3D Battlefield Modeling and Simulation of War Games

Baki Koyuncu, Erkan Bostancı

***Abstract*—** In this study, a real time 3D virtual simulation software for visualizing the military battlefields was developed. Developed software, named Sandbox, used elevation data stored in DEM format corresponding to the battlefield. Sandbox uses this data to create the platform on which the military units will be added. Different military units can be added in the software. Military units were viewed by both using a recent military symbology, NATO-APP-6A, and 3D models of the real military units. Software uses translation animation for the position updates. Since data transmission between different platforms was considered, developed software extensively uses XML based data. A database was used for long term storage of received reports. Web services were used to transmit and receive reports to/from remote field observers and change the state of the software.

***Keywords*—**3D Battlefield Visualization, Sandbox, Military Decision Making, XNA, Database, MS SQL Server, Web Services

I. Introduction

IN a crisis environment, commanders have to view the situation in a clear and accurate way, identify a winner strategy and then act accordingly. This requires a good perception of the battlefield. Technological advances in software provide new opportunities for this issue.

In today's modern warfare, the uncertainty of battlefield is much greater than it was before. Battlefields have spread to very wide areas of ground and the importance of military intelligence has increased tremendously. Consequently classical map techniques, in which a printed map of the battlefield was used to view the battle-space, had proven to be insufficient.

Commanders needed [1] efficient visualization systems to view the most recent data about the current military situations in order to make efficient decisions that would lead to victory. With these needs in mind, a virtual military sandbox software was developed for modeling and simulation of battlefield and war games.

Manuscript received December 5, 2009.

B.K. Author is head of the Department of Computer Engineering in Ankara University. Ankara, 06500, Turkey (e-mail: bkoyuncu@ankara.edu.tr)

E. B. Author is with the Department of Computer Engineering in Ankara University. Ankara, 06500, Turkey. (corresponding author to provide phone: +90-536-633-8620; e-mail: erkanbostanci@gmail.com).

II. Development

The study was based on obtaining 3D digital geographical data, visualizing military units with their real time information and transmitting this information for other remote decision making mechanisms.

An important issue here is using a recent and common format for entity representation in the system. For this reason, a common and well documented symbology (NATO-APP-6A) was used in this study.

The emphasis was made on visualization in this study therefore it is aimed to include features such as modeling the terrain data obtained with different textures representing grass, desert or snow terrain; viewing the battlefield from different directions, zooming-in/out and using two different camera angles (vertical, oblique and mobile).

In addition to the features mentioned, for more efficient visualization some 3D models of the military units were introduced to the system. This both decreased the time spent for learning the system and resulted a better view of the current military situation.

Web service features were implemented in the system to enhance the communication between commanders. This feature prevented the decision making process from being one-dimensional.

A. Graphical System

A Window's forms application was developed to visualize the battlefield. In order to draw the battlefield and the military units XNA was employed [2]. XNA is a graphics framework and is used with C# programming language.

1) Basic View

In this study, military units were displayed in the battlefield by using military symbology named NATO-APP-6-A [3]. This symbology included the necessary information for each military unit such as unit type, affiliation, position information and report date and time as given in Fig.1. In a previous implementation simple geometric shapes were used for modeling military units and it was concentrated more on the efficient visualization of the terrain [4].

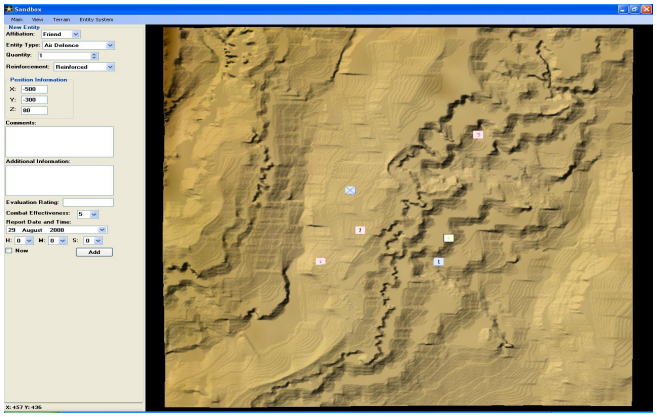

Fig.1 Objects were displayed as symbols in the battlefield in the basic software

Battlefields were modeled with the data obtained from USGS web site in DEM format. A specific file-parser class named DEMFile was implemented for this type of data. This parser was inherited from a more general Elevation-Data class which included the general information about regional elevation data and elevation limits.

The connection between the military objects and their representation are shown in Fig. 2. An entity object has symbol and affiliation objects in it. These objects ease the handling of each entity, for instance Affiliation class stores both the affiliation information itself and the colour used to represent the affiliation type. Similarly, SymbolInformation class stores the image file used as a texture to view the entity.

