ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

ASKERİ SAVAŞ ALANLARININ GERÇEK ZAMANDA 3-BOYUTLU SANAL SİMÜLASYONU

Gazi Erkan BOSTANCI

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2009

ÖZET

Yüksek Lisans Tezi

ASKERİ SAVAŞ ALANLARININ GERÇEK ZAMANDA 3 BOYUTLU SANAL SİMÜLASYONU

Gazi Erkan BOSTANCI

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Baki KOYUNCU

Bu tez çalışması kapsamında askeri savaş alanlarının gerçek zamanlı 3 boyutlu sanal benzetiminin yapıldığı bir yazılım geliştirilmiştir. Geliştirilen Sandbox isimli yazılım, savaş alanına ait yükselti verisini DEM biçimindeki dosyalardan okuyarak, üzerinde askeri birimlerin ekleneceği platformu oluşturur. Farklı askeri birimlerin güncellenmesi işlemleri sağlanmıştır. Askeri birimler hem güncel bir askeri semboloji olan NATO-APP-6A ile hem de askeri birimlerin 3B modelleri halinde gösterilir. Yazılım, birimlerin konumlarında yapılan değişiklikleri öteleme animasyonu ile gösterir. Genel olarak XML verileri üzerinde çalışan yazılımda farklı platformlar arasında veri akışı esas alınmıştır. Rapor bilgilerinin uzun süreli saklanması için bir veritabanı kullanılmıştır. Web servisleri aracılığıyla savaş alanında bulunan uzak gözlemleyicilerin de yazılımın durumunu gerçek zamanda değiştirmeleri sağlanmıştır. Başlangıç koşulları verilen iki takım için savaşın nasıl gelişeceğini inceleyen Lanchester Savaş Modelleri de bu görsel yazılımla bütünleştirilerek, matematiksel modellerin de savaş alanı gösteriminde kullanılması ortaya konmuştur. Asıl tez konusuna ek olarak yapılan iki çalışmanın ilkinde arazi görselleştirme üzerine bir yöntem sunulmuş, ikincisinde ise askeri durumlarda karar vermede yardımcı olmak üzere bir karar destek aracı geliştirilmiştir.

Temmuz 2009, 103 sayfa

Anahtar Kelimeler: Sandbox, 3B Savaş Alanı Gösterimi, Komuta kontrol, Ağ-merkezli Savaş, Karar Verme, Matematiksel Modelleme, Veritabanı, Web Servisleri

ABSTRACT

Master Thesis

3D VIRTUAL SIMULATION OF MILITARY BATTLEFIELDS IN REAL TIME

Gazi Erkan BOSTANCI

Ankara University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. Baki KOYUNCU

Within the context of this thesis, a real time 3D virtual simulation software for visualization 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 updated in the software. Military units were viewed 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 receive reports from remote field observers and change the state of the software. Usage of mathematical models with battlefield visualization was presented by integrating the Lanchester Combat models, which examines how the battle will reveal for two teams given initial conditions. In addition to the main thesis topic, two additional studies were carried out. One of these studies aimed at developing a method for terrain visualization, and in the other study, a decision support tool was developed in order to be used in military situations.

July 2009, 103 pages

Key Words: Sandbox, 3D Battlefield Visualization, Command and control, Network-centric Warfare, Decision Making, Mathematical Modelling, Database, Web Services

TEŞEKKÜR

Çalışmalarımı yönlendiren, araştırmalarımın her aşamasında bilgi, öneri ve yardımlarını esirgemeyerek akademik ortamda olduğu kadar beşeri ilişkilerde de engin fikirleriyle yetişme ve gelişmeme katkıda bulunan danışman hocam sayın Prof. Dr. Baki Koyuncu'ya en derin duygularla teşekkür ederim

Çalışmalarım süresince maddi, manevi pek çok fedakârlık göstererek beni destekleyen ve her anımda yanımda olan aileme tüm içtenliğimle minnettarlığımı belirterek, en derin duygularla teşekkür ederim.

Gazi Erkan BOSTANCI
Ankara, Temmuz, 2009

İÇİNDEKİLER

KISALTMALAR DİZİNİ

2B	2 Boyutlu
3B	3 Boyutlu
AEC	Architectural, Engineering and Construction
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BBN	Bayesian Belief Network
C4I	Command, Control, Communications, Computer and Intelligence
DOM	Document Object Model
DTD	Document Type Definition
EOD	Explosive Ordnance Disposal
FOV	Field of View
HQ	Headquarters
HTTP	Hyper Text Transfer Protocol
MP	Military Police
NATO	North Atlantic Treaty Organization
NBC	Nuclear, Biological and Chemical
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
VRML	Virtual Reality Modelling Language
XML	Extensible Markup Language

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Ülke savunması ilk insan topluluklarının oluşmasıyla en basit düzeyde başlamış; modern devletlerin kurulmasıyla da bugünkü seviyesine, giderek daha da önem kazanarak ulaşmıştır.

Tarihteki savaşlar belirli bir bölgede yoğunlaşmışken, günümüzde geniş bir alana yayıldıkları gözlenmektedir. Özellikle de yeni teknolojilerin kullanılmasıyla neredeyse dünya üzerindeki her bölge savaş alanı olarak değerlendirilebilir. Bu durum askeri birimler ve karar organları arasında sağlanan iletişimin önemini daha da artırmıştır.

Özellikle bir kriz anında ülke savunmasında görevli komutanlar durumu en açık ve doğru biçimde gözlemleyerek, en kısa zamanda başarıya götürecek stratejiyi belirleyerek hareket etmelidir. Bunun için de muhabereden gelecek verilerin güncel ve güvenilir olması son derece önemlidir. Gelişen teknoloji bu bağlamda da yeni olanaklar sunmaktadır.

Askeri karar vermede izlenen iki geleneksel yöntem bulunmaktadır. Bu yöntemlerin ilki, kumandanların bir harita üzerinde belirlenmiş askeri sembolleri kullanarak birimlerin birbirlerine göre konumlarını muhabereden gelen bilgilere göre işaretleyerek savaşın nasıl gelişebileceği konusunda fikir üretmeleriydi. Bu yöntem için ilk önce bölgenin haritasının bastırılması gerekiyordu. Basılı harita üzerinde işaretlenen birimlerde herhangi bir hata veya güncelleme yapılması haritanın yeniden basılıp düzenlenmesi anlamına geliyordu. Bu haritanın son halinin bir de diğer kumandanlara gönderilerek bilgilerin paylaşılması sağlanmalıydı. Bu uygulama son derece zahmetli ve verimsiz bir yöntemdi.

İkinci yöntemde ise arazinin üç boyutlu modellenmesi bir masa üzerinde kum aracılığıyla yapılırdı. Arazinin engebesi kum ile gösterilmiş, üzerinde de maket modeller ile stratejiler belirlenirdi. Bu yöntemde de arazinin kum üzerinde modellenmesi son derece zor olmakla birlikte, elde edilen sonucun diğer bir yerdeki kumandanlara iletilmesi neredeyse imkânsızdır.

Yukarıda belirtilen yöntemlerin yerine günümüzün modern ihtiyaçları da göz önünde bulundurulduğunda savaş alanının daha etkili bir biçimde modellenmesi ancak günümüz teknolojilerinin kullanılmasıyla mümkün olacaktır.

