdu> firePDUs = new ArrayList<>();
    ArrayList<DetonationPdu> detPDUs = new ArrayList<>();
    //ArrayList<EntityStatePdu> esPDUs = new ArrayList<>();
    ArrayList<KillLog> killLogs = new ArrayList<>();

    Hashtable<String, FirePdu> fireHash = new Hashtable<>();
    Hashtable<String, DetonationPdu> detHash = new Hashtable<>();
    Hashtable<String, EntityStatePdu> esHash = new Hashtable<>();
    Hashtable<String, KillLog> killLogHash = new Hashtable<>();

    private GregorianCalendar cal = new GregorianCalendar();

    public Logger()
    {
        /****** set up socket *****/
// Default settings. These are used if no system properties are set.
// If system properties are passed in, these are over ridden.

InetAddress address = null; // maybe not needed because of broadcast...

try
{
  address = InetAddress.getByName(DEFAULT_MULTICAST_GROUP);
}
catch(Exception e)
{
  System.out.println(e + " Cannot create multicast address");
  System.exit(0);
}

// All system properties, passed in on the command line via -Dattribute=value
Properties systemProperties = System.getProperties();

// IP address we send to
String destinationIpString = systemProperties.getProperty("destinationIp");

// Port we send to, and local port we open the socket on
String portString = systemProperties.getProperty("port");

// Set up a socket to send information
try
{
  socket = new MulticastSocket(PORT);
```

```
    socket.setBroadcast(true);

    // Where we send packets to, the destination IP address
    if(destinationIpString != null)
    {
       address = InetAddress.getByName(destinationIpString);
    }
}
catch(Exception e)
{
    System.out.println("Unable to initialize networking. Exiting.");
    System.out.println(e);
    //System.exit(-1);
}
                    /********** END setup socket ***********/
        }

        public static void main(String[] args) {
                    Logger logger = new Logger();
                    Thread t = new Thread(logger);
                    t.start();


                    // goofy way of adding a enter
                    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
                    System.out.println("Type enter to stop: \n\n\n");
                    try {
                                reader.readLine();
                    } catch (IOException ex) {
                                ex.printStackTrace();
                    }
                    System.out.println("Terminating..");
                    logger.loop = false;
                    try {
                                while(!logger.exit)
                                            Thread.sleep(1000);
                    } catch (InterruptedException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                    }
        }

        @Override
        public void run() {
                    try {
                                DatagramPacket packet;
                                // InetAddress address;
                                Pdu7Factory pduFactory = new Pdu7Factory();

                                // Loop infinitely, receiving datagrams
                                while (loop) {
                                            byte buffer[] = new byte[MAX_PDU_SIZE];
                                            packet = new DatagramPacket(buffer, buffer.length);

                                            socket.receive(packet);

                                            List<Pdu> pduBundle = pduFactory.getPdusFromBundle(packet.getData());
```

```
                        Iterator it = pduBundle.iterator();

                        while (it.hasNext()) {
                                EntityID eid;
                                Pdu aPdu = (Pdu) it.next();

                                if (aPdu instanceof EntityStatePdu) {
                                        EntityStatePdu esPdu = (EntityStatePdu) aPdu;

                                        if (esPdu.getEntityID().getApplicationID() != (int) APPLICATION_ID)
                                        {
                                           log(esPdu);
                                        }
                                } else {
                                        String name = aPdu.getClass().getName();
                                        /*if (!name.contains("DataPduReceived") && !name.contains("DataPdu"))
                                                System.out.println("Received " + name);*/
                                        if (aPdu instanceof FirePdu) {
                                                FirePdu fire = (FirePdu) aPdu;
                                                log(fire);
                                        }
                                        if (aPdu instanceof DetonationPdu) {
                                                DetonationPdu detonation = (DetonationPdu) aPdu;
                                                log(detonation);
                                        }
                                }
                                // System.out.println();
                        } // end trop through PDU bundle

                } // end while

                //Write Log
                try {
                        File log = new File("73EastingLog.txt");
                        if(log.exists())
                                log.delete();
                        log.createNewFile();
                        PrintWriter out = new PrintWriter(new FileWriter(log));
                        writeLog(out);
                        out.flush();
                        out.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
                exit = true; //done writing, exit.
        } // End try
        catch (Exception e) {
                System.out.println(e);
        }
}

private void log(DetonationPdu detonation) {
        try {
                numDet++;
                detPDUs.add(detonation);
                if(killLogHash.containsKey(getKey(detonation.getFiringEntityID())))
```

```
					killLogHash.get(getKey(detonation.getFiringEntityID())).log(detonation);
				if(killLogHash.containsKey(getKey(detonation.getTargetEntityID()))){
					killLogHash.get(getKey(detonation.getTargetEntityID())).log(detonation);
				}
				else
					numDetNothing++;
				detHash.put(getKey(detonation.getEventID()), detonation);
			} catch (Exception e) {
				// TODO Auto-generated catch block
				//System.out.println("Logger.log(detonation) " + getKey(detonation.getTargetEntityID()) +
((killLogHash.get(getKey(detonation.getTargetEntityID()))!=null)?"notNull":"Null"));
				//System.out.println("Logger.log(detonation) " + getKey(detonation.getExplodingEntityID()));
				e.printStackTrace();
				//System.out.println(detonation.getExplodingEntityID().getEntityID());
			}
		}

		private void log(FirePdu fire) {
			try {
				numFire++;
				firePDUs.add(fire);
				if(killLogHash.containsKey(getKey(fire.getTargetEntityID())))
					killLogHash.get(getKey(fire.getTargetEntityID())).log(fire);
				if(killLogHash.containsKey(getKey(fire.getFiringEntityID())))
					killLogHash.get(getKey(fire.getFiringEntityID())).log(fire);

				fireHash.put(getKey(fire.getEventID()), fire);
			} catch (Exception e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}

		private void log(EntityStatePdu esPdu) {
			try {
				numES++;
				//esPDUs.add(esPdu);
				if(!killLogHash.containsKey(getKey(esPdu.getEntityID()))){
					KillLog KL = new KillLog(esPdu);
					killLogHash.put(getKey(esPdu.getEntityID()), KL);
					killLogs.add(KL);
				}
				if(isKilled(esPdu))
					if(!killLogHash.get(getKey(esPdu.getEntityID())).isKilled)
						killLogHash.get(getKey(esPdu.getEntityID())).kill(esPdu);

				esHash.put(getKey(esPdu.getEntityID()), esPdu);
			} catch (Exception e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}

		private String getKey(EntityID entityID) {
			return entityID.getApplicationID() + ":" + entityID.getEntityID();
		}
		private String getKey(EntityType e) {
			return e.getEntityKind() + ":" + e.getDomain() + ":" +  e.getCountry() + ":" + e.getCategory() + ":" + e.getSubcategory() +
":" + e.getSpecific() + ":" + e.getExtra();
		}

		//check to see if the vehicle is
		private boolean isKilled(EntityStatePdu esPdu) {

			long app = esPdu.getEntityAppearance();
			long tmp = app>>3;// shift this 3 decimals to the right (drop paint scheme bit, mobility bit, firepower bit)
		   tmp = tmp&3; // the next 2 bits are the damage bits, keep them and zero out the rest with bitwise and to number three
			return (tmp!=3); // if tmp == 3, its completely destroyed
		}

		private String getKey(EventIdentifier eventID) {
			return eventID.getSimulationAddress().getApplication() + ":" + eventID.getEventNumber();
		}

		private void writeLog(PrintWriter out) {
			out.println("Number of ES PDUs: \t" + numES);
			out.println("Number of Fire PDUs: \t" + numFire);
			out.println("Number of Detonate PDUs: \t" + numDet);
			out.println("Number of Detonate PDUs containing null Targets: \t" + numDetNothing);

			processFires(out);

			out.println("\nAll Fire PDUs:");
			for(FirePdu f : firePDUs)
				out.println("\t" + getKey(f.getEventID()) + ": " + name(f.getFiringEntityID()) + " fired at " +
name(f.getTargetEntityID()) + "["+typeFire(f)+"] @ "+time(f.getTimestamp()));

			out.println("\nAll Detonate PDUs:");
			for(DetonationPdu f : detPDUs)
				out.println("\t" + getKey(f.getEventID()) + ": " + name(f.getFiringEntityID()) + " fired at " +
name(f.getTargetEntityID()) + "["+typeDet(f)+"] @ "+time(f.getTimestamp()));

			int numDead = 0;
			int deadRed = 0;
			int deadBlue = 0;
			for(KillLog kl : killLogs)
				if(kl.isKilled){
					numDead++;
					if(kl.original.getForceId()==1)
						deadBlue++;
					else
						deadRed++;
				}

			out.println("Number of Dead Entities: \t" + numDead);
			out.println("Number of Dead Blue: \t" + deadBlue);
			out.println("Number of Dead Red: \t" + deadRed);

			for(KillLog kl : killLogs)
				writeLog(out, kl);
		}

		private void processFires(PrintWriter out) {
```

```
		//num hits usa hits / misses
		//num hits iraqi hits / misses
		ArrayList<FirePdu> usaFires = new ArrayList<>();
		int usaHits=0, usaMisses = 0;
		double usaHitSumDist = 0, usaMissSumDist = 0;
		ArrayList<FirePdu> iraqiFires = new ArrayList<>();
		int iraqiHits=0, iraqiMisses = 0;
		double iraqiHitSumDist = 0, iraqiMissSumDist = 0;
		for(FirePdu f : firePDUs)
		{
			String eskey = getKey(f.getFiringEntityID());
			EntityStatePdu firingEntity = esHash.get(eskey);
			if(firingEntity==null)
				System.out.println("Logger.processFires() firingEntity==null for " + eskey + " : " +
getKey(f.getEventID()));
			else
			{
				String evtKey = getKey(f.getEventID());
				DetonationPdu d = detHash.get(evtKey);
				if(firingEntity.getForceId()==USA_FORCE_ID){
					usaFires.add(f);
					if(d==null)
						System.out.println("Logger.processFires() no associated detonate with " +
evtKey);
					else
						if(d.getDetonationResult()==1) //hit
						{
							usaHits++;
							usaHitSumDist += f.getRange();
						}
						else
						{
							usaMisses++;
							usaMissSumDist += f.getRange();
						}
				}// usa shooting
				else{
					iraqiFires.add(f);
					if(d==null)
						System.out.println("Logger.processFires() no associated detonate with " +
evtKey);
					else
						if(d.getDetonationResult()==1) //hit
						{
							iraqiHits++;
							iraqiHitSumDist += f.getRange();
						}
						else
						{
							iraqiMisses++;
							iraqiMissSumDist += f.getRange();
						}
				}//else iraqi shooting
			}//if shooter field not null
		}// for every fire pdu