A separate class named SymbolImageManager is implemented to store all the images used in the symbology so that the user does not have to wait for the loading of the image in run time when he/she tries to update the symbol.

In the implementation, unit objects also included a Shape type object in order to be viewed in the graphical part. This object stores information about a simple box on which the symbology will be applied as a texture. This object stores transformation information. Transformation information included translation, scaling and rotation information about the model. In addition, necessary matrices such as view, projection and world for the rendering of all units are stored in the object.

Similarly, an Entity object stores an object of type Shape3D. This class is very similar to the Shape object; however it stores a 3D object of the military unit instead of a simple box. With these classes provided, both the symbology and the 3D model of the units can be displayed.

In order to handle the entities created either by user entry or remote information in a compact manner, a container class name EntityManager was created. This class provided the basic operations needed to access a selected entity by using its identification. In other words it is the EntityManager object in the system which was called as the “Entity System”.

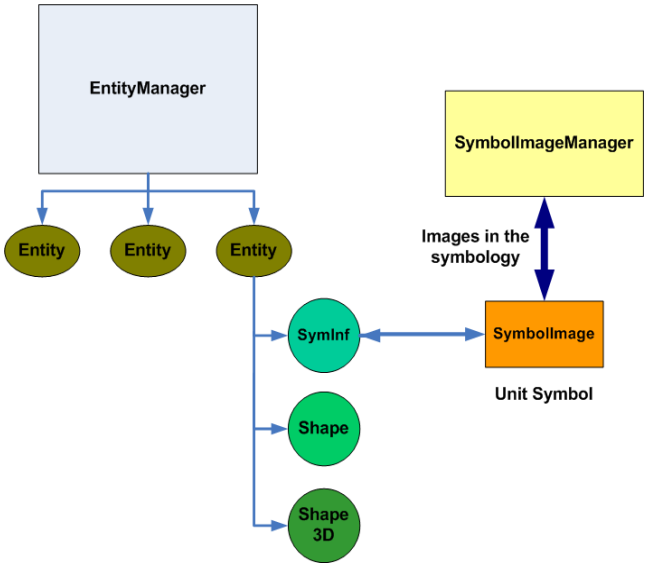

Fig.2 Entity system

In this container class, a SortedList type of data structure was used to increase the retrieval speed for an entity. An entity is looked-up in the list, given its identification, and returned as the active entity when the user selected the entity from the user interface by using mouse clicks.

2) Inclusion of Military Models

In addition to the military symbology, 3D models of the military units were added to the software in order to improve reality. Hence, the visual quality was improved and learning of the symbology was simplified.

All the models were edited in the 3D Studio Max software and their polygon numbers were optimized. Here the methods used previously were used. Using the MultiRes modifier, polygon numbers were reduced.

Later the models were converted in FBX format which can be used in XNA. This conversion process required the following steps:

1. Original model was copied to the Content folder of the project.
2. Then the model is imported by the 3D Max software.
3. The model is exported in FBX format by using the Panda plug-in.
4. Models texture sizes are adjusted to be 2n x 2n.
5. Finally, processed textures are copied to the model’s folder.

This process allowed the models to be used in XNA easily.

Before the 3D models are viewed by the software, the differences between the models arising from modeling process were minimized. An important size difference was identified as the scale inconsistency. For example, a jeep model can be few times greater than a tank model if no scaling is applied to one of them or both. This difference exists since different modelers do not use a common scale while generating their 3D models. Appropriate scale levels were introduced by bringing several models together and identifying their relative sizes. These processes had to be performed manually for both single

and groups of models which constituted one of the most time consuming parts in the study.

In addition to scaling, a default heading vector for all models was considered as the positive z axis (0, 0, 1). This heading vector was used in the rotation of the models according to their new positions entered by the user. After this operation the default rotation angles were specified for the models. These angles will be the initial rotation for the calculations those will be performed when a new position for the unit is entered.

After the operations mentioned above are performed, the optimal transformations were obtained.

With the default scale and rotations specified, the models can be viewed on the battlefield according to its symbol. User can select between the symbol or the 3D models to view the military unit by using a pop-up menu.

A class named Shape3D was designed to manipulate the 3D models. This class worked as an element of the Entity class. It stored the necessary transformation information in order to view the model in the exact position entered by the user. Major classes used in Sandbox are summarized as shown in Fig.3.