Bu amaçla yapılan tez çalışmasında sanal bir kum sandığı uygulaması geliştirilmiştir. Çalışma ile kumandanların araziyi etkili bir biçimde görmelerine, arazi üzerindeki birimlerin verilerine güncel bir şekilde erişebilmelerine ve buna göre daha doğru kararlar vererek başarılı stratejiler ortaya koymalarına yardımcı olmak amaçlanmıştır.

Çalışmanın temeli 3 boyutlu sayısal bir coğrafi bilginin alınıp, üzerinde birimlerin gerçeğe uygun bir biçimde gerçek zamanlı konumlarıyla birlikte görselleştirilmesi ve elde olan bilgilerin uzakta bulunan diğer karar mekanizmalarına aktarımı esasına dayanmaktadır.

Askeri birimlerin güncel ve ortak bir gösterim biçimi ile gösterilmesi yapılan yorumların farklı komutanlarca anlaşılması açısından önemlidir. Bu nedenle yapılan çalışmada askeriyede kullanılan en yeni semboloji de eklenmiştir.

Geliştirilen yazılımda görselliğin ön planda olması birincil hedef olarak belirlenmiştir. Bu sebepten dolayı bilgisi okunan arazi verisinden elde edilecek modelin farklı yüzey koşulları belirtecek biçimde çimen, çöl, kar gibi dokularla kaplanabilmesi, modelin üzerindeki birimlerle birlikte farklı açılardan görüntülenebilmesi, görüntülere yaklaşılıp uzaklaşılabilmesi ve farklı tür kameralar açısı (dikey, eğik ve hareketli) ile görüntülenebilmesi sağlanmıştır.

Bunun yanında daha etkili bir gösterim için kullanılan sembolojinin yanı sıra bazı askeri birimlerin 3 boyutlu modellerinin de sisteme eklenmesi hem yazılımın kullanıcılar tarafından öğrenilmesini kolaylaştırmış, hem de belirli bir askeri durumun daha etkili bir biçimde ortaya konulmasını sağlamıştır.

Yazılıma zaman kavramı da bütünleşik bir biçimde eklenerek, verilen rapor bilgilerinin güncelliğini kontrol edebilmelerinin sağlanması amaçlanmıştır. Bu özelliğin yanı sıra

daha önceki bir zamana ait verilerin görüntülenmesi istendiğinde de kullanıcıya bunu sağlayabilecek bir senaryo mekanizması da eklenmesi planlanmıştır. Bu sayede karar verilirken önceki durumlar da göz önünde bulundurularak daha sağlıklı stratejilerin ortaya konması sağlanmıştır.

Daha etkili bir biçimde karar verebilmenin en önemli unsuru olan güncel ve doğru bilgi aktarımı için de uzak bilgisayarlarla bilgi alışverişini sağlayacak olan web servisi özelliği de kullanılarak yazılımın komuta kontrol mekanizması tek boyutlu olmaktan kurtarılmıştır.

Asıl çalışmanın yanı sıra yapılan iki çalışma ile de hem bir arazi modelinin oluşturulması için basit bir yöntem öne sürülerek araziyi görselleştiren bir yazılımın geliştirilmiş, hem de bir karar verme sürecinde kullanılabilecek matematik modeller üzerine kurulu etkileşimli bir yazılım ortaya konmuştur.

Daha önce geliştirilen sistemlerin incelenmesi, geliştirmek için kullanılan yöntemleri göstermiş ve bu sistemlerin eksik kısımlarının nasıl geliştirilebileceği konusunda fikirler vermiştir.

İncelenen çalışmalar sonucunda genel olarak kullanılan yöntemler aşağıdaki gibi belirtilebilir:

- Bir arazi verisinin okunması ve bu veriye göre arazi modelinin oluşturulması
- Arazi üzerindeki birimlerin; semboloji, 3B modeller veya her ikisi birlikte kullanılarak yerleşimlerinin gösterilmesi
- Nesne yönelimli bir yapı kullanılarak sistemin daha ileri zamanlardaki gereksinimlere göre de geliştirilebilir olarak tasarlanması
- Kullanım kolaylığı ve uyumluluğu nedeniyle dağıtık sistemler üzerinden XML ile veri aktarımı ve bu aktarılan veriye göre sistem bilgisinin güncellenmesi
- Geliştirilen sistemlerin mümkün olduğunca kolay bir biçimde öğrenilebilmesi ve kullanıcı dostu olması

Shiau (2007), gerçek zamanlı Internet üzerinden çalışan bir askeri benzetim sistemi geliştirmiştir. Sistemde arazi ve çevre koşulları modellenmiş, gerçekçilik için de çarpışma tespiti (collision detection) gibi yöntemler uygulanmıştır. Teoride arazi üzerindeki askeri birimlerden bahsedilmiş fakat uygulaması henüz yapılmamıştır. Daha çok arazi ve benzetim altyapısı üzerinde durulmuştur.

Thibault (2005), NATO APP-6-A sembolojisini tam olarak tek bir kaynakta derleyip, açıklamalı olarak göstermiştir.

Arnett (2004), tarafından ABD ordusunun savaş alanı gösterimi ihtiyacını karşılamak için geliştirilen çalışmada bir arazi verisi üzerinde askeri bir sunucudan gelen verilere göre birimlerin pozisyon bilgileri ve birimler gösterilmektedir. İlk olarak etkili bir biçimde gösterim üzerine yoğunlaşan çalışmada birimler basit şekiller kullanarak görüntülenmiştir.

Hutton (2003), Java3D ve VRML kullanarak oluşturduğu sanal ortamı XML verileri ile destekleyerek incelenen savaş alanının 3 boyutlu bir görünümünü elde etmiştir.

Hobbs (2003), harekât planlamada XML verisi kullanmayı öne sürmüştür ve geliştirdiği XML tabanlı SCML adlı dille senaryo tabanlı karar vermeyi açıklamıştır.

Bart (2003) Lanchester Savaş Modellerini incelemiş ve bir Excel dosyası üzerinde karşılıklı birimlerin zaman içinde sayılarının nasıl değişeceğini göstermiştir.

Wisher (2001), topçu eğitim sınıfları için akıllı bir ders sistemi geliştirmiştir. Uygulamalarda öğrencilerin belirli bazı noktalara çoklu roket sistemleri kurmaları beklenmiştir. Sanal bir kum sandığı ile uygulamalar sırasında öğrencilere gerekli geribildirimler yapılarak eğitimin etkili olması amaçlanmıştır.

Hix (1999), bir sanal gerçeklik uygulamasının kullanıcı güdümlü tasarımı ve değerlendirmesi üzerine çalışmıştır. Uzman bulgusal değerlendirme, kullanılabilirlik

değerlendirmesi ve son değerlendirme gibi yöntemlerin art arda yinelenmesi ile geliştirilen sistemin istenene en yakın olmasını amaçlamıştır.

Durbin (1998), Dragon isimli çalışmalarını duyarlı bir sanal tezgâh üzerinde iki farklı büyük askeri durum için uygulamış ve bir askeri savaş alanı gösteriminde gerekli olan özellikleri belirtmiştir.