		out.println("USA forces fired \t" + (usaHits + usaMisses) + "\t times");
		out.println("USA forces hit \t" + usaHits + "\t times");
		out.println("USA forces missed \t" + usaMisses + "\t times");
		out.println("The mean distance of USA shots which hit was \t" + ((double)usaHitSumDist/(double)usaHits) + "\t ");
		out.println("The mean distance of USA shots which missed was \t" + ((double)usaMissSumDist/(double)usaMisses) + "\t 
");

		out.println("Iraqi forces fired \t" + (iraqiHits + iraqiMisses) + "\t times");
		out.println("Iraqi forces hit \t" + iraqiHits + "\t times");
		out.println("Iraqi forces missed \t" + iraqiMisses + "\t times");
		out.println("The mean distance of Iraqi shots which hit was \t" + ((double)iraqiHitSumDist/(double)iraqiHits) + "\t ");
		out.println("The mean distance of Iraqi shots which missed was \t" + ((double)iraqiMissSumDist/(double)iraqiMisses) + "\t 
");

		//sort weapon types used
		Hashtable<String, Integer> usaWeaponHash = new Hashtable<>();
		Hashtable<String, Integer> iraqiWeaponHash = new Hashtable<>();
		Hashtable<String, Integer> usaWeaponKillHash = new Hashtable<>();
		Hashtable<String, Integer> iraqiWeaponKillHash = new Hashtable<>();
		for(FirePdu f : usaFires)
		{
			DetonationPdu d = detHash.get(getKey(f.getEventID()));
			if(d!=null)
			{
				boolean hit = d.getDetonationResult()==1;
				String key = getKey(d.getDescriptor().getMunitionType());
				if(!usaWeaponHash.containsKey(key))
					usaWeaponHash.put(key, new Integer(0));
				usaWeaponHash.put(key, usaWeaponHash.get(key)+1);
				if(hit)
				{
					if(!usaWeaponKillHash.containsKey(key))
						usaWeaponKillHash.put(key, new Integer(0));
					usaWeaponKillHash.put(key, usaWeaponKillHash.get(key)+1);
				}
			}
		}//for each usa fires//*/
		for(FirePdu f : iraqiFires)
		{
			DetonationPdu d = detHash.get(getKey(f.getEventID()));
			if(d!=null)
			{
				boolean hit = d.getDetonationResult()==1;
				String key = getKey(d.getDescriptor().getMunitionType());
				if(!iraqiWeaponHash.containsKey(key))
					iraqiWeaponHash.put(key, new Integer(0));
				iraqiWeaponHash.put(key, iraqiWeaponHash.get(key)+1);
				if(hit)
				{
					if(!iraqiWeaponKillHash.containsKey(key))
						iraqiWeaponKillHash.put(key, new Integer(0));
					iraqiWeaponKillHash.put(key, iraqiWeaponKillHash.get(key)+1);
				}
			}
		}//for each iraqi fires//*/

		out.println("USA Weapon Table");
		for (Enumeration<String> e = usaWeaponHash.keys(); e.hasMoreElements();) {
					String key="";
				try {
							key = e.nextElement();
							int numWeaponShots = usaWeaponHash.get(key);
							int kills = usaWeaponKillHash.get(key);
							out.println(key + "\t" + numWeaponShots + "\t" + kills + "\t"+(kills/numWeaponShots)+"");
					} catch (Exception e1) {
							out.println(key);
							e1.printStackTrace();
					}
			}

			out.println("\nIraqi Weapon Table");
			for (Enumeration<String> e = iraqiWeaponHash.keys(); e.hasMoreElements();) {
					String key="";
				try {
							key = e.nextElement();
							int numWeaponShots = iraqiWeaponHash.get(key);
							int kills = iraqiWeaponKillHash.get(key);
							out.println(key + "\t" + numWeaponShots + "\t" + kills + "\t"+(kills/numWeaponShots)+"");
					} catch (Exception e1) {
							out.println(key);
							e1.printStackTrace();
					}
			}
	}// processFires

	private void writeLog(PrintWriter out, KillLog kl) {
			out.println();
			out.print ("["+getKey(kl.original.getEntityID())+"]"+name(kl.original.getEntityID()));
			out.println(kl.isKilled?"(@"+time(kl.original.getTimestamp())+") was Killed in sim at
"+time(kl.firstKilled.getTimestamp()):" survived the sim.");
			if(kl.firePDUs.size()>0){
					out.println("\nFire Log:");
					for(FirePdu f : kl.firePDUs)
							out.println("\t" + getKey(f.getEventID()) + ": " + name(f.getFiringEntityID()) + " fired at " +
name(f.getTargetEntityID()) + "["+typeFire(f)+"] @ "+time(f.getTimestamp()));
			}
			if(kl.detPDUs.size()>0){
					out.println("\nDetonate Log:");
					for(DetonationPdu d : kl.detPDUs)
							out.println("\t" + getKey(d.getEventID()) + ": " + name(d.getFiringEntityID()) + " detonate at " +
name(d.getTargetEntityID()) + "["+typeDet(d)+"] @ "+time(d.getTimestamp()));
			}

	}

	private String typeDet(DetonationPdu d) {
			return
d.getDetonationResult()+"."+d.getEventID().getSimulationAddress().getApplication()+":"+d.getEventID().getEventNumber()+"."+toString(d.get
Descriptor().getMunitionType());
	}

	private String typeFire(FirePdu f) {
			return
f.getRange()+"."+f.getEventID().getSimulationAddress().getApplication()+":"+f.getEventID().getEventNumber()+"."+toString(f.getDescriptor().
getMunitionType());
```

```
		}

		private String toString(EntityType t) {
			return
t.getEntityKind()+":"+t.getDomain()+":"+t.getCountry()+":"+t.getCategory()+":"+t.getSubcategory()+":"+t.getSpecific();
		}

		private String name(EntityID entityID)
		{
			if(esHash.get(getKey(entityID))!=null)
				return new String(esHash.get(getKey(entityID)).getMarking().getCharacters(),
Charset.forName("US-ASCII")).trim();
			else
				return "Nothing";
		}

		private String time(long timestamp)
		{
			long now = System.currentTimeMillis();
			/*int val = this.getDisTimeUnitsSinceTopOfHour();
    val = (val << 1) | ABSOLUTE_TIMESTAMP_MASK; // always flip the lsb to 1
    return val;
    */
			long val = (timestamp >> 1); // drops the lsb off the timestamp

    /*
    // It turns out that Integer.MAX_VALUE is 2^31-1, which is the time unit value, ie there are
    // 2^31-1 DIS time units in an hour. 3600 sec/hr X 1000 msec/sec divided into the number of
    // msec since the start of the hour gives the percentage of DIS time units in the hour, times
    // the number of DIS time units per hour, equals the time value
    double val = (((double) diff) / (3600.0 * 1000.0)) * Integer.MAX_VALUE;
    int ts = (int) val;*/

			double dval = (double)val;
			dval = dval/Integer.MAX_VALUE;
			dval = dval*(3600.0 * 1000.0);

			// set cal object to current time
    //long currentTime = System.currentTimeMillis(); // UTC milliseconds since 1970
    cal.setTimeInMillis(now);
    // Set cal to top of the hour, then compute what the cal object says was milliseconds since 1970
    // at the top of the hour
    cal.set(Calendar.MINUTE, 0);
    cal.set(Calendar.SECOND, 0);
    cal.set(Calendar.MILLISECOND, 0);
    long time = (long) (cal.getTimeInMillis()+dval); // top of hour + timestamp val


    cal.setTimeInMillis(time);

			return cal.get(Calendar.MINUTE)+":"+cal.get(Calendar.SECOND)+"."+cal.get(Calendar.MILLISECOND);
		}// time

}// Logger
```

9.2. T14 Simulation

Demo.java is where the main method is.

```
/*
 * Using DIS7
 * This demo puts 3 threads(itself, receiver, and simulation) into execution pool and runs them.
 * DataRepository class shall contain all the EntityState Pdus  and "Event" Pdus(FirePdu, DenotationPdu, at.el)
 * Simulation class shall maintain the main simulation logic, handle event queue
 * The 'main' thread simply listens for a keystroke to gracefully shut down the Application.
 * @author Rui Wang and Phil Showers
 */

import java.io.*;
import java.nio.charset.Charset;
import java.util.*;

import edu.nps.moves.dis7.*;

public class Demo {

        public enum NetworkMode {
                UNICAST, MULTICAST, BROADCAST
        };

        /** default multicast group we send on */
        public static final String DEFAULT_MULTICAST_GROUP = "10.56.0.255"; //TODO: Change this to reflect your local reality

        /** Port we send on */
        public static final int PORT = 3000;

        public static void main(String[] args) throws IOException {
                DataRepository dataRepository = new DataRepository();

                EspduReceiver e = new EspduReceiver(dataRepository);
                Simulation s = new Simulation(dataRepository);
                s.start(); // simulation
                e.start(); // receiver

                // goofy way of adding a listen for enter key
                BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
                System.out.println("Type enter to stop: \n\n\n");
                try {
                        reader.readLine();
                } catch (IOException ex) {
                        ex.printStackTrace();
                }
                //System.out.println("Terminating...");

                System.out.println("Finished");
                System.exit(-1);
        }

}//Demo
```

The Data Repository is where the sim stores entity and remote entity data. It is passed between the sim update methods during each tick.

DataRepository.java