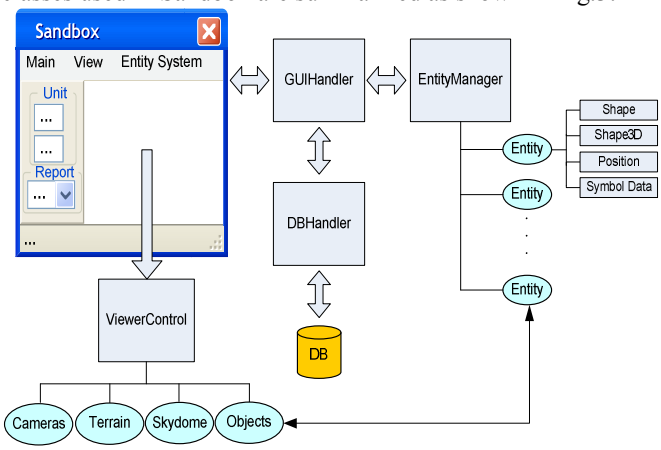

Fig.3 Summary of classes used in the software

3) Determination of an Object's New Directional Vector

An object has an initial position and a vectoral motional direction as shown in Fig.3. When a new positional data for this object is given by the user, the object has to rotate to head to this new position. This rotation will result a realistic animation of the models when the models are translated to their new positions.

The angle required for rotation is calculated by using dot product between two vectors. These motional vectors are shown in Fig.4.

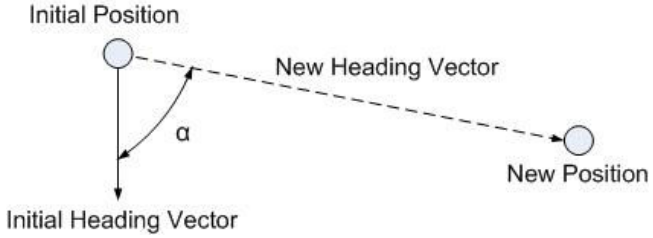

Fig.4 Positions, heading vectors and rotation angle

Given the initial heading vector and the new heading vector, the dot product between the two vectors is given by:

$$a \cdot b = |a||b| \cos \alpha$$

$|a|$ and $|b|$ denote the lengths of the vectors a and b. α is the angle of rotation. Then α becomes:

$$\alpha = \arccos\left(\frac{a \cdot b}{|a||b|}\right)$$

The following code segment shows how this formula is implemented in the software:

```
position2D = new Vector2(position.X,
   position.Z);
newPosition2D = new
   Vector2(newPosition.X,
   newPosition.Z);
newDirection2D =
   Vector2.Subtract(newPosition2D,
   position2D);
cosAlpha = Vector2.Dot(newDirection2D,
   heading2D) / (newDirection2D.Length()
   * heading2D.Length());
alpha =
   MathHelper.ToDegrees((float)Math.Acos
   (cosAlpha));
```

Once the rotation angle is obtained, the software needs to know in which direction the rotation will be applied. This is simply decided by checking the X coordinate of the new position. If this value is positive, the rotation will be in clockwise direction and the rotation will be in opposite direction otherwise.

After these calculations, the model will be aimed to the new position as shown in Fig. 5.

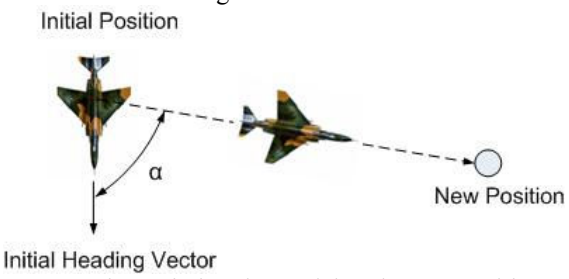

Fig.5 Aiming the model to the new position

4) Position Update Animation

In [5], selected units appeared directly in the new position when an update is performed. This can be acceptable for viewing the last position but not sufficient to estimate the direction of its movement.

In this study, update operation is performed by the UpdateEntities() method of EntityManager class . This method checks all the objects stored in the software one by one and updates the positions if necessary.

For the movement (translation animation) of the displayed objects, the graphics component is refreshed repetitively. In XNA applications, an Update() method is readily provided for

this purpose [7]. This method is used to perform the changes for the scene to be redrawn.

The following assignment was used to allow the graphics component to refresh itself repetitively:

```
Application.Idle += delegate
{Invalidate();}
```

The graphics component will draw the scene continuously as long as there is no event to be processed. UpdateEntities() will calculate the positions of the objects in every refreshment according to their initial and new positional data. Linear interpolation is used to find the points between these two limits of movement.

Linear interpolation is implemented by using the Lerp (…) method of the MathHelper class provided by XNA. In order to calculate the final position of an object; initial position, new position and the amount of increment that will be used as a step size are considered.