Julier (1997), geliştirilen sistem ile birçok farklı kaynaktan veri alan ve kullanan bir mimari uygulamıştır ve bu mimariyi açıklamıştır. Veriler tutarlı bir yapı içerisinde 3B gösterimde kullanıcıya sunulmaktadır. Bu tip sistemlerin felaket yardımı ve rehine kurtarma gibi işlerde kullanılabileceği öne sürülmüştür.

Ming (1996), dinamik bir karar süreci gerektiren askeri durumlar için çelişki analizini (conflict analysis) kullanmıştır. Gelen verilerin göreceli güvenilirliklerinin de hesaba katıldığı bu çalışma ile belirsizliğin hâkim olduğu bir savaş alanında doğru karar vermek için yöntemler öne sürülmüştür.

Kirby (1995), network ortamı olan bir 3boyutlu sanal ortam oluşturmuştur ve bu ortamda kontrol değerlerinin işletilmesini sağlamıştır. Sanal bir arazi üzerinde 3B modeller kullanmıştır.

Taylor (1993), hem deterministik hem de stokastik yöntemler kullanarak askeri analiz, savaş oyunları, savaş modellemesi veya dinamik modelleme gibi işlemlerde Lanchester Savaş Modellerini incelemiştir.

2. KURAMSAL TEMELLER

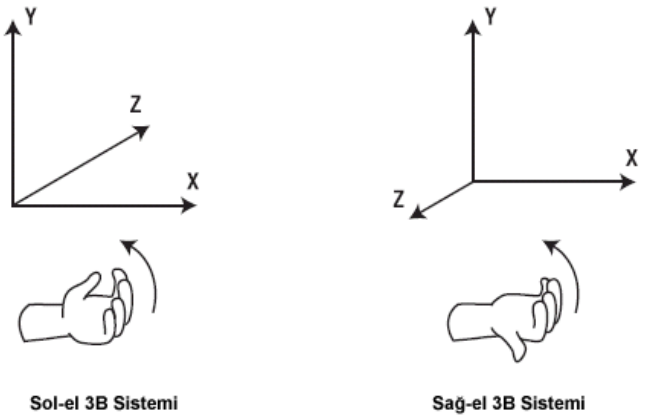
Tez kapsamında yapılan çalışmalarda kullanılan yöntemlerin arkasında yatan teorik bilgi bu kısımda anlatılacaktır. Bilgisayar grafiğinin temelleri kısaca anlatılmıştır. Bunun yanı sıra karar vermede kullanılabilecek matematik modeller, XML, veritabanları ve web servisleri gibi konular da ele alınmıştır. Bu kısım ilerideki konuların daha rahat anlaşılması için tezle ilişkilendirilmiştir.

2.1 Bilgisayar Grafiği

Bilgisayar kullanılarak gerçek veya sanal nesnelerin görüntülerinin oluşturulmasına bilgisayar grafiği denir. Temel olarak, ekranda 2 boyutlu (2B) resimler üretmek için kullanılan algoritmalar ve yöntemler olarak da açıklanabilir. Genellikle bu yöntemler, nesneleri doğrular, daireler, gölgeli çokgenler (poligon) ve metin olarak çizmemizi sağlar. Aşağıdaki alt bölümler bilgisayar grafiğinde kullanılan terminolojiyi açıklayacaktır.

2.1.1 Koordinat sistemleri

Nesnelerin 3 boyutlu (3B) uzaydaki konumlarını belirlemek için Kartezyen koordinat sistemi kullanılır. Bu sistemin bilgisayar grafiğinde genel olarak kullanılan iki türü vardır: sol-el ve sağ-el. Bu isimler z koordinat ekseninin x ve y eksenlerine göre konumuna göre verilir. Bir elimizin parmaklarını pozitif x eksenine doğru yöneltip, parmaklarımızı saat yönünün tersinde hareket ettirdiğimizde başparmağımız z eksenini gösterir. Bu durum Şekil 2.1'de gösterilmiştir.



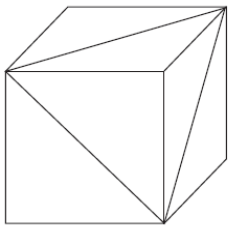
Şekil 2.1 Kartezyen koordinat sistemleri

Koordinatlar tez kapsamında geliştirilen yazılımlarda büyük bir önem taşımaktadır. Askeri birimlerin konumları bu koordinatlar aracılığıyla belirlenecek ve kullanıcılara sunulacaktır. Birimlerin birbirlerine göre konumları bu koordinat sistemi aracılığıyla görüntülenecektir.

2.1.2 Noktalar ve temel yapılar

Canlı bir organizmanın yapı ve görev bakımından en küçük yapı birliğinin hücre olması gibi her 3B nesnenin yapıtaşı noktalardır. Matematiksel olarak noktalar bulundukları koordinatlar ile tanımlanır. Bilgisayar grafiğinde ise bu koordinatların yanında noktanın renk bilgisi, doku özellikleri ve normal vektörü gibi bilgileri de tutulur.

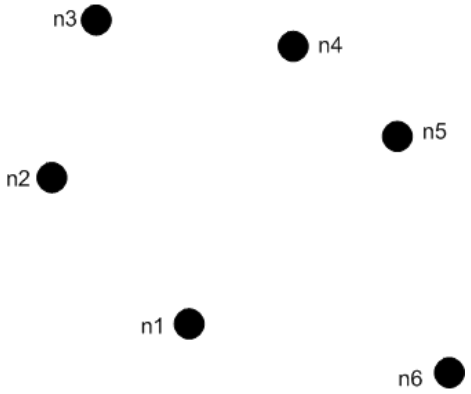
Nesnelerin çiziminde koordinat bilgisinin yanı sıra bu noktaların nasıl bağlanacağı da belirtilmelidir. Çizim yapılırken noktalar yalnızca yol gösterici olarak kullanılır. Ekranda gösterilen nesneler aslında çok sayıda üçgenden oluşmaktadır. Bu sayede çizilen nesnenin ne olduğu anlaşılabilir. Örneğin bir küp çizmek için Şekil 2.2'de gösterildiği gibi üçgenler kullanılır.



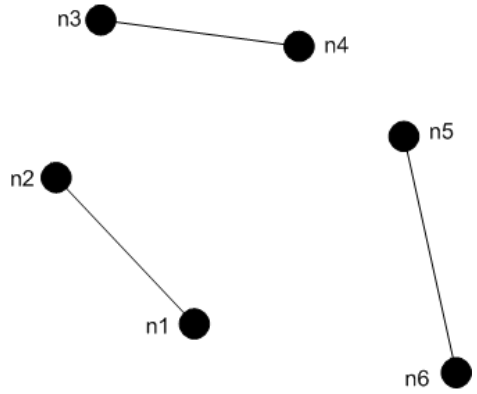
Şekil 2.2 Üçgenlerden oluşan bir küp

Elimizde bulunan noktaları üçgenler çizmek için kullanabildiğimiz gibi diğer farklı şekiller için de kullanabiliriz. Aşağıda aynı nokta kümesinin farklı biçimlerde kullanımları gösterilmektedir. Şekil 2.3'deki kullanım basit olarak yalnızca noktaları çizerken, Şekil 2.4'deki kullanım her iki nokta arasında bir doğru çizer. Şekil 2.5'te tüm noktaları kapsayan bir doğru oluşturulur. Şekil 2.6'daki kullanımda ardışık 3 nokta alınarak üçgenler oluşturulur. Karmaşık nesnelerin çiziminde en etkili yöntem olan Şekil 2.7'deki kullanımda ise eklenen her nokta ile en son eklenen 2 nokta kullanılarak yeni bir üçgen çizilir. Bu sayede farklı üçgenler için aynı noktaların tekrar edilmesine gerek kalmaz. Şekil 2.8'de ise noktaların bir yelpaze şeklinde birleştirilerek üçgenler oluşturması gösterilmiştir.