```
import java.util.ArrayList;
import java.util.Dictionary;
import java.util.Hashtable;

import edu.nps.moves.deadreckoning.DIS_DR_RVW_04;
import edu.nps.moves.deadreckoning.DIS_DeadReckoning;
import edu.nps.moves.dis7.* ;
import edu.nps.moves.disutil.DisTime;


/*
 * Pdu database
 * shall contain remote  Espdu  table
 * shall contain local  Espdu  table
 * shall contain event pdu  table
 * @author Rui Wang and Phil Showers
 */

public class DataRepository {

        private  Dictionary<String, EntityStatePdu> m_remoteEspdus;
        private  Dictionary<String, DIS_DeadReckoning> m_dr;
        private Dictionary<String, EntityStatePdu> m_localEspdus;

        private ArrayList<T14> localTanks = new ArrayList<>();
        private T14Sender sender = null;
        private TerrainServerInterface terrainServer;
        private int eventUID = 4000;

        public ArrayList<FirePdu> shootList = new ArrayList<>();

        /***************/
        public DataRepository() {
                try {
                        m_remoteEspdus= new Hashtable<>();
                        m_localEspdus = new Hashtable<>();
                        sender = new T14Sender();
                        terrainServer = new TerrainServerInterface();
                        m_localEspdus = new Hashtable<>();
                        m_dr =  new Hashtable<>();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }


        public String getkey(EntityID id) {
                return ""+id.getApplicationID()+id.getEntityID();
        }

        /**
         * @throws Exception *************/
        public void update_remoteEsPdus(EntityStatePdu Rpdu) throws Exception {
                String key = getkey(Rpdu.getEntityID());
                if (m_remoteEspdus.get(key) == null){
                        m_remoteEspdus.put(key, Rpdu); // insert new entity
                } else {
                        //m_remoteEspdus.remove(key); // delete old one
                        m_remoteEspdus.put(key, Rpdu); // update with new one
                } // end else
        }

        /**************/

        public void update_localEsPdus(EntityStatePdu Lpdu) {
                int a =Lpdu.getEntityID().getSiteID();
                int b =Lpdu.getEntityID().getApplicationID();
                int c =Lpdu.getEntityID().getEntityID();
                String key = String.valueOf(a)+String.valueOf(b)+String.valueOf(c);

                if (m_localEspdus.isEmpty()) {
                        m_localEspdus.put(key, Lpdu);
                } else { // local localEsPdus not empty
                        if (m_localEspdus.get(key) == null){
                                m_localEspdus.put(key, Lpdu); // insert new entity
                        } else {
                                m_localEspdus.put(key, Lpdu); // update with new one
                        } // end else

                } // end else

        }
        /**************/

        public void update_dr( EntityStatePdu Rpdu) throws Exception {
                String key = getkey(Rpdu.getEntityID());

                if (m_dr.isEmpty()) {
                        DIS_DeadReckoning Rdr = getDR(Rpdu);
                        m_dr.put(key, Rdr);
                } else { //
                        DIS_DeadReckoning dr = m_dr.get(key);
                        if (dr == null){
                                DIS_DeadReckoning Rdr = getDR(Rpdu);
                                m_dr.put(key, Rdr); // insert new entity

                        } else {

                                double lx , ly, lz, ophi,opsi,otheta, lvx, lvy,lvz,Ax, Ay, Az, AVx, AVy, AVz = 0;

                                lx = Rpdu.getEntityLocation().getX();
                                ly =Rpdu.getEntityLocation().getY();
                                lz = Rpdu.getEntityLocation().getZ();
                                ophi = Rpdu.getEntityOrientation().getPhi();
                                opsi=Rpdu.getEntityOrientation().getPsi();
                                otheta = Rpdu.getEntityOrientation().getTheta();
                                lvx = Rpdu.getEntityLinearVelocity().getX();
                                lvy = Rpdu.getEntityLinearVelocity().getY();
                                lvz = Rpdu.getEntityLinearVelocity().getZ();
                                Ax = Rpdu.getDeadReckoningParameters().getEntityLinearAcceleration().getX();
                                Ay = Rpdu.getDeadReckoningParameters().getEntityLinearAcceleration().getY();
                                Az = Rpdu.getDeadReckoningParameters().getEntityLinearAcceleration().getZ();
```

```
                        AVx = Rpdu.getDeadReckoningParameters().getEntityAngularVelocity().getX();
                        AVy = Rpdu.getDeadReckoningParameters().getEntityAngularVelocity().getY();
                        AVz = Rpdu.getDeadReckoningParameters().getEntityAngularVelocity().getZ();

                        // make the arrays of location and other parameters
                //          loc         orien           lin V        Accel       Ang V
                double[] locOr = {lx, ly, lz,   ophi,opsi,otheta,   lvx, lvy,lvz,  Ax, Ay, Az,   AVx, AVy, AVz};
                        dr.setNewAll(locOr);
                    } // end else

                } // end else



        }
        /***************/
        public void update_firePdus(FirePdu Fpdu) {
                // more to be done

        }

        /***************/
        public void update_detonationPdus (DetonationPdu dePdu) {

                // more to be done
        }
        /***************/


        public Dictionary<String, EntityStatePdu> getRemoteEspdus() {
                return m_remoteEspdus;
        }

        /***************/

        public Dictionary<String, EntityStatePdu> getLocalEspdus() {
                return m_localEspdus;
        }
        /***************/

        public Dictionary<String, DIS_DeadReckoning> getDeadReckonings() {
                return m_dr;
        }
        /***************/

        public void notify_FirePdu(FirePdu fire) {
                //System.out.println("DataRepository.notify_FirePdu()" + fire.getFiringEntityID().getEntityID() + " at " +
fire.getTargetEntityID().getEntityID());
                EntityID id = fire.getTargetEntityID();
                for(T14 tank : localTanks)
                        if(tank.getEntityID(null).equalsImpl(id))
                        {
                                tank.fireTarget(fire);
                                return;
                        }
        }
```

```
	public void notify_DetonationPdu(DetonationPdu detonation) {
		//System.out.println("DataRepository.notify_DetonationPdu()" + detonation.getFiringEntityID().getEntityID() + " at " +
detonation.getTargetEntityID().getEntityID());
		EntityID id = detonation.getTargetEntityID();
		for(T14 tank : localTanks)
			if(tank.getEntityID(null).equalsImpl(id))
			{
				tank.detonationTarget(detonation);
				return;
			}
	}

	public void addTank(T14 t14) {
		if(!localTanks.contains(t14))
			localTanks.add(t14);
	}
	public ArrayList<T14> getTanks() {
		return localTanks;
	}
	public void sendESPdu(EntityStatePdu entityStatePdu) {
		entityStatePdu.setTimestamp(DisTime.getInstance().getDisRelativeTimestamp());
		entityStatePdu.setLength(entityStatePdu.getMarshalledSize());
		sender.send(entityStatePdu);
	}


	public static DIS_DeadReckoning getDR( EntityStatePdu obj) throws Exception{
		//DIS_DeadReckoning dr = new DIS_DR_FPW_02();
		DIS_DeadReckoning dr = new DIS_DR_RVW_04();
		double lx , ly, lz, ophi,opsi,otheta, lvx, lvy,lvz,Ax, Ay, Az, AVx, AVy, AVz = 0;
		lx = obj.getEntityLocation().getX();
		ly =obj.getEntityLocation().getY();
		lz = obj.getEntityLocation().getZ();
		ophi = obj.getEntityOrientation().getPhi();
		opsi=obj.getEntityOrientation().getPsi();
		otheta = obj.getEntityOrientation().getTheta();
		lvx = obj.getEntityLinearVelocity().getX();
		lvy = obj.getEntityLinearVelocity().getY();
		lvz = obj.getEntityLinearVelocity().getZ();
		Ax = obj.getDeadReckoningParameters().getEntityLinearAcceleration().getX();
		Ay = obj.getDeadReckoningParameters().getEntityLinearAcceleration().getY();
		Az = obj.getDeadReckoningParameters().getEntityLinearAcceleration().getZ();
		AVx = obj.getDeadReckoningParameters().getEntityAngularVelocity().getX();
		AVy = obj.getDeadReckoningParameters().getEntityAngularVelocity().getY();
		AVz = obj.getDeadReckoningParameters().getEntityAngularVelocity().getZ();
		// make the arrays of location and other parameters
    //		loc		orien		lin V		Accel	Ang V
		double[] locOr = {lx, ly, lz,   ophi,opsi,otheta,   lvx, lvy,lvz,  Ax, Ay, Az,   AVx, AVy, AVz};
		// set the parameters
    dr.setNewAll(locOr);

    return dr;
	}

	public void sendFirePDU(FirePdu fPDU) {
		fPDU.setTimestamp(DisTime.getInstance().getDisRelativeTimestamp());
		fPDU.setLength(fPDU.getMarshalledSize());
```

```
			shootList.add(fPDU);
			sender.send(fPDU);
		}

		public void sendDetonationPDU(DetonationPdu dPDU) {
			dPDU.setTimestamp(DisTime.getInstance().getDisRelativeTimestamp());
			dPDU.setLength(dPDU.getMarshalledSize());
			sender.send(dPDU);
		}

		public TerrainServerInterface getTerrainServer(){
			return terrainServer;
		}


		public int getEventID() {
			eventUID++;
			return eventUID;
		}

		public boolean isTerrainServerLive() {

			if(terrainServer==null)
				return false;
			else
				try{
					terrainServer.getAltitude(3817940.9117024885, 4036729.835147949, 3123425.9327854933);
					return true;
				}catch(Exception e){
					e.printStackTrace(System.out);
					return false;
				}
		}
} // end DataRepository class
```

The Simulation class is where the local entity update loop runs in its own thread. Simulation.java

```
import edu.nps.moves.dis7.*;
import edu.nps.moves.disutil.*;