The formula used to calculate the final position by linear interpolation is as follows:

$$initialPosition + (newPosition - initialPosition) * increment$$

The increment value is between 0 and 1. When the increment is 0, formula gives initial position and when increment is 1, formula gives the new position. The code segment implementation of the calculated position is given as:

```
calculatedPosition.X =
   MathHelper.Lerp(position.X,
   newPosition.X, 0.01f);
calculatedPosition.Z =
   MathHelper.Lerp(position.Z,
   newPosition.Z, 0.01f);
```

calculatedPosition stores the position obtained after each call to the Lerp(…) method. Since the update is performed continuously, calculatedPosition approaches to the newPosition by small increments and stops when it reaches. This approach yields a gradual animated motion of military units in the scenario.

B. Military Unit Data Storage

The developed Sandbox simulation software has facilitated the storing of the military unit data. This will allow the users to save some specific situations in the database and later query the objects according to some criteria such as object identification. Hence the data stored, in a way, will act as military intelligence for future considerations. Microsoft SQL Server 2005 was used to design and develop the relational database [8].

1) Database Design

The design of the database was planned according to a scenario concept that is embedded in the software. Each military situation is viewed as a new scenario and stored in the data base. This can be briefly explained as follows:

A military situation consists of a battlefield and the military units in it. These military units have to store enough information about a real world event using real world data in order to perform the visualization. Therefore the database design followed the same notation with the military symbology mentioned earlier.

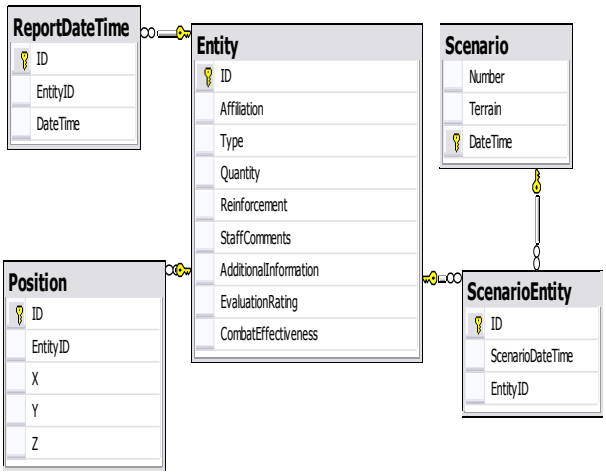

Fig.6 Diagram of relations between the tables in the database

A specific military situation is stored in the Scenario table with a) its unique identification given with the DateTime table column and b) the battle field terrain table column as shown in Fig.6. Entity is the table which stores the information of a military unit in data base.

The table adopted the military symbology with their fields such as Affiliation, Type, and Quantity etc Report times and positions of military units were stored in the ReportDate-Time and Position tables respectively. The mapping between a scenario and the entities given in that scenario was stored in the ScenarioEntity table.

2) Entity Query

Access to the database was implemented using the ADO.NET API technology which allows accessing database programmatically. A separate form was designed to view the stored entities. This form enables users to access the military data directly and update it.

The approach followed in the software is also known as data binding [11] where the data in the database is accessed through the user interface components. A DataGridView was used to view the data retrieved from the database in a table. This control requires a DataSource object from which the data will be obtained in order to be displayed. Several queries were generated with these data sources in the implementation. The mapping between the data table and the data sources was performed by using TableAdapter class in order to execute the queries on the data source and return the results to the data table.

Three TableAdapters were used to provide access to Entity, Position, ReportDateTime tables. Each adapter has one or more queries to execute. A sample query used in the soft-ware was shown below:

```
SELECT ReportDateTime.EntityID,
   ReportDateTime.DateTime, Entity.Type,
   Entity.StaffComments
FROM ReportDateTime, Entity
 WHERE(ReportDateTime.EntityID =
@entityID AND Entity.ID =@entityID)
```

In the given SQL statement, data from ReportDateTime and Entity tables are retrieved as the query result. ID is the primary key in the Entity table and is a foreign key in ReportDateTime table. Therefore it was checked against the given entityID variable. The '@' symbol denotes that entityID is a variable will receive its value in the runtime from the corresponding method of the table adapter.

C. System Update Using Web Services

Developed software allows exchanging reports from field observers to affect the system in addition to the input made by a single computer. Web services were used to communicate the two tiers named client and server. This architecture is depicted in Fig.7.