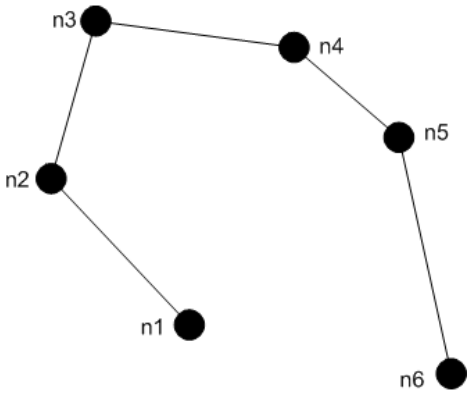
Çizelge 2.1 Aynı nokta kümesinin farklı yöntemlerle birleştirilmesi



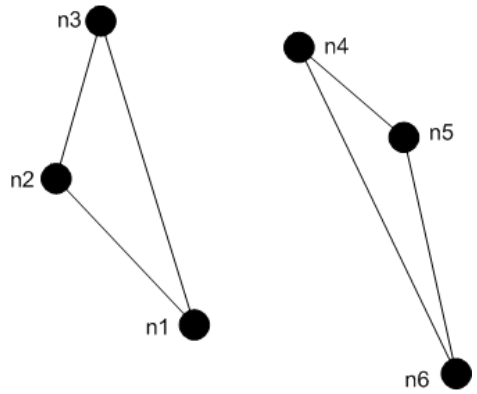
Şekil 2.3 Noktaların ayrı olarak çizimi



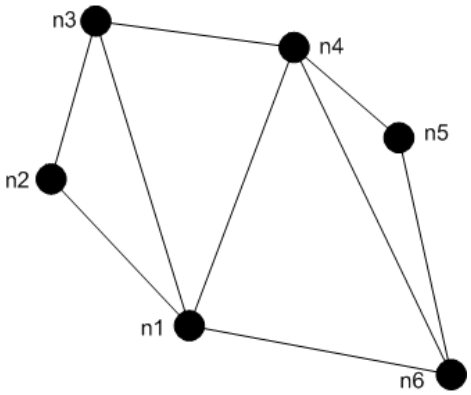
Şekil 2.4 Noktaların doğru olacak biçimde birleştirilmesi



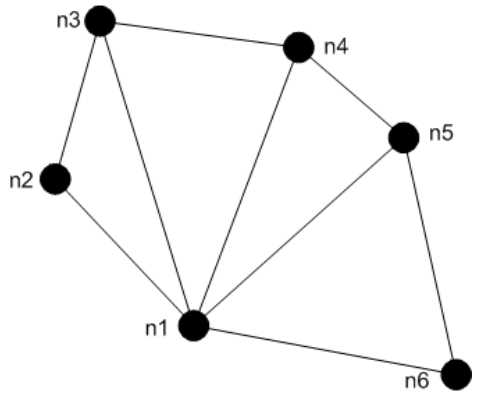
Şekil 2.5 Noktaların bir doğru bandı olarak birleştirilmesi



Şekil 2.6 Noktaların üçgenler olacak biçimde birleştirilmesi



Şekil 2.7 Noktaların bir üçgen bandı olacak şekilde birleştirilmesi



Şekil 2.8 Noktaların bir yelpaze olacak şekilde birleştirilmesi

2.1.3 Vektörler, matrisler ve 3B dönüşümler

Noktaların konum bilgisi vektörlerde tutulur. Her nokta için o noktanın x, y ve z koordinatları:

$N = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ olarak ifade edilir. Bu vektör, tüm noktalar için ayrı olarak tutulur ve herhangi bir dönüşüm yapıldığında bu dönüşüm tüm vektörlere uygulanır. Tüm noktalarına dönüşüm uygulanan nesnenin kendisi de bu şekilde dönüşümden etkilenmiş olur.

Genel olarak kullanılan 3 tip dönüşüm vardır:

a. Öteleme: 3B uzaydaki bir konuma, $N = (x, y, z)$, öteleme uzaklıkları t_x, t_y, t_z eklenerek yeni konum, $N' = (x', y', z')$, elde edilmiş olur: $x' = x + t_x$, $y' = y + t_y$, $z' = z + t_z$. Bu dönüşüm aşağıdaki matris çarpımı olarak ifade edilebilir:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ veya } N' = T.N$$

b. Dönme: Bir nesnenin belirli bir eksen etrafında θ açısıyla yaptığı harekettir. Elimizdeki N noktasını z ekseni etrafında çevirmek istersek:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ veya } N' = R_z(\theta).N$$

Diğer eksenler etrafında dönme de bu formülün koordinat parametrelerinin dairesel permütasyonları olarak hesaplanır. $x \to y \to z \to x$ şeklinde x'i y ile, y'yi z ile ve z'yi x ile değiştirerek x ekseni ve y ekseni etrafındaki dönme formülleri de bulunur (Hearn, Baker, 2003). Buna göre y ekseni etrafında dönme:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$ veya $N' = R_y(\theta).N$ olarak tanımlanır.

c.Ölçekleme: Bir nesnenin boyutlarının belirlenmesidir. Bir koordinat değeri belirli bir katsayı ile çarpıldığında ölçeklenmiş olur. Bu katsayı 1'den büyükse cisim daha büyük; 1'den küçükse (0'dan büyük olacak şekilde) cisim daha küçük olarak çizilir. Bu dönüşüm

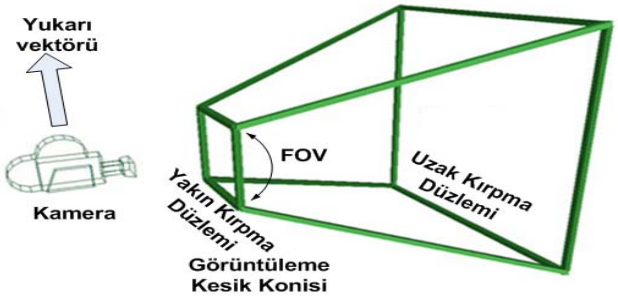
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$ veya kısaca $N' = S.N$ olarak gösterilir.

Bahsedilen dönüşümler yapılan tez çalışmasında birçok yerde kullanılmıştır. Örneğin, askeri birimlerin yer değiştirmeleri ötelemeye birer örnektir. Bir konumdan diğer bir konuma hareket edilirken yönün belirlenmesi ve yönelim dönme işlemine örnektir. Askeri modeller kullanılmadan önce, modellerin birbirlerine göre büyüklük oranlarının ayarlanması da ölçeklemeye örnek olarak gösterilebilir.