/*
 * main logic
 * This is where the update loop lives
 * @author rui wang and phil showers
 */
public class Simulation extends Thread {
		public static final short EXERCISE_ID = 1;
		public static final short FORCE_ID = 2; // 2 for Iraqis
		public static final short APPLICATION_ID = 1;// 595;
		public static final short SITE_ID = 1;

		private DataRepository dataObj;
```

```
public Simulation(DataRepository data) {
        dataObj = data;
}

public void run() {

        /******* Verify Connectivity to Terrain Server *******/
        while(true)
                if(dataObj.isTerrainServerLive())
                        break;

        /******* create local entity *******/
        T14 t14_1 = new T14(EXERCISE_ID, FORCE_ID, APPLICATION_ID, SITE_ID, "_AR 29_");
        t14_1.setLocation(29.515003, 46.600697, 249, dataObj);
        T14 t14_2 = new T14(EXERCISE_ID, FORCE_ID, APPLICATION_ID, SITE_ID, "_AR 33_");
        t14_2.setLocation(29.514684, 46.602762, 249, dataObj);
        T14 t14_3 = new T14(EXERCISE_ID, FORCE_ID, APPLICATION_ID, SITE_ID, "_AR 46_");
        t14_3.setLocation(29.514522, 46.603010, 249, dataObj);
        T14 t14_4 = new T14(EXERCISE_ID, FORCE_ID, APPLICATION_ID, SITE_ID, "_AR 47_");
        t14_4.setLocation(29.514521, 46.602031, 249, dataObj); // set tank location with its position in (lat, lon, alt)
        T14 t14_5 = new T14(EXERCISE_ID, FORCE_ID, APPLICATION_ID, SITE_ID, "_AR 48_");
        t14_5.setLocation(29.514702, 46.602278, 249, dataObj);
        T14 t14_6 = new T14(EXERCISE_ID, FORCE_ID, APPLICATION_ID, SITE_ID, "_AR 49_");
        t14_6.setLocation(29.514522, 46.602515, 249, dataObj);
        T14 t14_7 = new T14(EXERCISE_ID, FORCE_ID, APPLICATION_ID, SITE_ID, "_AR 50_");
        t14_7.setLocation(29.514999, 46.600476, 250, dataObj);
        T14 t14_8 = new T14(EXERCISE_ID, FORCE_ID, APPLICATION_ID, SITE_ID, "_AR 51_");
        t14_8.setLocation(29.514993, 46.600259, 250, dataObj);
        T14 t14_9 = new T14(EXERCISE_ID, FORCE_ID, APPLICATION_ID, SITE_ID, "_AR 72_");
        t14_9.setLocation(29.514299, 46.602272, 250, dataObj);
        T14 t14_10 = new T14(EXERCISE_ID, FORCE_ID, APPLICATION_ID, SITE_ID, "_AR 73_");
        t14_10.setLocation(29.514311, 46.602773, 250, dataObj);

        // old test checkpoint
        //29.517024, 46.555961, 250, dataObj);

        dataObj.addTank(t14_1);
        dataObj.addTank(t14_2);
        dataObj.addTank(t14_3);
        dataObj.addTank(t14_4);
        dataObj.addTank(t14_5);
        dataObj.addTank(t14_6);
        dataObj.addTank(t14_7);
        dataObj.addTank(t14_8);
        dataObj.addTank(t14_9);
        dataObj.addTank(t14_10);
        /******* END create local entity *******/

        DisTime dt = DisTime.getInstance();
        int timestamp = dt.getDisRelativeTimestamp();
        System.out.println("begin at " + timestamp);

        while (true) {
                for (T14 tank : dataObj.getTanks()){
                        //System.out.println("Simulation.run() update " + tank.marking);
                        tank.update(dataObj);
                }
```

```
				}

		} // end run

		public static void printLocation(EntityStatePdu e) {
				System.out.print(" EID=[" + e.getEntityID().getSiteID() + "," + e.getEntityID().getApplicationID() + ","
							+ e.getEntityID().getEntityID() + "]");
				System.out.print(" DIS coordinates location=[" + e.getEntityLocation().getX() + ","
							+ e.getEntityLocation().getY() + "," + e.getEntityLocation().getZ() + "]");
				double c[] = { e.getEntityLocation().getX(), e.getEntityLocation().getY(), e.getEntityLocation().getZ() };
				double lla[] = CoordinateConversions.xyzToLatLonDegrees(c);
				System.out.println(" Location (lat/lon/alt): [" + lla[0] + ", " + lla[1] + ", " + lla[2] + "]");
		}

}// end simulation class
```

TerrainServerInterface class is where requests are formatted and parsed to and from the Terrain Server

```
TerrainServerInterface .java
import java.io.*;
import java.net.*;

/**
 * This opens a tcp socket and passes data requests to the Terrain Server
 * @author Phil Showers
 *
 */
public class TerrainServerInterface {

		public static String HOST = "10.56.1.167";
		public static int PORT = 8001;
		private Socket socket = null;
		private PrintWriter out = null;
		BufferedReader in = null;

		public TerrainServerInterface() {
				try{
						socket = new Socket(HOST, PORT);
						out = new PrintWriter(socket.getOutputStream(), true);
						in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
				} catch (UnknownHostException e) {
						System.err.println("Don't know about host " + HOST);
						System.exit(1);
				} catch (IOException e) {
						//System.err.println("Couldn't get I/O for the connection to " + hostName);
						e.printStackTrace(System.out);
						System.exit(1);
				}
		}// constructor

		public double getAltitude(double x, double y, double z)
		{
				try {
						String message = "ELEVATION " + x + " " + y + " " + z;
						out.println(message);
```

```
					String reply = in.readLine();
					double elevation = Double.parseDouble(reply.trim());
					return elevation;
				} catch (NumberFormatException e) {
					// The first message that the Terrain server sends back is a bad number, like "ï»¿250.285759982316", so just
throw it out.
					//e.printStackTrace();
				} catch (IOException e) {
					e.printStackTrace(System.out); // probably a bad thing...
				}
				return -1000;
		}

		public boolean canSee(double x1, double y1, double z1, double x2, double y2, double z2)
		{
				String message = "CANSEE " + x1 + " " + y1 + " " + z1 + " " + x2 + " " + y2 + " " + z2;
				try {
					out.println(message);
					String reply = in.readLine();
					boolean visible = Boolean.parseBoolean(reply);
					return visible;
				} catch (Exception e) {
					e.printStackTrace(System.out);
					System.exit(646464);
				}
				return false;
		}

}// TerrainServerInterface
```

The EspduReceiver class is responsible for receiving PDUs and passing them to the Data Repository.

EspduReceiver.java

```
/*
 * receiver thread
 * @author rui wang and phil showers
 */

import java.net.*;
import java.util.*;

import edu.nps.moves.disutil.*;
import edu.nps.moves.dis7.*;

public class EspduReceiver extends Thread {

		private DataRepository dataObj;

		/**
		 * Max size of a PDU in binary format that we can receive. This is actually
		 * somewhat outdated--PDUs can be larger--but this is a reasonable starting
		 * point
		 */
		public static final int MAX_PDU_SIZE = 8192;
```

```
        public enum NetworkMode {
                UNICAST, MULTICAST, BROADCAST
        };

        public EspduReceiver(DataRepository data) { // DataRepository data
                dataObj = data;
        }

        public void run() {
                /****** set up socket *****/
                MulticastSocket socket = null;
// Default settings. These are used if no system properties are set.
// If system properties are passed in, these are over ridden.

//InetAddress address = null; // maybe not needed because of broadcast...

try
{
   //address = InetAddress.getByName(Demo.DEFAULT_MULTICAST_GROUP);
}
catch(Exception e)
{
   System.out.println(e + " Cannot create multicast address");
   System.exit(0);
}

// All system properties, passed in on the command line via -Dattribute=value
Properties systemProperties = System.getProperties();

// IP address we send to
String destinationIpString = systemProperties.getProperty("destinationIp");

// Set up a socket to send information
try
{
   socket = new MulticastSocket(Demo.PORT);
   socket.setBroadcast(true);

   // Where we send packets to, the destination IP address
   if(destinationIpString != null)
   {
      //address = InetAddress.getByName(destinationIpString);
   }
}
catch(Exception e)
{
   System.out.println("Unable to initialize networking. Exiting.");
   System.out.println(e);
   //System.exit(-1);
}
                /********** END setup socket ***********/

                DatagramPacket packet;
                // InetAddress address;
                Pdu7Factory pduFactory = new Pdu7Factory();
```

```
		try {
			// Loop infinitely, receiving datagrams
			while (true) {
				byte buffer[] = new byte[MAX_PDU_SIZE];
				packet = new DatagramPacket(buffer, buffer.length);

				socket.receive(packet);

				List<Pdu> pduBundle = pduFactory.getPdusFromBundle(packet.getData());

				Iterator<Pdu> it = pduBundle.iterator();

				while (it.hasNext()) {
					//EntityID eid;
					Pdu aPdu = (Pdu) it.next();

					if (aPdu instanceof EntityStatePdu) {
						EntityStatePdu esPdu = (EntityStatePdu) aPdu;

						if (esPdu.getEntityID().getApplicationID() != (int)
Simulation.APPLICATION_ID)
						{
							dataObj.update_remoteEsPdus(esPdu);
						   dataObj.update_dr(esPdu);
						}
					} else {
						//String name = aPdu.getClass().getName();
						//if (!name.contains("DataPduReceived") && !name.contains("DataPdu"))
						//	System.out.println("Received " + name);
						if (aPdu instanceof FirePdu) {
							FirePdu fire = (FirePdu) aPdu;
							dataObj.notify_FirePdu(fire);
						}
						if (aPdu instanceof DetonationPdu) {
							DetonationPdu detonation = (DetonationPdu) aPdu;
							dataObj.notify_DetonationPdu(detonation);
						}
					}
					// System.out.println();
				} // end trop through PDU bundle

			} // end while
		} // End try
		catch (Exception e) {

			System.out.println(e);
		}

	} // end run

} // end PDU Receiver
```

The T14Sender class transmits PDUs sent to the Data Repository for transmission.