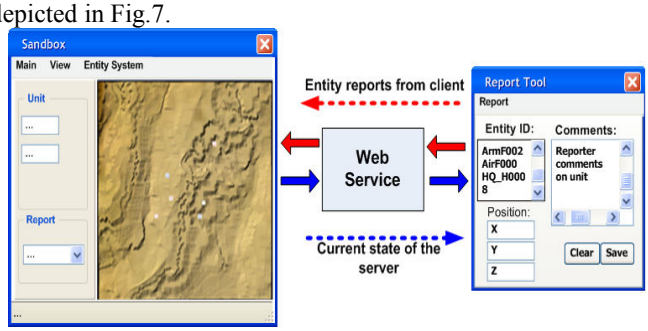

Fig. 7. Communicating two applications via web services

The web service resides in the server part. This server does the necessary communication between the main application and the clients. All the communication is made using Communicate web service. This service does the two-way communication from client to server and from server to client. In the first direction, new information for the military units from observers in the field is sent. The opposite direction server sends the current state (which entities are available in the system) by both the identities of the units and a snapshot of the visualized battlefield.

In order to send unit data with this service, two user-defined classes named EntityData and Report are created. The former class stores data about a military unit such as unit identity, position information, affiliation and reporter comments on the unit. The latter class stores a dynamic list of EntityData's and additionally stores time information about the report. Time is added to the report just before the report is sent from the client.

The service has several methods to send/receive the reports from/to client to/from the server. Fig. 8 shows how the reports are transferred both ways in XML format. ReportList tag shown in the figure constitutes a parent node for several military units. Each units position, affiliation and comments on this unit are transferred. Following the list time data exists to denote the time when the report is sent.

```
POST /CommunicateService/Service.asmx/GetReport HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<Report xmlns="http://tempuri.org/">
  <ReportList>
    <EntityData>
      <ID>string</ID>
      <X>int</X>
      <Y>int</Y>
      <Z>int</Z>
      <Affiliation>string</Affiliation>
      <Comments>string</Comments>
    </EntityData>
    <EntityData>
      <ID>string</ID>
      <X>int</X>
      <Y>int</Y>
      <Z>int</Z>
      <Affiliation>string</Affiliation>
      <Comments>string</Comments>
    </EntityData>
  </ReportList>
  <Datetime>dateTime</Datetime>
</Report>
```

Fig. 8 Report in XML format

Sandbox sends its current state to the observers in a frequency of ten seconds. This information is sent in form of an image which is obtained by changing the default rendering target to an image target instead of the user screen for a while. This will allow the instant view to be copied on an image. The image will be sent to the clients using the web service.

With the web service introduced, Sandbox software will be able to view the most recent information about the battlefield obtained from remote observers.

III. Results

Developed software allows users to open a USGS DEM terrain file and displays the terrain from different view-points such as vertical and orthographical views. The battlefield can be displayed using one of grass, sand or snow textures in order to model the terrain conditions in the battlefield.

Military units can be added to the battlefield by using the "Entity System" menu. This menu provides several features for addition, update or deletion of these units from the software.

The objects can be shown using both the symbology mentioned and the 3d models of military units. Users are able to select any unit from the graphical part and display its data on the left panel. This panel is also used to update the selected unit's data. Changes are reflected on the graphical part in real time. Especially when the user updates the position information for an object, the object moves to the new position in an acceptably realistic manner.

A sample output of the software is given in Fig.9 where the units are positioned in the battlefield. A tank battalion is located on the north-east and an enemy aircraft is approaching towards them. Artilleries are positioned on the east and a reconnaissance jeep is gathering field information on the west.