2.1.4 Kameralar

Bir doğrultuda baktığımızda sadece gözlerimizin görme alanı içine giren nesneleri görürüz ve bu alanın dışındaki nesneleri var oldukları halde göremeyiz. Bilgisayar grafiğinde insan gözünün yerine sanal kameralar vardır. Bu kamera görebildiği nesne görüntülerinin işlenmesini sağlar ve ekranda gösterir.

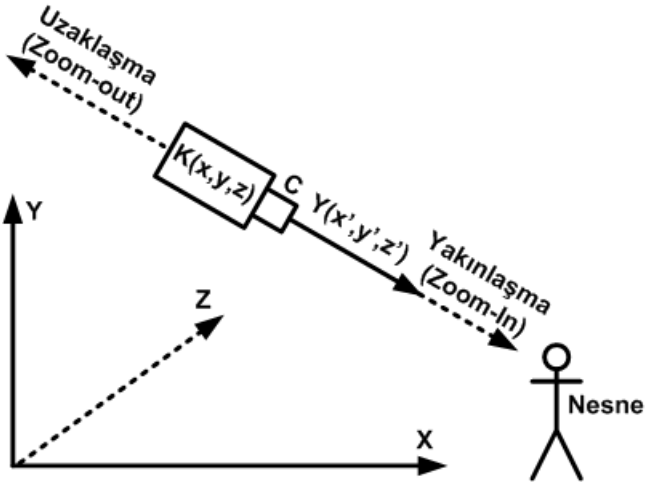
Benzer şekilde grafik uygulamalarında da yalnızca görülebilen alan işlenerek ekrana gösterir. Bu yolla görüntüde hiçbir değişiklik olmadığı halde hem işlemci gücünden hem de bellekten tasarruf edilir.



Şekil 2.9 Kameranın görüntülediği hacim

Kameranın gördüğü alanı tanımlayan bölge bir kesik koni ile belirtilir (Şekil 2.9). Burada yakın ve uzak kırpma düzlemleri arasında kalan bölüm ekranda görüntülenecektir. Uzak kırpma düzleminin arkasındaki alan çok uzakta olan cisimleri görüntüleyemeyeceğimiz için işlenmez. Benzer şekilde yakın kırpma düzleminin gerisinde kalan cisimler de kameranın arkasında kalacağından görüntülenmez. Yukarı vektörü kameranın duruşunu gösterir. Görüş alanı (FOV) ise kameranın kaç derecelik bir açıyla görüntü aldığını gösterir.

Bahsedilen parametrelere ek olarak kameranın konumu ve baktığı yön vektörleri sayesinde görüntülenecek alan tanımlanır. Bazı uygulamalarda sabit bir kamera konumu yeterli olurken, kimi zaman da kameranın konumunun değiştirilerek çizilen 3B sahnenin farklı noktalarının görüntülenmesi gerekir. Bazen de kameranın baktığı yön sabit kalarak konumun nesnelere yaklaşıp uzaklaşması (zoom in/out) söz konusudur. Bakış noktası sabit kalacak şekilde yakınlaşıp uzaklaşma aşağıdaki biçimde yapılır.



Şekil 2.10 Kameranın yaklaşıp uzaklaşması

Şekil 2.10'da K kameranın konumunu, Y ise kameranın baktığı yönü göstermektedir. Buna göre kamerayı aynı doğrultu üzerinde nesneye yakınlaştırmak veya nesneden uzaklaştırmak istendiğinde yapılacak işlem Y ve K noktalarından geçen doğru denkleminin ($y = mx + n$) bulunmasıdır. Bu doğru üzerinde nesneye yakın bir x koordinatı belirlenir ve denklemde yerine konulduğunda buna karşılık gelen y koordinatı bulunur. Bu sayede kameranın yeni konumu belirlenmiş olur.

Tez kapsamında geliştirilen yazılımda farklı kamera tipleri kullanılarak savaş alanının farklı açılardan görüntülenmesi sağlanmıştır.

2.1.5 Animasyon

Daha gerçekçi uygulamalar için animasyonlar kaçınılmazdır. Bir nesnenin canlandırılması, o nesneye dönme, hareket etme ve hatta büyüyüp küçülme gibi yeteneklerin verilmesidir. Bu tür hareketler de dönüşüm olarak adlandırılır ve dönüşümler aşağıdaki biçimleri alır:

a. **Birim dönüşüm:** Dönüşüm için hiçbir değerin tanımlanmadığı başlangıç değeridir.
b. **Ölçekleme:** Nesnenin boyutundaki değişiklikleri belirler.
c.Öteleme: Nesnenin; X, Y ve Z düzlemlerindeki aşağı, yukarı, sağ, sol hareketleridir.
d. **Dolanım:** Nesnenin kendi etrafında yaptığı dönme hareketidir
e.Yörünge hareketi: Nesnenin kendisi dışında sabit bir nokta etrafında yaptığı dönme hareketidir.

Yukarıda verilen hareketler zamanla ilişkilendirildiğinde belirli bir uyum içerisinde bir animasyon oluştururlar. Örneğin, bir askeri birim için yeni bir konum girildiğinde eski konum ile yeni konum arasında doğrusal ara-değerleme yapılarak birimin yavaş bir biçimde yeni konuma gitmesi sağlanır. Matematiksel olarak anlatmak istersek doğrusal ara-değerleme, $öncekiKonum + (yeniKonum - öncekiKonum) * artış$ formülü ile ifade edilebilir. Bu formül, ekranın her yenilenmesinde kullanılarak, askeri birim için yeni konum belirlenir. Böylece birim direk olarak yeni konumda görünmek yerine yavaşça belirtilen konuma ilerler.

2.2 Karar Vermede Kullanılan Matematiksel Modeller

Bu kısımda uygulanan üç matematiksel model detaylı bir biçimde incelenecektir. Bu modeller sırasıyla Karar Matrisi, Lanchester Savaş Modeli ve Basit Silahlanma Yarışı Modelidir.

2.2.1 Karar matrisi

Askeri uygulamalarda karar verme (Ming, 1996) son derece önemlidir. Kumandanların birçok parametresi olan anlık bir durumu dikkatle incelemesi gerekir. Askeri bir karar basit olarak hedef, şu anki durum ve kısıtlayan etkenler (risk)'den oluşan bir vektör ile ifade edilebilir.

Bu çoklu özellikler her zaman aynı derecede önem taşımayabilir. Örneğin, bazı durumlarda lojistik destek veya destek kuvvetlerin önemi, karşı kuvvetlerin oranından daha fazla olabilir.

Karar matrisi, bir karar verme sürecinde:

a. Sorunu tam olarak ortaya koymak ve gerekenleri bir liste biçiminde belirlemek

b. Etkenleri, sorundaki göreceli önemlerine göre değerlendirmek

c.Birçok seçenek arasından en iyi olanı seçmek amacıyla kullanılır.

Bahsedilen araç tam bir karar destek sisteminin sadece küçük bir kısmını oluşturabilir fakat bir durumu etkileyen etkenlerin ve incelenecek olası seçeneklerin açığa çıkarılmasında son derece önemlidir.

Karar matrisini oluşturmak için 4 adımdan oluşan bir işlem yapılır. İlk olarak, kararı etkileyen etkenler (Criteria) belirlenir. İkinci olarak olası seçenekler (Options) listelenir. Üçüncü adımda her etkenin göreceli önemi (Weights) ortaya konur. Son olarak da seçenekler duruma uygunluklarına göre değerlendirilir (Scores). Bu yöntem, adımların baş harfleri alınarak COWS olarak da adlandırılır.