```
T14Sender.java
import java.io.*;
import java.net.*;
import java.util.*;

import edu.nps.moves.dis7.*;

/**
 * Creates and sends ESPDUs in IEEE binary format.
 * @author phil showers
 */
public class T14Sender {

		private MulticastSocket socket = null;

		public enum NetworkMode {
			UNICAST, MULTICAST, BROADCAST
		};

		public T14Sender()
		{
			/****** set up socket *****/
			try {
				socket = new MulticastSocket(Demo.PORT);
				socket.setBroadcast(true);
			} catch (Exception e) {
				System.out.println(e + " Cannot create multicast address");
				System.exit(0);
			}
			/********** END setup socket ***********/
		}

		long lastSent = -1;
		public void send(Pdu pdu) {
			Set<InetAddress> bcastAddresses = getBroadcastAddresses();
    Iterator<InetAddress> it = bcastAddresses.iterator();
			try {
				ByteArrayOutputStream baos = new ByteArrayOutputStream();
				DataOutputStream dos = new DataOutputStream(baos);
				pdu.marshal(dos);

				// The byte array here is the packet in DIS format. We put
				// that into a datagram and send it.
				byte[] data = baos.toByteArray();

				while(it.hasNext())
            {
              InetAddress bcast = (InetAddress)it.next();
              //System.out.println("Sending bcast to " + bcast);
              DatagramPacket packet = new DatagramPacket(data, data.length, bcast, 3000);
              try {
							socket.send(packet);
						} catch (IOException e) {
							e.printStackTrace(System.out);
						}
            }
			} // end try
			catch (Exception e) {
				e.printStackTrace(System.out);
			}
```

```
    }


    /**
     * FROM edu.nps.moves.examples.EspduSender
     *
     * A number of sites get all snippy about using 255.255.255.255 for a bcast
     * address; it trips their security software and they kick you off their
     * network. (Comcast, NPS.) This determines the bcast address for all
     * connected interfaces, based on the IP and subnet mask. If you have
     * a dual-homed host it will return a bcast address for both. If you have
     * some VMs running on your host this will pick up the addresses for those
     * as well--eg running VMWare on your laptop with a local IP this will
     * also pick up a 192.168 address assigned to the VM by the host OS.
     *
     * @return set of all bcast addresses
     */
     public static Set<InetAddress> getBroadcastAddresses()
     {
        Set<InetAddress> bcastAddresses = new HashSet<InetAddress>();
        Enumeration<NetworkInterface> interfaces;

        try
        {
           interfaces = NetworkInterface.getNetworkInterfaces();

           while(interfaces.hasMoreElements())
           {
              NetworkInterface anInterface = (NetworkInterface)interfaces.nextElement();

              if(anInterface.isUp())
              {
                 Iterator<InterfaceAddress> it = anInterface.getInterfaceAddresses().iterator();
                 while(it.hasNext())
                 {
                    InterfaceAddress anAddress = (InterfaceAddress)it.next();
                    if((anAddress == null || anAddress.getAddress().isLinkLocalAddress()))
                       continue;

                    //System.out.println("Getting bcast address for " + anAddress);
                    InetAddress abcast = anAddress.getBroadcast();
                    if(abcast != null)
                     bcastAddresses.add(abcast);
                 }
              }
           }

        }
        catch(Exception e)
        {
           e.printStackTrace();
           System.out.println(e);
        }

        return bcastAddresses;
     }
}
```

The T14 class is the brains of the simulation. Each T14 instance does its own targeting, damage, and pHit and pKill calculations.

T14.java

```
/*
 * T14 Tank Class
 * @author Rui Wang
 * @author Phil Showers
 */

import java.io.*;
import java.nio.charset.Charset;
import java.util.*;

import edu.nps.moves.dis7.*;
import edu.nps.moves.disutil.CoordinateConversions;
import edu.nps.moves.disutil.DisTime;

public class T14 {

		/******** T14 Field ********/
		public static long FUNCTIONAL_APPEARANCE = 4194304; //NOT FROZEN 400000 in HEX
		public static long MOBILITYKILLED_APPEARANCE = 4194306; // 408038 in hex
		public static long KILLED_APPEARANCE = 4227128; // 408038 in hex
		public static double MIN_FIRE_DISTANCE = 50;
		public static double MAX_FIRE_DISTANCE = 4000;

		public static double FIRE_DELAY = 10; // 10 seconds

		private static int m_count = 302; // keep track on how many T14 have been created
		private int ID = -1;

		protected String marking;

		protected DisTime m_disTime; // time padding
		protected short ExerciseID;
		protected short ForceId;
		protected short ApplicationID;
		protected short SiteID;

		private int heatRounds = 20;
		private int sabotRounds = 25;

		protected double CrewQualityMultiplier = 0.8; // range is 0 to 1, with 0

// being absent and 1 being

// the best.

		protected double m_disCoordinates[]; // tank's Coordinates in X, Y, Z
		protected Vector3Double m_location; // Tank's location in lan, lon, alt

		protected EulerAngles m_orientation;
		protected Vector3Float m_linearVelocity;
	protected FirePdu m_firepdu; // fire event
	protected DetonationPdu m_depdu; // been hit, report demage
	protected EntityStatePdu aimed_target = null;

	protected long m_appearance;

	/********* End Field **************/

	/*** constructor with location ***/
	public T14(short exerciseID, short forceID, short applicationID, short siteID, String name) {
		ExerciseID = exerciseID;
		ForceId = forceID;
		ApplicationID = applicationID;
		SiteID = siteID;
		localClock = System.currentTimeMillis();
		m_count++; // increment T14 count
		ID = m_count;
		marking = name;

		m_appearance = FUNCTIONAL_APPEARANCE;
		m_disTime = DisTime.getInstance();
		System.out.println("T14() " + marking + " " + ID);
	} // end constructor

	private boolean sendFlag = true;
	private long lastTsent = -1;

	long lastUpdated = -1;
	private boolean firedFlag = false;
	private ArrayList<FirePdu> firePdus = new ArrayList<>();
	private boolean detonateFlag = false;
	private ArrayList<DetonationPdu> detonatePdus = new ArrayList<>();
	private long localClock = -1;
	private double fireTime = 0.0;
	private boolean fireWait = false;
	private FirePdu currentFire = null;

	public void update(DataRepository dataObj) {
		if(m_appearance!=FUNCTIONAL_APPEARANCE)
		{
			if (shouldSendUpdate())
				dataObj.sendESPdu(this.getEntityStatePdu());
			return;
		}

try{
		// Does the Tank see something to shoot at?
		if (aimed_target == null) {
			findNearestTarget(dataObj);
			// calculate fire delay
			fireTime = System.currentTimeMillis();
		} else {
			// check if the target dead
			EntityID targetID = aimed_target.getEntityID();
			if(targetID==null)
			{
```

```
                                                System.out.println("T14.update() targetID == null");
                                                System.exit(-1);
                                }
                                String key = dataObj.getkey(targetID);
                                if(key==null)
                                {
                                                System.out.println("T14.update() dataObj.getkey(targetID) == null");
                                                System.exit(-1);
                                }
                                Dictionary<String, EntityStatePdu> remTable = dataObj.getRemoteEspdus();
                                if(remTable==null)
                                {
                                                System.out.println("T14.update() dataObj.getRemoteEspdus() == null");
                                                System.exit(-1);
                                }
                                EntityStatePdu esp = dataObj.getRemoteEspdus().get(key);
                                if(esp==null)
                                {
                                                System.out.println("T14.update() dataObj.getRemoteEspdus().get("+key+") == null");
                                                System.exit(-1);
                                }

                                long killVar = esp.getEntityAppearance()>>3;
                                killVar = killVar&3;
                                if (killVar==3){//(esp.getEntityAppearance() == KILLED_APPEARANCE) {
                                                // target is dead
                                                aimed_target = null;
                                                fireWait = false;
                                } else {
                                                // shoot at target if time>=firetime
                                                long t = System.currentTimeMillis();
                                                if (t >= fireTime){
                                                                if (!fireWait) {
                                                                                // firewait is false - so shoot!
                                                                                currentFire = shootAt(targetID, dataObj); // generates fire pdu and xmits it
                                                                                if(currentFire!=null) // abort fire - target ES lookup failed to find in remote ES
dictionary
                                                                                {
                                                                                                fireWait = true;
                                                                                                fireTime = t+500; // wait half a second and then calculate hit
                                                                                }
                                                                                else // out of ammo
                                                                                {
                                                                                                // do no more fighting until I die.
                                                                                }
                                                                } else {
                                                                                // fire wait is true - so calculate pHit

                                                                                // Target was hit - send a detonate!
                                                                                detonateAt(targetID, dataObj,
isTargetHit(getRange(dataObj.getDeadReckonings().get(dataObj.getkey(targetID))
                                                                                                .getUpdatedPositionOrientation())), currentFire);

                                                                                // decide to shoot at target again
                                                                                fireWait = false;
                                                                                currentFire = null;
```

```
								// as CrewQualityMultiplier -> 0, firedelay -> 15
								fireTime = t + (FIRE_DELAY + (5.0 - 5.0 * CrewQualityMultiplier)) * 1000;
						}
					}/*
					else
					{
						System.out.println("T14.update() waiting for fireTime " + fireTime + ":" + t);
					}//*/
				}
			}

			// Has the tank been shot at?
			if (firedFlag) {
				fireUpdate();

			}
			if (detonateFlag)
				detonateUpdate(dataObj);

			// if it's time to update Espdus
			if (shouldSendUpdate())
				dataObj.sendESPdu(this.getEntityStatePdu());
}catch(Exception derp)
{
		derp.printStackTrace(System.out);
}
		}// update

		private void detonateAt(EntityID target, DataRepository dataObj, boolean hit, FirePdu fire) {
			DetonationPdu dPDU = new DetonationPdu();
			dPDU.setExerciseID(ExerciseID);
			dPDU.setFiringEntityID(fire.getFiringEntityID());
			if(hit)
				dPDU.setTargetEntityID(fire.getTargetEntityID());
			dPDU.setEventID(fire.getEventID());
			dPDU.setDescriptor(fire.getDescriptor());
			dPDU.setVelocity(fire.getVelocity());

			double[] locAndOrientation = dataObj.getDeadReckonings().get(dataObj.getkey(target))
					.getUpdatedPositionOrientation();
			Vector3Double location = dPDU.getLocationInWorldCoordinates();
			location.setX(locAndOrientation[0]);
			location.setY(locAndOrientation[1]);
			location.setZ(locAndOrientation[2]);

			dPDU.setDetonationResult((short) (hit?1:0));