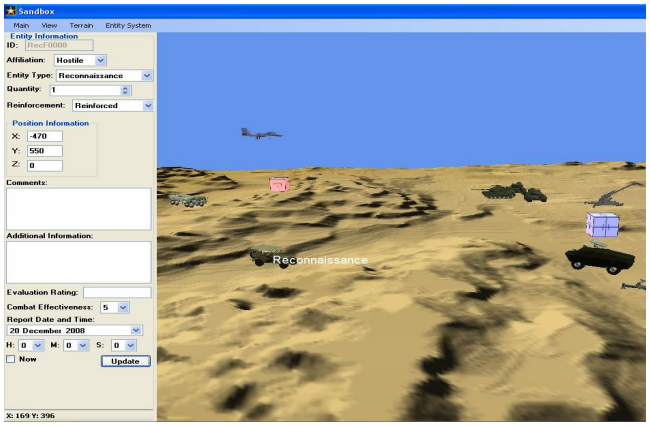

Fig.9 Military units in the battlefield

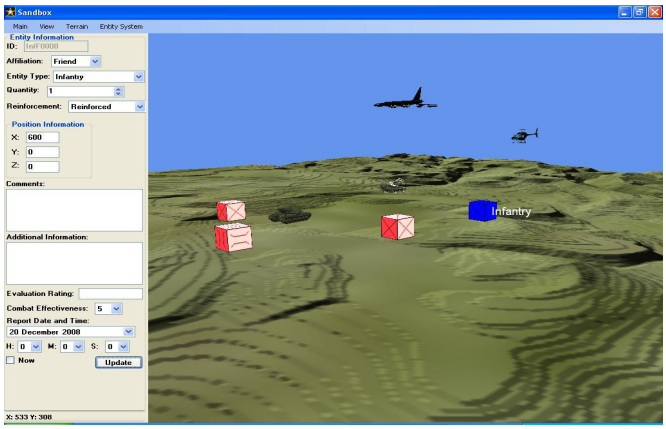

Fig.10 Displaying the battlefield in different scenario

A different war scenario is given in Fig.10. An air-defense unit is targeting a combat supply plane and a helicopter is delivering direct supporting fire to this unit. Infantries in the battlefield are simply shown by the symbology (boxes with cross on them).

A database was introduced in the developed software in order to store the data about military units. Additionally, scenarios are also saved in XML data format as proposed in [12].

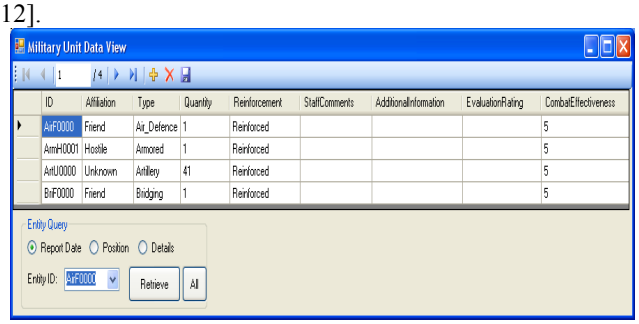

Fig.11 Data view provided to access the database

Fig.11 shows the data view part of the software which allows direct access to the database within the application. Users are able to make several queries about a specific unit. Results of the queries are displayed in a data table.

One of the most important parts of the software is certainly updating the system by reports received from remote observers. A sample scenario is given as Fig.12 in order to explain this usage. Three reconnaissance jeeps are collecting field information; however they get close to the enemy lines.

The report tool is given in Fig.13 and Fig.14. This tool is updated by the web service frequently. The tool has two views. First is the part where reports are entered, and the second part is where the view from the Sandbox software is presented.

Field observer who learned that the enemy tanks are aware of the jeeps reports this situation using the report tool as in Fig.15. Reports can be sent for any number of units. Save button is clicked for each unit report.

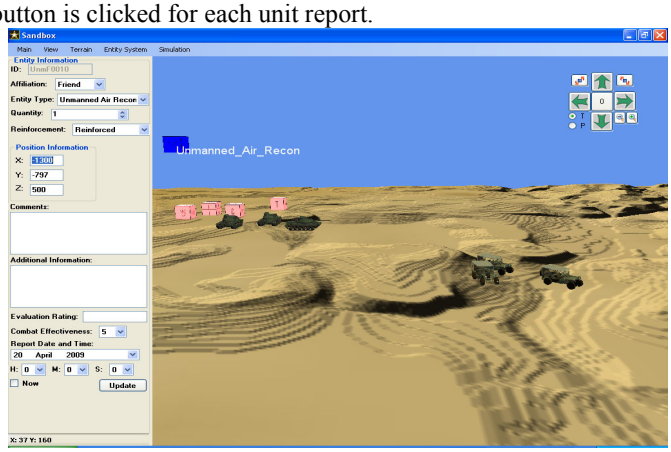

Fig. 12 A sample beginning scenario for the report tool usage

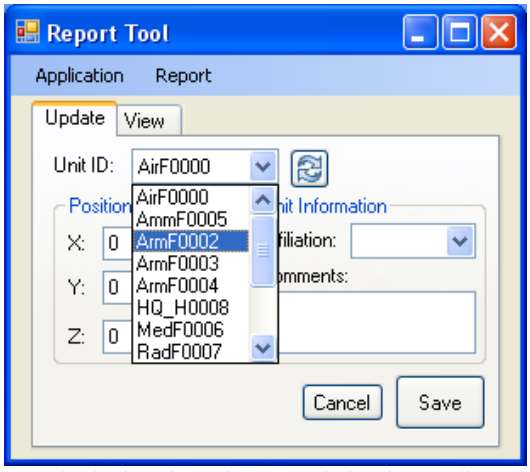

Fig. 13. Displaying the unit IDs (existing in Sandbox) in the report tool

"Send Report" option of the Report menu is clicked in order to send the report for all units. Hence, the report is send via the web service to the Sandbox software.

When Sandbox software receives the reports, it updates the units indicated in the report and starts animating these units in the graphical part. This update is reflected to the battlefield as in Fig.16.