Örnek bir askeri karar verme sürecinde 4 etken kuvvet oranı, arazi, lojistik destek ve risk olarak belirlensin. Tüm bu bilgilerin, arazide bulunan uzak gözlemleyiciler tarafından alınacağı varsayılsın. Farklı seçenekler her madde belirli bir puan aralığından alınan notlar ile değer alacaktır. Değerlendirme yöntemi aşağıda verilmiştir:

G={1,2,3,4,5,6,7}= {en yetersiz, çok yetersiz, yetersiz, önemsiz, yeterli, çok yeterli, en yeterli}

Örneğin, arazi düzse arazi üzerinde hareket olanağı yeterlidir ve gözlemleyicinin algısına ve deneyimine göre 5, 6 veya 7 ile notlandırılabilir. İlk üç etken için mantık basittir. Risk etkeni için de ters çalışır; yüksek risk daha büyük bir sayı ile ifade edilecektir fakat mantık bunu önleyici bir etken olarak yorumlayacaktır.

Karar matrisini oluştururken önemli bir konu ağırlıkların atanmasıdır. Her seçenek için etkenler bu ağırlıklara göre notlandırılacak ve sonuç bu notlara göre değerlendirilecektir. Puanlar basitçe aşağıdaki gibi hesaplanır:

$$puan = ağırlık \times değerlendirme \quad (2.1)$$

Tüm sistem için puanlar aşağıda verilen matris çarpımı ile hesaplanır. Formülde, w_i bir etkenin ağırlığını; $o_i c_j$, o_i seçeneği için c_j etkenine göre değerlendirmeyi ve s_i, o_i seçeneği için son puanı verir.

$$\begin{bmatrix} s1 & s2 & s3 \end{bmatrix} = \begin{bmatrix} w1 & w2 & w3 & w4 \end{bmatrix} \times \begin{bmatrix} o1c1 & o2c1 & o3c1 \\ o1c2 & o2c2 & o3c2 \\ o1c3 & o2c3 & o3c3 \\ o1c4 & o2c4 & o3c4 \end{bmatrix} \quad (2.2)$$

Elbette tüm seçenekler için hesaplamalar yapıldıktan sonra verilecek son karar kumandanın sorumluğundadır. Ancak bu araç, tüm olası seçeneklerin dikkatli bir biçimde incelenmesinden sonra üzerinde kararın sonuçlandırılacağı noktaları listelemekle görevini yerine getirmiş olur.

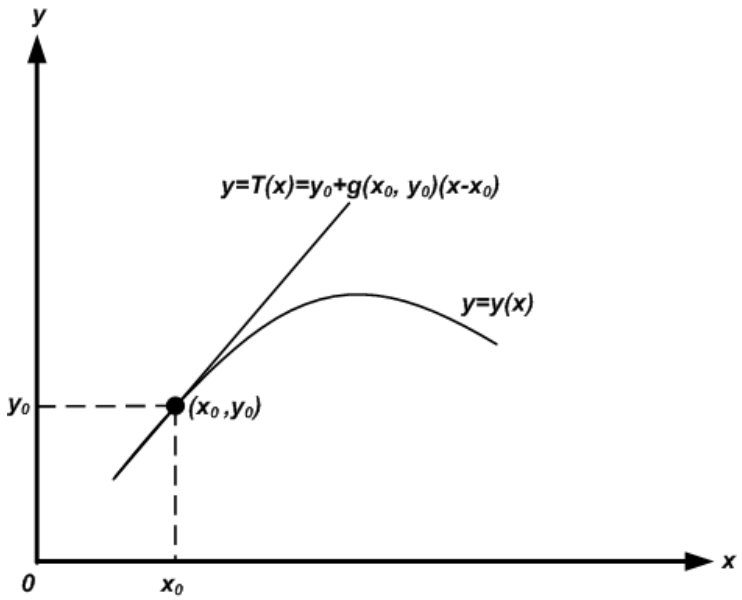
2.2.2 Euler yöntemi

$dy/dx = g(x,y), y(x_0) = y_0$ biçiminde verilen bir başlangıç değer problemi için tam bir çözüm fonksiyonu gerekmiyorsa, problem bilgisayar yardımıyla bir yaklaşımlar tablosu oluşturularak çözülebilir. Bu tür yöntemlere nümerik yöntem denir ve Euler yöntemi de bu sınıfa girer.

$dy/dx = g(x,y)$ olarak verilen bir denklemde başlangıç değeri $y(x_0) = y_0$ ise çözüme denklemin teğet doğrusu kullanılarak yaklaşım yapılabilir.

$$T(x) = y_0 + g(x_0, y_0)(x - x_0)$$

$g(x_0, y_0)$ değeri verilen eğrinin ve teğet çizgisinin (x_0, y_0) noktasındaki eğimidir. Şekil 2.11'de de gösterildiği üzere $T(x)$ fonksiyonu x_0 yakınlarında $y(x)$ için iyi bir yaklaşım oluşturur. Euler yönteminin temeli bu teğet çizgi yaklaşımlarından oluşan bir dizi kullanmaktır.

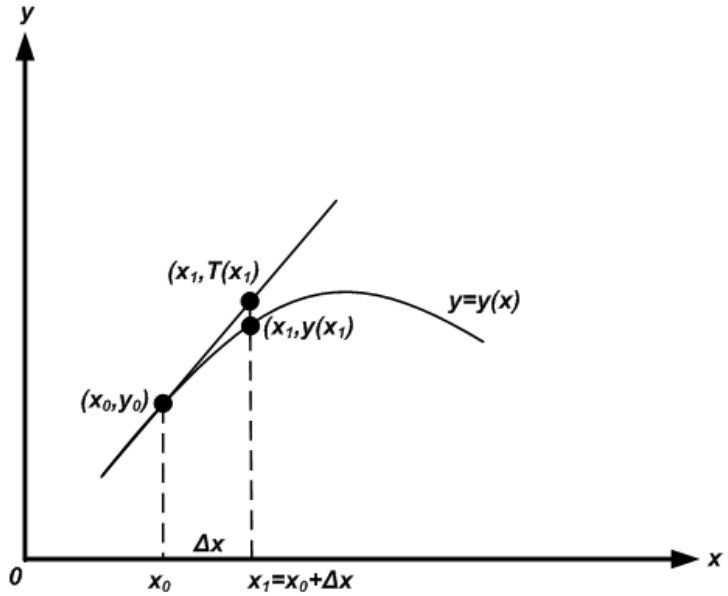


Şekil 2.11 T(x) fonksiyonunun gösterimi

(x_0, y_0) noktasının çözüm eğrisi üzerinde olduğu kesin olarak bilinmektedir. Bağımsız değişken için yeni bir değer belirlenirse $x1 = x0 + \Delta x$, ve Δx de oldukça küçük olursa

$$y_1 = T(x_1) = y_0 + g(x_0, y_0)\Delta x,$$

gerçek çözüm $y = y(x_1)$ için oldukça iyi bir yaklaşım olacaktır. Böylece asıl eğrimizin üzerinde bulunan bir noktadan yeni bir (x_1, y_1) noktası bulmuş olunur ki bu da çözüm eğrisi üzerindeki $(x_1, y(x_1))$ noktasına çok yakındır. Bu durum Şekil 2.12'de gösterilmiştir.