			Vector3Float loc = dPDU.getLocationOfEntityCoordinates();
			loc.setX(-4.0f);
			loc.setY(0.0f);
			loc.setZ(0.0f);

			dPDU.setTimestamp(DisTime.getInstance().getDisRelativeTimestamp());

			System.out.println("T14.detonateAt() " + hit);
			dataObj.sendDetonationPDU(dPDU);
			// isTargetHit
```

```
	}

	private FirePdu shootAt(EntityID target, DataRepository dataObj) {
		EntityStatePdu targetES = dataObj.getRemoteEspdus().get(dataObj.getkey(target));
		if(targetES==null)
			return null;

		FirePdu fPDU = new FirePdu();
		fPDU.setExerciseID(ExerciseID);

		EntityID firingId = fPDU.getFiringEntityID();
		firingId.setSiteID(SiteID);
		firingId.setApplicationID(ApplicationID);
		firingId.setEntityID(ID);

		EntityID targetId = fPDU.getTargetEntityID();
		targetId.setSiteID(target.getSiteID());
		targetId.setApplicationID(target.getApplicationID());
		targetId.setEntityID(target.getEntityID());
		fPDU.setTargetEntityID(targetES.getEntityID());

		EventIdentifier eventId = fPDU.getEventID();
		eventId.getSimulationAddress().setApplication(ApplicationID);
		eventId.getSimulationAddress().setSite(SiteID);
		eventId.setEventNumber(dataObj.getEventID());

		double[] locAndOrientation = dataObj.getDeadReckonings().get(dataObj.getkey(target))
					.getUpdatedPositionOrientation();
		Vector3Double location = fPDU.getLocationInWorldCoordinates();
		location.setX(locAndOrientation[0]);
		location.setY(locAndOrientation[1]);
		location.setZ(locAndOrientation[2]);


		MunitionDescriptor descriptor = fPDU.getDescriptor();
		descriptor.setQuantity(4);
		EntityType munitionType = descriptor.getMunitionType();
		munitionType.setEntityKind((short) 2); // munition (vs platform,
																		// lifeform,
munition, sensor,
																		// etc.)
		munitionType.setDomain((short) 1); // Land (vs air, surface, subsurface,
																// space)
		munitionType.setCountry(222); // 102 Iraq
		// if shooting m1s(1 1 3), use 2 11 1
		if(targetES.getEntityType().getCategory()==1 && targetES.getEntityType().getSubcategory()==1 &&
targetES.getEntityType().getSpecific()==3){
			if(sabotRounds==0){
				//out.println(marking".shootAt() ran out of sabots trying to shoot " + name(targetES));
				munitionType.setCategory((short) 2);
				munitionType.setSubcategory((short) 11);
				munitionType.setSpecific((short) 3);
			}
			munitionType.setCategory((short) 2);
			munitionType.setSubcategory((short) 11);
			munitionType.setSpecific((short) 1);
			sabotRounds--;
			}
			else // use 2 11 3
			{
				if(heatRounds==0){
					//out.println(marking".shootAt() ran out of HEAT trying to shoot " + name(targetES));
					munitionType.setCategory((short) 2);
					munitionType.setSubcategory((short) 11);
					munitionType.setSpecific((short) 1);
				}
				munitionType.setCategory((short) 2);
				munitionType.setSubcategory((short) 11);
				munitionType.setSpecific((short) 3);
				heatRounds--;
			}

			if(heatRounds==0 && sabotRounds==0)
				return null;
			descriptor.setRate(600); // just 'cause

			fPDU.setRange(getRange(locAndOrientation));
			calculateVelocity(fPDU, locAndOrientation);

			fPDU.setTimestamp(DisTime.getInstance().getDisRelativeTimestamp());


			System.out.println("T14.shootAt() " + dataObj.getkey(fPDU.getFiringEntityID()) + " shot at " +
dataObj.getkey(fPDU.getTargetEntityID()));

			dataObj.sendFirePDU(fPDU);
			// isTargetHit
			return fPDU;
		}

		private void calculateVelocity(FirePdu fPDU, double[] locAndOrientation) {
			double x = locAndOrientation[0] - m_location.getX();
			double y = locAndOrientation[1] - m_location.getY();
			double z = locAndOrientation[2] - m_location.getZ();
			double vel = getRange(locAndOrientation) / fPDU.getDescriptor().getRate();
			fPDU.getVelocity().setX((float) (x / vel));
			fPDU.getVelocity().setY((float) (y / vel));
			fPDU.getVelocity().setZ((float) (z / vel));
		}

		private float getRange(double[] location) {
			double lx = m_disCoordinates[0];
			double ly = m_disCoordinates[1];
			double lz = m_disCoordinates[2];

			double rx = location[0];
			double ry = location[1];
			double rz = location[2];
			double distance = Math.sqrt(Math.pow((lx - rx), 2.0) + Math.pow((ly - ry), 2.0) + Math.pow((lz - rz), 2.0));

			return (float) distance;
		}

		private void detonateUpdate(DataRepository dataObj) {
```

```
		for (DetonationPdu d : detonatePdus) {
			if(!canKill(d.getDescriptor().getMunitionType()))
				continue;

			System.out.println(d.getFiringEntityID().getEntityID() + " detonated " + this.ID);

			String key = dataObj.getkey(d.getFiringEntityID());
			double[] r = dataObj.getDeadReckonings().get(key).getUpdatedPositionOrientation();
			double rx = r[0];
			double ry = r[1];
			double rz = r[2];
			double lx = this.getM_location().getX();
			double ly = this.getM_location().getY();
			double lz = this.getM_location().getZ();

			double fire_distance = Math
					.sqrt(Math.pow((lx - rx), 2.0) + Math.pow((ly - ry), 2.0) + Math.pow((lz - rz), 2.0));

			if (isDead(fire_distance)){
				m_appearance = KILLED_APPEARANCE;
				break;
			}
		}
		detonatePdus.clear();
		detonateFlag = false;
	}

	//returns true if the munition has the ability to kill the tank
	private boolean canKill(EntityType t) {
		return t.getCategory()==1 || (t.getCategory()==2 && t.getSubcategory()==13 && t.getSpecific()==2);
	}

	private void fireUpdate() {
		for (FirePdu f : firePdus)
			System.out.println(f.getFiringEntityID().getEntityID() + " fired at " + this.ID);
		firePdus.clear();
		firedFlag = false;
	}

	// if the tank is moving then send a ES pdu when local dead reckoning
	// and actual sim location diverge
	// otherwise send "heartbeat" ES pdu once every 5 seconds.
	// TODO: this algorithm is not exactly right...
	private boolean shouldSendUpdate() {
		long num = 1; // send a ESPdu every num seconds

		long t = ((System.currentTimeMillis() - localClock) / 250); // send a pdu every 1/2 sec?
		// System.out.println("\t" + t);
		if (t % num == 0) // if the time since the sim started is divisable by
								// num then update ES PDU
		{
			if (sendFlag) {
				sendFlag = false; // send once and then set flag to false
				// System.out.println("\t" + t + "%" + num + " = " + (t % num));
				return true;
			} else {
				// System.out.println("not sending");
			}
		} else {
			sendFlag = true; // set flag to true once the time is past
		}

		if (t != lastTsent) {
			lastTsent = t;
			sendFlag = true;
		}

		return false;
	}// shouldSendUpdate

	/**
	 * @param Marking ***************************************************************/
	public EntityStatePdu getEntityStatePdu() {
		EntityStatePdu espdu = new EntityStatePdu();
		espdu.setExerciseID(ExerciseID);
		espdu.setForceId(ForceId);

		espdu.setNumberOfVariableParameters((short) 4); // this selects the

	// symbol in VRForces

		// m_setEntityIDParameter(applicationID, siteID);
		EntityID eid = espdu.getEntityID();
		eid.setSiteID(SiteID);
		eid.setApplicationID(ApplicationID);
		eid.setEntityID(ID);

		// m_setEntityTypeParameter();
		EntityType entityType = espdu.getEntityType();
		entityType.setEntityKind((short) 1); // Platform (vs lifeform, munition,
																	// sensor,
etc.)
		entityType.setDomain((short) 1); // Land (vs air, surface, subsurface,
																// space)
		entityType.setCountry(222); // 102 Iraq
		entityType.setCategory((short) 1); // Tank
		entityType.setSubcategory((short) 2); // M1 Abrams
		entityType.setSpecific((short) 1); // M1A2 Abrams
		entityType.setExtra((short) 1);

		/*
		 * EntityType alternativeEntityType = espdu.getAlternativeEntityType();
		 * alternativeEntityType.setEntityKind((short) 1); // Platform (vs
		 * lifeform, munition, sensor, etc.)
		 * alternativeEntityType.setDomain((short) 1); // Land (vs air, surface,
		 * subsurface, space) alternativeEntityType.setCountry(222); //102 Iraq
		 * alternativeEntityType.setCategory((short) 1); // Tank
		 * alternativeEntityType.setSubcategory((short) 2); // M1 Abrams
		 * alternativeEntityType.setSpecific((short) 1); // M1A2 Abrams
		 * alternativeEntityType.setExtra((short) 1);
		 */

		espdu.setTimestamp(DisTime.getInstance().getDisRelativeTimestamp());
		espdu.getMarking().setCharacterSet((short) 1); // ascii
		try {
			espdu.getMarking().setCharacters(marking.getBytes("US-ASCII"));
		} catch (UnsupportedEncodingException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		espdu.setEntityAppearance(m_appearance);

		m_location = espdu.getEntityLocation();
		m_location.setX(m_disCoordinates[0]);
		m_location.setY(m_disCoordinates[1]);
		m_location.setZ(m_disCoordinates[2]);

		if (m_orientation != null)
			espdu.setEntityOrientation(m_orientation);
		if (m_linearVelocity != null)
			espdu.setEntityLinearVelocity(m_linearVelocity);

		espdu.setLength(espdu.getMarshalledSize());
		return espdu;
	}

	public DisTime getM_disTime() {
		return m_disTime;
	}

	public void setM_disTime(DisTime m_disTime) {
		this.m_disTime = m_disTime;
	}

	public int getT14Count() {
		return m_count;
	}

	public EntityID getEntityID(EntityID eid) {
		if (eid == null)
			eid = new EntityID();
		eid.setApplicationID(this.ApplicationID);
		eid.setEntityID(this.ID);
		eid.setSiteID(this.SiteID);
		return eid;
	}