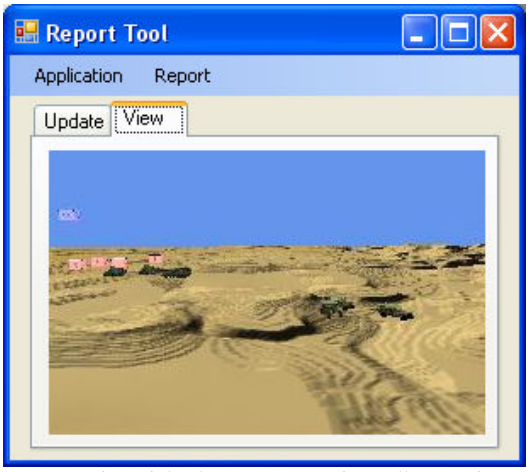

Fig.14. Tranfer of the latest status of Sandbox software to report tool

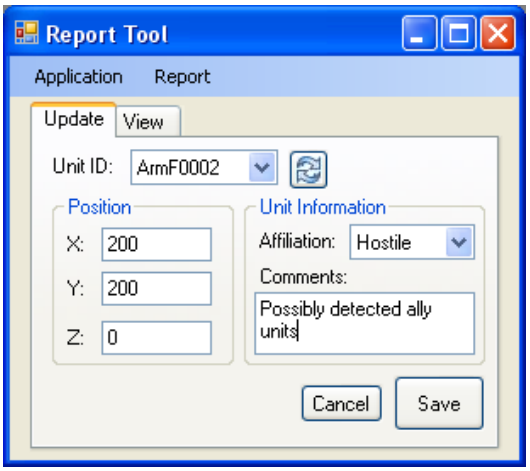

Fig.15 Reporting new positions and comments for the selected unit

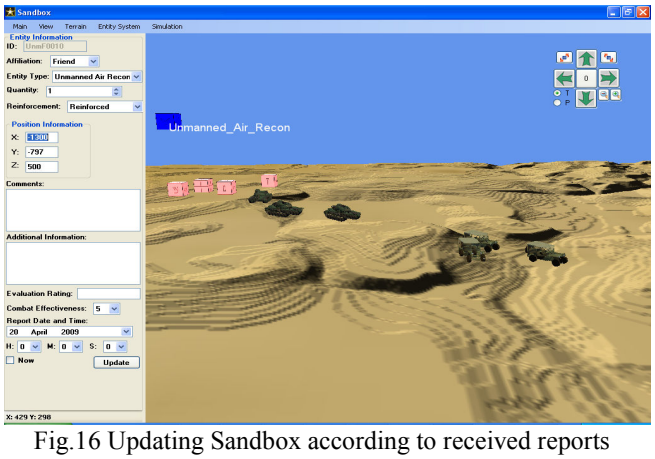

Fig.16 Updating Sandbox according to received reports

This panel shown in Fig.17 is the navigation panel that is visible on the upper right corner of the software. This tool allows users to see the battlefield from any direction and zoom levels. Angle of the camera and the camera positions can be changed here. The battlefield can be rotated to right and left. Also the zoom in and out can be performed by the users.

Fig.17 Navigation panel

IV. Conclusion

Developed software will facilitate the decision making processes with received intelligence. A realistic visualization of the battlefield will allow the commanders to perceive the current situation in a more efficient manner.

In previous work, the military units were represented with simple geometrics shapes [4]. This notation is not sufficient to represent all types of military units although how many geometric shapes are used. In the thesis study, a subset of the NATO-APP-6A symbology developed by [3], was used in order to view a wide range of military unit types.

The software extensively uses XML format in data transmission as proposed by [12]. This usage will facilitate the data transmission between different platforms and will allow changing the data format in an easy manner. Received reports from remote field observers are added to the system with the usage of web services. In addition, the current state of the simulation software is transmitted to these observers and checked by them. Therefore Sandbox simulation software is compliant with the requirements of a modelling and simulation environment specified in [19].

The software developed is not very complex in structure it has a comprehensive coverage of the military information. In addition, the flexibility of the software will allow responding to future requirements. Although this study tried to cover many technologies for battlefield management, this area is a continuously improving and open-ended in integrating the most recent technologies. These suggestions presented here may guide the future work:

• More 3D models can be added to the system in order to provide a more realistic virtual environment. Architectural models, some earth features will help in displaying the battle field in a greater detail. In addition, for some operations such as hostage rescue, whole model of a city can be introduced.

• Clash effects and explosions may be introduced. The level of adding these effects depends on the purpose of development. It is both possible to develop a strategy game working with real world data and a tool for decision making with some limited animation features.

• Web services can be used to transmit reports from several types of mobile devices such as smart phone or PDA. With the applications that will be developed, in addition to the remote observers even the war-fighters will be able to report the most recent situation to the system. Similarly technologies presented in [18] can be used as well.