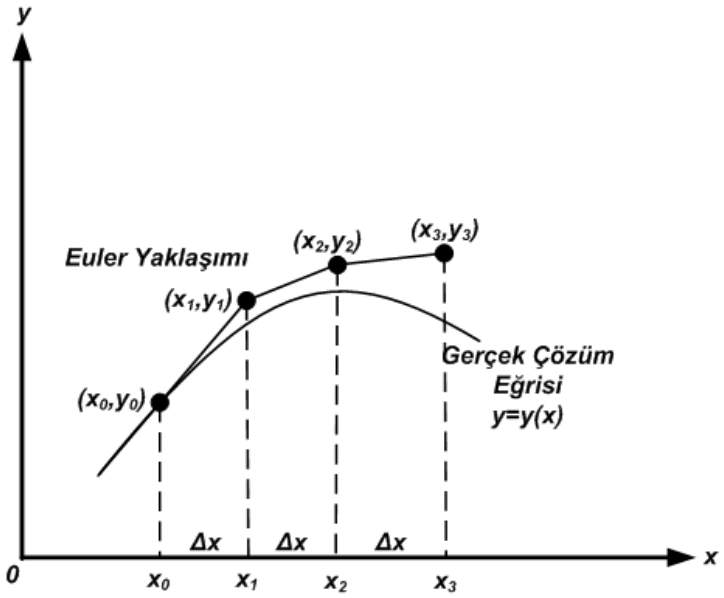


Şekil 2.12 Yaklaşımın ilk adımının hesaplanması

Elde edilen (x_1, y_1) noktası ve $g(x_1, y_1)$ kullanılarak bir işlem daha yapılırsa:

$$y_2 = y_1 + g(x_1, y_1)\Delta x$$

(x_2, y_2) noktası da elde edilmiş olur. Benzer adımlarla üçüncü ve diğer adımlar için de yaklaşımlar bulunmuş olur (Şekil 2.13).



Şekil 2.13 Euler yaklaşımının ilk 3 adımının gösterilmesi

Tez çalışmasında Euler yöntemi bir sonraki kısımda kullanılan diferansiyel denklem sisteminin çözümünde kullanılmıştır.

2.2.3 Lanchester savaş modelleri

Bu model (Giordano, 1997), (Taylor, 1993) Birinci Dünya Savaşı sırasında savaştaki durumları inceleyen F.W. Lanchester tarafından ortaya konmuştur ve modellere onun adı verilmiştir. Problem aşağıdaki gibi belirlenebilir:

Elimizde bir savaşta karşı karşıya gelen iki kuvvet bulunmaktadır; Kırmızı Takım ve Mavi Takım. Her iki takım da homojen kuvvetlerden oluşmaktadır. Sonucu etkileyen birçok çeşitli etken bulunmaktadır. Bunlar düşmanın ateş gücüne bağlı olarak harekât kayıpları, elektriksel veya mekanik problemlerden kaynaklı harekât dışı kayıplar ve destek ekiplerdir.

Bu model savaşın ne kadar süreceğini ve verilen parametreler ışığında nasıl gelişeceğini ortaya koyar. Kırmızı ve Mavi takımlardaki asker sayısı değişimini sırasıyla $\frac{dK}{dt}$ ve $\frac{dM}{dt}$ olarak gösterildiğinde elimizdeki ilk parametre için (harekât kayıpları), sistem aşağıdaki gibi modellenebilir:

$$\frac{dK}{dt} = -aM(t) \qquad (2.3)$$

$$\frac{dM}{dt} = -a'K(t) \quad (2.4)$$

Bu basit değişim fonksiyonu her takımın düşman üzerindeki zarar verici etkisini gösterir. Harekât dışı kayıplar da verilen bir zamanda takımın kendi birim sayısına bağlı olarak değişen bir fonksiyon olarak (2.3) ve (2.4)'e sırasıyla $-bK(t)$ ve $-b'M(t)$ olarak eklenebilir. Benzer şekilde destek kuvvetler de her iki takımın diğeri üzerindeki baskınlığını artıracak şekilde $K'(t)$ ve $M'(t)$ olarak eklenebilir. Bu işlemler sonucunda iki sistem için de aşağıdaki ifadeler elde edilir:

$$\frac{dK}{dt} = -aM(t) - bK(t) + K'(t) \qquad (2.5)$$

$$\frac{dM}{dt} = -a'K(t) - b'M(t) + M'(t) \qquad (2.6)$$

Bu diferansiyel denklemler sistemi elbette ilk değerlerden etkilenecektir. Bu ilk değerler, $K(0) = K_O$ ve $M(0) = M_O$, başlangıçta iki takımda bulunan birim sayısı olarak yorumlanmalıdır.

Verilen model Euler yöntemi ile kolay bir biçimde çözülebilir. Burada elimizde *t* bağımsız zaman değişkeni bulunmaktadır ve bu sayede savaşın geçtiği zaman aralığını birim zamanlara (2.7) ve (2.8)'de olduğu gibi bölünebilir:

$$\begin{aligned} t_1 &= t_o + \Delta t \\ t_2 &= t_1 + \Delta t \\ &\vdots \\ t_n &= t_{n-1} + \Delta t \end{aligned} \qquad (2.7)$$

$$\Delta t = 1 \quad (2.8)$$

Zaman aralığını (2.8)'de olduğu gibi birim olarak ayarlamak, sistemin ayrık gün veya haftalar için incelenmesine olanak sağlar. Sıradaki adım iki taraf için de genel bir formül oluşturmaktır.

Savaş sırasında herhangi bir zaman için, her takımdaki kuvvetlerin sayısı bir önceki zamanda bulunan kuvvetlerin sayısına bağlı olacaktır. Bu nedenle aşağıdaki ifadeler yazılabilir:

$$K_n = K_{n-1} + \Delta K \quad (2.9)$$

$$M_n = M_{n-1} + \Delta M \quad (2.10)$$

Burada ΔK and ΔM sırasıyla (2.5) ve (2.6)'daki formüllerle hesaplanacaktır ve sistemin sonucu aşağıdaki gibi olacaktır:

$$K(n) = K(n-1) - aM(n-1) - bK(n-1) + K'(t-1) \qquad (2.11)$$

$$M(n) = M(n-1) - a'K(n-1) - b'M(n-1) + M'(t-1) \quad (2.12)$$

(2.11) ve (2.12)'de t değişkeni de n gibi hesaplanmıştır; fakat burada değişkenin farklı gösterilmesinin nedeni verilen bir zamanda gönderilen destek kuvvetlerin sayısının takımdaki kuvvet sayısından bağımsız olmasıdır.

Hesaplanan nümerik yaklaşım (Stewart, 2007) 'ın da gösterdiği gibi basit bir biçimde kodlanabilir.

Bu model farklı etkenlerin bir savaşın gidişatını nasıl değiştireceğini göstermek açısından oldukça güçlü ve yararlıdır. Daha önce anlatılan karar matrisi ile birlikte kullanılarak bir savaşın nasıl gelişeceğini görmede kullanılabilir.