	/***** END EntityID ********/

	/***** END EntityType ********/

	/***** Location ********/
	public void setLocation(double lat, double lon, double alt, DataRepository dr) {
		m_disCoordinates = CoordinateConversions.getXYZfromLatLonDegrees(lat, lon, alt);
		double elev = alt;
		//System.out.println("T14.setLocation() " + m_disCoordinates[0] + " "+ m_disCoordinates[1] + " "+ m_disCoordinates[2]);
		elev = dr.getTerrainServer().getAltitude(m_disCoordinates[0], m_disCoordinates[1], m_disCoordinates[2]);
		m_disCoordinates = CoordinateConversions.getXYZfromLatLonDegrees(lat, lon, elev+3);
		//System.out.println("T14.setLocation() " + m_disCoordinates[0] + " "+ m_disCoordinates[1] + " "+ m_disCoordinates[2]);
```

```
}

public double[] getM_disCoordinates() {
                return m_disCoordinates;
}

public void setM_disCoordinates(double[] m_disCoordinates) {
                this.m_disCoordinates = m_disCoordinates;
}

public Vector3Double getM_location() {
                return m_location;
}

public void setM_location(Vector3Double m_location) {
                this.m_location = m_location;
}

public double getLat() {
                return m_location.getX();
}

public double getLon() {
                return m_location.getY();
}

public double getAlt() {
                return m_location.getZ();
}

public void fireTarget(FirePdu fire) {
                firedFlag = true;
                firePdus.add(fire);
}

public void detonationTarget(DetonationPdu detonation) {
                detonateFlag = true;
                detonatePdus.add(detonation);
}

/*
 * public void printLocation() { System.out .print(" EID=[" +
 * m_eid.getSiteID() + "," + m_eid.getApplicationID() + "," +
 * m_eid.getEntityID() + "]");
 * System.out.print(" DIS coordinates location=[" + m_location.getX() + ","
 * + m_location.getY() + "," + m_location.getZ() + "]"); double c[] = {
 * m_location.getX(), m_location.getY(), m_location.getZ() }; double lla[] =
 * CoordinateConversions.xyzToLatLonDegrees(c);
 * System.out.println(" Location (lat/lon/alt): [" + lla[0] + ", " + lla[1]
 * + ", " + lla[2] + "]"); }
 */
/***** END Location ********/

/***** orientation ********/
public EulerAngles getM_orientation() {
                return m_orientation;
}
```

```
public void setM_orientation(float phi, float psi, float theta) {
        if (m_orientation == null)
                m_orientation = new EulerAngles();
        m_orientation.setPhi(phi);
        m_orientation.setPsi(psi);
        m_orientation.setTheta(theta);
}

/***** END orientation ********/

/***** velocity ********/

public Vector3Float getM_linearVelocity() {
        return m_linearVelocity;
}

public void setM_linearVelocity(float xValue, float yValue, float zValue) {
        if (m_linearVelocity == null)
                m_linearVelocity = new Vector3Float();
        m_linearVelocity.setX(xValue);
        m_linearVelocity.setY(yValue);
        m_linearVelocity.setZ(zValue);

}

/***** END velocity ********/

/****** probability of hit function ***************/
private Random rng = new Random();
public boolean isTargetHit(double distance) {
        //god mode ;-)
        //if(true)
                //return true;
        double r1 = rng.nextDouble();
        double coefficient = 1;
        if (distance < 1000) { // unit is meter
                //coefficient = 0.089;
                double final_probability = 0.9 * coefficient;
                if (r1 < final_probability) {
                        return true;
                }

        } else if (distance < 2000) {
                //coefficient = 0.05;
                double final_probability = 0.8 * coefficient;
                if (r1 < final_probability) {
                        return true;
                }
        } else if (distance < 3000) {
                //coefficient = 0.027;
                double final_probability = 0.75 * coefficient;
                if (r1 < final_probability) {
                        return true;
                }
        } else if (distance < 4000) {
                //coefficient = 0.014;
                double final_probability = 0.7 * coefficient;
                if (r1 < final_probability) {
                        return true;
                }

        }

        return false;
}

/****** END PH function ***************/

/****** probability of kill function ***************/
public boolean isDead(double distance) {
        double r1 = rng.nextDouble();
        double coefficient = 1;
        if (distance < 2000) { // unit is meter
                // coefficient = 0.089;
                double final_probability = 0.8 * coefficient;
                if (r1 < final_probability) {
                        return true;
                }

        } else if (distance < 3000) {
                // coefficient = 0.05;
                double final_probability = 0.6 * coefficient;
                if (r1 < final_probability) {
                        return true;
                }
        } else if (distance < 4000) {
                // coefficient = 0.027;
                double final_probability = 0.30 * coefficient;
                if (r1 < final_probability) {
                        return true;
                }
        }
        return false;
}

/****** END PK function ***************/
/****** isSeeTarget *********************/
int skipCt = 10;
public boolean findNearestTarget(DataRepository dataObj) {
        double currentMax = MAX_FIRE_DISTANCE;
        String currentKey = "";

        if(skipCt!=0){
                skipCt--;
                return false;
        }

        //for (Enumeration<EntityStatePdu> e = dataObj.getRemoteEspdus().elements(); e.hasMoreElements();) {
        for (Enumeration<String> e = dataObj.getRemoteEspdus().keys(); e.hasMoreElements();) {
           String key = e.nextElement();
                EntityStatePdu temp = dataObj.getRemoteEspdus().get(key);
                if(temp==null)
                        break;
                        long killVar = temp.getEntityAppearance()>>3;
                        killVar = killVar&3;
                        if ((temp.getForceId() != this.ForceId) && (killVar!=3)) {
                            if(dataObj.getDeadReckonings().get(key) == null)
                            {
                                System.out.println("T14.findNearestTarget() " + key + " returned null");
                                System.out.println("T14.findNearestTarget() " + dataObj.getkey(temp.getEntityID()) + "
returned null");

                                System.exit(-1);
                            }

                            double[] r = dataObj.getDeadReckonings().get(key).getUpdatedPositionOrientation();
                            double[] lla = CoordinateConversions.xyzToLatLonDegrees(r);
                            double[] rVis = CoordinateConversions.getXYZfromLatLonDegrees(lla[0], lla[1], lla[2]+3);

                            // = CoordinateConversions.getXYZfromLatLonDegrees(lat, lon, alt);

                            if(!dataObj.getTerrainServer().canSee(m_disCoordinates[0], m_disCoordinates[1],
m_disCoordinates[2], rVis[0], rVis[1], rVis[2]))
                                continue;

                            double distance = getRange(r);

                            if ((distance > MIN_FIRE_DISTANCE) && (distance < MAX_FIRE_DISTANCE))
                                if (distance < currentMax) {
                                    currentMax = distance;
                                    aimed_target = temp;
                                    currentKey = key;
                                    //System.out.println("T14.findNearestTarget() found  " + name(aimed_target)
+ " at " + distance);
                                }
                        }
                    }
                    if(skipCt==0)
                        skipCt = 10;
                    if(aimed_target != null)
                        System.out.println("T14.findNearestTarget() found  " + name(aimed_target) + "("+currentKey+") at " +
currentMax + " - " + aimed_target.getEntityID().getEntityID());
                    return aimed_target != null;
            }// seeTarget

            private String name(EntityStatePdu esp) {
                    return new String(esp.getMarking().getCharacters(), Charset.forName("US-ASCII")).trim();
            }

} // end T14 class
```

Terrain Server

```
// Terrain Server.cpp : main project file.

//#include "stdafx.h"

/*******************************************************************************
** Copyright (c) 2004 MAK Technologies, Inc.
** All rights reserved.
*******************************************************************************/


//This example demonstrates how to use the DtCgf class to create objects
//and task entities, as you would if you were to embed VR-Forces in your own
//application.

#ifdef WIN32
#include <winsock2.h>
#include <windows.h>
#endif

//VR-Forces Headers
#include "vrfcgf/cgf.h"
#include "vrfcgf/cgfDispatcher.h"
#include "vrfcore/simManager.h"
#include "vrftasks/moveToTask.h"
#include "vrfobjcore/vrfObject.h"

//VR-Link headers
#include <vl/exerciseConn.h>
#include <vl/exerciseConnInitializer.h>
#include <vlutil/vlPrint.h>
#include <vlutil/vlProcessControl.h>
#include <vl/hostStructs.h>
#include "vrfutil/vrfVlKeyboard.h"

#include "vrfobjcore/physicalWorld.h"
#include "terrainInterface/terrainInterface.h"


//Window headers
#include <winsock2.h>
#include <winsock.h>
#include <iostream>
#include <stdio.h>
#include <windows.h>
```

```
#using "System.dll"

//.NET
using namespace System;
using namespace System::Net;
using namespace System::Net::Sockets;
using namespace System::Text;
using namespace System::IO;


//Function to find closest terrain intesection above or below a point
String ^ClosestTerrainAltitude(DtVector& localLocation, DtCgf* cgf)
{
   bool blocked = FALSE;
   bool dataAvailable = FALSE;
   double height;
   DtChordIntersectionRecord record;
   DtIntersectionRecord::DtIntersectionType irtFlag = DtIntersectionRecord::INT_ALL_DATA
          ;

   while (!dataAvailable)
   {
          blocked = cgf->physicalWorld()->terrainInterface()->terrainHeightAndIntersection(localLocation, height, record,
irtFlag, &dataAvailable);
          if (blocked) std::cout << "Altitude above sea level at surface: " << height << '\t' << dataAvailable << '\n';
          // Must tick terrain interface to get background jobs running
          cgf->tick();
   }