• BBN (Bayesian Belief Networks) is an efficient modeling tool to construct cause and effect relations in many different areas in addition to other decision making tools as described in [15], [16] and [17]. If some new information is present and the reliability of this information is not certain, then it will be possible to make decisions on this information. For instance, if it is known that a military unit exists in the field. Then if an activity report is received for this unit, it is possible to construct a decision mechanism according to the reliability of this report. Sample BBN for this situation is given in Fig. 18.

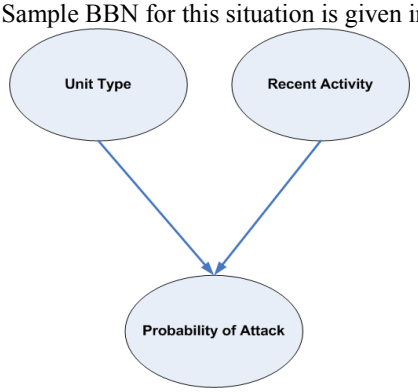

Fig. 18 Suggested BBN model

Finally, Sandbox will bring new initiatives for battlefield management and for simulation of the war games by using both the visual features and the advanced data handling capabilities of the developed software. The software and its source code can be accessed freely from http://comp.eng.ankara.edu.tr

REFERENCES

[1] Taylor G, Wood S and Knudsen K, "Enabling Battlefield Visualization: An Agent Based Information Management Approach", 10th International Command and Control Research and Technology Symposium: The Future of C2, USA, 2005
[2] Lobao, Evangelista and Faris, "Beginning XNA 2.0 Game Programming", Apress, USA, 2008
[3] Thibault D.U, "Commented APP-6A-Military symbols for land based systems", Canada, 2005
[4] Cadet Daniel Arnett. et al. "Tactical Terrain Visualization System" Simmons, Systems and Software Technology Conference, USA, 2004
[5] Koyuncu B, Bostancı E, "A Scenario Based Virtual Military Sandbox Implementation Using Web Services", Proceedings of International Conference on Advanced Computer Control, Singapore, 2009
[6] Kanbur, "3D Studio Max Visualization and Modelling", Pusula Publishing, Turkey, 2005
[7] MSDN XNA Documentation, http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.game.update.aspx (Last access: November 17th, 2009)
[8] Elmasri, Navathe, "Fundamentals of Database Systems", Addison-Wesley, 2004
[9] Durbin, J.; Swan, J.E. et al, "Battlefield visualization on the responsive workbench", Visualization apos;98. Proceedings, pp. 463-466, 1998
[10] Simon Julier et al. "The Software Architecture of a Real-Time Battlefield Visualization Virtual Environment", Proceedings of the IEEE Virtual Reality, ISBN:0-7695-0093-5, USA, 1999
[11] Deitel and Deitel, "C# for Programmers", Prentice Hall, USA, 2006
[12] Reginald L. Hobbs. "Using XML to Support Military Decision Making", XML Conference and Exposition Proceedings, USA, 2003
[13] Shiau Y, Liang S, "Real-time Network Virtual Military Simulation System", 11th International Conference Information Visualization, IEEE, Switzerland, 2007
[14] Robert A. Wisher et al. "The Virtual Sand Table: Intelligent Tutoring for Field Artillery Training", Research Report 1768, US Army Research Institute for Behavioral and Social Sciences, USA, 2001
[15] Gia Sirbiladze, Bezhan Ghavaberidze, Pridon Dvalishvili, Proceedings of the 11th WSEAS International Conference on Automatic Control, Modelling and Simulation, ISBN: 978-960-474-082-6, pp.297-302
[16] Jerzy Balicki, Multi-criterion Decision Making by Artificial Intelligence Techniques, Proceedings of the 8th WSEAS Int. Conf. on Artificial Intelligence , Knowledge Engineering and Databases (AIKED '09), ISBN: 978-960-474-051-2, pp.319-324
[17] Bara Adela,Diaconita Vlad, Lungu Ion, Velicanu Manole, Decision Support Systems – Improving Performance with Object Oriented Implementation, Proceedings of the 8th WSEAS Int. Conf. on Artificial Intelligence , Knowledge Engineering and Databases (AIKED '09), ISBN: 978-960-474-051-2, pp.331-336
[18] Meiqun Liu, Kun Gao, Knowledge Extracting Platform Based on Web Service, Proceedings of the 3rd WSEAS International Conference on Computer Engineering and Applications (CEA'09), ISBN: 978-960-474-41-3, pp.81-86
[19] Tolk A., "A Common Framework for Military M&S and C4I Systems", Spring Simulation Interoperability Workshop, 2003, Florida, USA