   String ^strHeight = height.ToString();
   if (!dataAvailable)strHeight = "NOTAVAILABLE";
   Console::WriteLine(strHeight);
   return strHeight;
}

DtPoint ClosestTerrainIntPoint(DtVector& localLocation, DtCgf* cgf)
{
   bool blocked = FALSE;
   bool dataAvailable = FALSE;
   double height;
   DtChordIntersectionRecord record;
   DtIntersectionRecord::DtIntersectionType irtFlag = DtIntersectionRecord::INT_ALL_DATA;

   while (!blocked && !dataAvailable)
   {
          blocked = cgf->physicalWorld()->terrainInterface()->terrainHeightAndIntersection(localLocation, height, record,
irtFlag, &dataAvailable);

          // Must tick terrain interface to get background jobs running
          cgf->tick();
```

```
    }
    DtPoint intersectionPoint = record.intersectionPoint();
    return intersectionPoint;
}

String ^interVisible(DtPoint& p1, DtPoint& p2, DtCgf* cgf)
{
    DtChord chord(p1, p2);
    DtPoint isectPoint;
    double intersectionTime;
    bool blocked = FALSE;
    bool dataAvailable = FALSE;
    int counter = 0;

    //while (!blocked && !dataAvailable)
    while (counter < 20 && !dataAvailable  && !blocked)
    {
            counter++;
            blocked = cgf->physicalWorld()->terrainInterface()->intersect(chord, isectPoint, intersectionTime, &dataAvailable);
            std::cout << blocked << '\t' << isectPoint << '\t' << intersectionTime << '\t' << '\t' << counter << '\t' << dataAvailable
<< '\n';

            // Must tick terrain interface to get background jobs running
            cgf->tick();
    }
    bool visible = !blocked;
    String ^strVisible = visible.ToString();
    if (!dataAvailable && !blocked) strVisible = "NOTAVAILABLE";
    Console::WriteLine(strVisible);
    return strVisible;
}

void processMessage(String ^line, TextWriter ^connectionWriter, DtCgf* cgf)
{
    String^ delimStr = " ";
    array<Char>^ delimiter = delimStr->ToCharArray();
    Console::WriteLine(line);

    if (line->StartsWith("ELEVATION"))
    {
            double x, y, z;
            String ^str = line;
            array<String^>^ splits = str->Split(delimiter, StringSplitOptions::RemoveEmptyEntries);
            Double::TryParse(splits[1], x);
            Double::TryParse(splits[2], y);
            Double::TryParse(splits[3], z);

            DtVector p(x, y, z);
            String ^strHeight = ClosestTerrainAltitude(p, cgf);
            Console::WriteLine(line);

            connectionWriter->WriteLine(strHeight);
            connectionWriter->Flush();
    }

    else if (line->StartsWith("CANSEE"))
    {
            double x1, y1, z1, x2, y2, z2;
            String ^str = line;
            array<String^>^ splits = str->Split(delimiter, StringSplitOptions::RemoveEmptyEntries);
            Double::TryParse(splits[1], x1);
            Double::TryParse(splits[2], y1);
            Double::TryParse(splits[3], z1);
            Double::TryParse(splits[4], x2);
            Double::TryParse(splits[5], y2);
            Double::TryParse(splits[6], z2);

            DtPoint p1(x1, y1, z1);
            DtPoint p2(x2, y2, z2);

            String ^strVisible = interVisible(p1, p2, cgf);

            connectionWriter->WriteLine(strVisible);
            connectionWriter->Flush();
    }
    //if (!(line->StartsWith("CANSEE")) && !(line->StartsWith("ELEVATION")))
    else
    {
            //String ^str = line;
            //array<String^>^ splits = str->Split(delimiter, StringSplitOptions::RemoveEmptyEntries);
            connectionWriter->WriteLine("Error.  You sent: ", line);
            connectionWriter->Flush();
    }

}// processMessage


//int main(int argc, char *argv[])
int main(array<System::String ^> ^args)
//int main()
{
    String^ start_param = String::Concat(args[0]);
    Console::WriteLine(start_param);

    //DIS/RPR simulation address

    DtSimulationAddress simAddress(1, 3095);

    //Create the CGF
```

```
DtCgf* cgf = new DtCgf(simAddress);



int port = 3928;                    //UDP port number
int exerciseId = 1;                 //DIS exercise ID

DtExerciseConn* exerciseConn = new DtExerciseConn(port, DtInetAddr("127.255.255.255"), exerciseId,
        simAddress.siteId(), simAddress.applicationId());

//Initialize the CGF
cgf->init(exerciseConn);

//Create the dispatcher so this application can interface with
//front-end
DtCgfDispatcher* cgfDispatcher = new DtCgfDispatcher(cgf,
        cgf->simManager()->messageInterface());

//Create a new scenario, using the Ground-db.gdb terrain database and the EntityLevel
//simulation model set to allow for entity creation
//cgf->newScenario("../userData/terrains/VR-TheWorld Online - MAK Earth - Base.mtf",
//    "../data/simulationModelSets/EntityLevel.sms");

cgf->newScenario("../userData/terrains/73 Easting v2.mtf",
        "../data/simulationModelSets/EntityLevel.sms");
//Tick the cgf to get the terrain loaded
cgf->tick();

// Set up chord for intersection test. Query starts at p1 and ends at p2.
//DtPoint p1(3819717.54686021, 4031218.60673729, 3126807.93341056);
//DtPoint p3(3818296.01076867, 4032287.5276991, 3127159.4084278);

//DtPoint p1(3817496.7401120, 4037113.3755965, 3121932);
//DtPoint p3(3817004.7747151, 4038369.3141057, 3120944);
DtPoint p1(3817515.5630267, 4037114.6118975, 3121935);
//DtPoint p3(3817855.0150201, 4036753.7006537, 3121987);
DtPoint p3(3817940.9117024885, 4036729.835147949, 3123425.9327854933);
//3817492.3980511194, 4036255.6192080416, 3123056.5349689415
//DtPoint p2(0, 0, 0);
DtChord chord(p1, p3);
DtVector v1(3817940.9117024885, 4036729.835147949, 3123425.9327854933);
DtVector v2(3817492.3980511194, 4036255.6192080416, 3123056.5349689415);
DtPoint p2(3817492.3980511194, 4036255.6192080416, 3123056.5349689415);

DtPoint isectPoint;
double intersectionTime;
bool blocked = FALSE;
bool blocked2 = FALSE;
bool dataAvailable = FALSE;
```

```
    double lat = 29.537826;
    double lon = 46.559891;

    DtPoint intersectionPoint;


    dataAvailable = false;

    while (!dataAvailable)
    {
            double altitude = cgf->physicalWorld()->terrainInterface()->terrainHeightAboveSeaLevel(p3, &dataAvailable);

            if (dataAvailable)
            {
                    std::cout << "Altitude of terrain at " << p3 << " is:" << '\t' << altitude << '\n';
                    std::cout << "Altitude of terrain at " << p2 << " is:" << '\t' << altitude << '\n';
            }
            else
            {
                    // Must tick terrain interface to get background jobs running
                    cgf->tick();
            }
    }

    dataAvailable = false;

    int counter = 0;

    std::cout << "blocked" << '\t' << "isectPoint" << '\t' << "intTime" << '\t' << "counter" << '\t' << "dataAvailable" << '\n';

    while (!blocked && !dataAvailable)
    {
            counter++;
            blocked = cgf->physicalWorld()->terrainInterface()->intersect(chord, isectPoint, intersectionTime, &dataAvailable);
            std::cout << blocked << '\t' << isectPoint << '\t' << intersectionTime << '\t' << '\t' << counter << '\t' << dataAvailable
<< '\n';

            // Must tick terrain interface to get background jobs running
            cgf->tick();
    }


    bool connection_flag = true;

    do {
            try
            {
                    IPAddress ^ipAd = IPAddress::Parse(start_param);
                    // Initializes the Listener
                    TcpListener ^myList = gcnew TcpListener(ipAd, 8001);

                    // Start Listeneting at the specified port
                    myList->Start();

                    Console::WriteLine("The server is running at port 8001...");
                    Console::WriteLine("The local End point is  :" + myList->LocalEndpoint);
                    Console::WriteLine("Waiting for a connection.....");

                    Socket ^connection = myList->AcceptSocket();
                    Console::WriteLine("Connection accepted from " + connection->RemoteEndPoint);
                    NetworkStream ^connectionReadStream = gcnew NetworkStream(connection, FileAccess::Read, false);
                    NetworkStream ^connectionWriteStream = gcnew NetworkStream(connection, FileAccess::Write, false);
                    TextReader ^connectionReader = gcnew StreamReader(connectionReadStream, Encoding::UTF8);
                    TextWriter ^connectionWriter = gcnew StreamWriter(connectionWriteStream, Encoding::UTF8);
                    String ^line = nullptr;
                    while (nullptr != (line = connectionReader->ReadLine()))
                    {
                              line = line->Trim();

                              if (String::Equals("exit", line, StringComparison::InvariantCultureIgnoreCase)) connection_flag =
false;
                              if (String::Equals("quit", line, StringComparison::InvariantCultureIgnoreCase))connection_flag =
false;
                              if (String::Equals("goodbye", line,
StringComparison::InvariantCultureIgnoreCase))connection_flag = false;
                              if (String::Equals("good-bye", line, StringComparison::InvariantCultureIgnoreCase))
connection_flag = false;

                              if (connection_flag == false) break;

                              processMessage(line, connectionWriter, cgf);

                    }

                    connection->Shutdown(SocketShutdown::Both);
                    connection->Close();
                    myList->Stop();

          }
          catch (Exception^ e)
          {
                    Console::WriteLine("Error..... " + e->StackTrace);
          }

  } while (connection_flag);

  delete cgf;
  delete exerciseConn;
```

```
    system("PAUSE");
    return 0;
}
```