2.2.4 Basit silahlanma yarışı modeli

Bu model Lewis Fry Richardson tarafından geliştirilmiştir ve dolayısıyla Richardson Silahlanma Modeli (Williams, 2008) olarak da bilinir. Model kısaca aşağıdaki gibi açıklanabilir:

Modelde iki ülke (X ve Y) bir silahlanma yarışına girişirler. İki tarafın da amacı karşı tarafı yıldırmaktır, bunun yanında iki tarafta da ortak bir korku da hâkimdir. Bu durum her iki tarafın da karşı tarafın eklediği her bir birim silah başına, kendi elinde bulundurduğu silahlara belirli bir oranda silah eklediği bir duruma yol açar.

Matematiksel olarak ifade etmek istersek; X ülkesi, Y ülkesinin hiç silahı olmadığı durumda (a) sayıda silaha sahip olması gerektiğini düşünür ve yeni silah alırken de Y 'deki silah sayısının belirli bir (p) oranı kadar silah alacaktır. Bu durumda Y 'nin de aynı stratejiyi izleyeceği düşünüldüğünde her iki ülke için aşağıdaki formüller çıkarılabilir:

$$X = a + pY \quad (2.13) \qquad Y = b + p'X \quad (2.14)$$

Verilen formüllerde (p) ve (p') bir üstte belirtilen oranları gösterir. Bu parametreler ülkelerin sahip olduğu teknolojilere göre değişir. Örneğin; X ülkesinin Y ülkesi her bir silah aldığında 3 silah alması, Y ülkesinin daha yüksek bir teknolojiye sahip olduğunu gösterir.

Sorun bu silahlanma yarışının sabit bir sona ulaşıp ulaşmayacağıdır. Sabit bir sona ulaşılamayan durumlar genelde savaşa yol açar ve halk genel olarak silahlanmaya yapılan harcamaların artmasına karşı bir tutum sergiler. Bu nedenle sabit bir biçimde sonlanmayan durumlar istenmez.

Bu modeli çözmek için, basit olarak dinamik bir sistemin sayısal çözümü izlenecek olursa:

$$X_n = a + pY_{n-1} \quad (2.15) \qquad Y_n = b + p'X_{n-1} \quad (2.16)$$

$$X_0 = 0 \quad (2.17) \quad Y_0 = 0 \quad (2.18)$$

Model sistemin uzun sürede davranışını verecektir. İlk sistem Lanchester Savaş Modeli'nde olduğu gibi diferansiyel denklemler sistemi kullanılarak da çözülebilirdi fakat burada basit olmasından dolayı bu yöntem seçilmiştir.

2.3 XML

XML (Genişletilebilir İşaretleme Dili, Extensible Markup Language) kullanıcıların kendilerine ait verileri kendi tanımladıkları biçimde saklayabilmelerine olanak sağlayan bir dildir. Ayrıca farklı donanımlar veya işletim sistemleri üzerinde çalışan uygulamaların ortak olarak veri işleyebilmeleri için getirilen bir standarttır.

XML'in geliştirilebilmesinin ardında birkaç ana amaç vardır. Bunlardan ilki kullanıcıların ihtiyacı olan her tür veriyi etkili bir biçimde temsil edebilecek olmasıdır. Bu aynı verinin kendi kendini tanımlayabilecek bir yapıda olmasını sağlar. Bir diğer amaç ise tanımlanan veri biçiminin Internet üzerinde hem taşınabilip hem de kullanılabilmesidir. Bu amacın arkasındaki neden ise birçok farklı platformun aynı veri üzerinde işlem yapabilmesi ihtiyacıdır.

Genel olarak XML'in yapısını şu şekilde özetleyebiliriz: Bir XML belgesini bir ağaç olarak düşünürsek, en üst eleman (en dıştaki etiket) ağacın köküdür ve belgenin ne tür bilgi sakladığını özetler. Kök elemanın altındaki elemanlar o köke bağlı olan ve altında kendi alt elemanları da olabilen dallar olarak düşünülebilir. Aşağıdaki XML verisi bu anlatımı özetlemektedir:

```
<Ogrenciler>
     <Ogrenci>
      <ogrenciNo>03292106</ogrenciNo>
      <ogrenciAdı>G. Erkan Bostancı</ogrenciAdı>
     </Ogrenci>
     .
     .
     .
     <Ogrenci>
          <ogrenciNo>03292110</ogrenciNo>
      <ogrenciAdı>G. Burcu Bostancı</ogrenciAdı>
```

```
    </Ogrenci>
</Ogrenciler>
```

Dosya öğrenciler hakkında bilgi tutmaktadır. En dış elemanın altında birden çok öğrenci bulunabilir ve her öğrencinin kendine ait alt elemanları vardır.

Yapılan tez çalışmasında XML, örnek bir savaş alanına ait bilgilerin alınması ve daha sonra bu veriyi bir karar amaçlı kullanmanın yanı sıra elimizdeki savaş alanı verisinin senaryolar biçiminde kaydedilmesi gibi kısımlarda kullanılmıştır

.

2.4 Veritabanları

Bir veritabanı ilişkili verinin organize bir biçimde toplanmasıdır. Verinin organize bir biçimde tutulması, veri üzerinde bir arama yapıldığında kolayca bulunması için yol gösterir. Veritabanındaki bir verinin diğerleriyle ilişkisinin olması gerekir. Birbiriyle hiç ilişkisi olmayan verilerin bir arada tutulması ancak işi zorlaştıracaktır.

Veritabanları günlük hayatımızda da birçok yerde karşımıza çıkar. Her gün kullandığımız telefonumuzdaki kişiler listesi bir veritabanıdır. Kişilere ait telefon, e-posta hatta adres gibi bilgiler bu veritabanında saklanır.

Eskiden veritabanları tek bir dosyadan oluşmaktaydı. Bir programcı ihtiyacı olan veriyi önceden belirlediği bir biçimde dosyaya yazar ve okuması gerektiği zaman da verinin hangi biçimde kayıtlı olduğunu kendisi bildiği için buna göre okuma işlemini gerçekleştirirdi. Bu yöntemin bir dezavantajı vardı. Tutulan veriye yeni bir alan eklenmek istendiğinde (örneğin önceden sadece telefonları tutan bir dosyaya ev adresi bilgisinin eklenmesi gibi) bu veritabanında okuma yazma işlemlerini yapacak programların yapıları da değişmek zorundaydı.

Daha sonraları hiyerarşik ve ağ yapılı veritabanları ortaya çıkmıştır. Bunların ilkinde birbirleriyle ilişkili kayıtlar bir ağaç yapısındaki çocuk düğümler gibi tutulmuştur. Bu yapının üzerine geliştirilen ağ yapısında ise benzeyen veriler arasında geçiş yapılarak

belirli bir alan üzerinde arama yapıldığında tüm kayıtların getirilmesi sağlanmıştır. "Ağ" adı da bu aradaki bağlantıdan gelmektedir.

Sonunda günümüzde de kullanılan İlişkisel Model (Relational Model) ortaya konmuştur. Bu modelde statik olarak bağlı olan bilgi yerine dinamik olarak çıkarılabilen ilişkiler bulunmaktadır. Bu kullanım birçok yerde büyük esneklik sağlar ve Microsoft SQL Server gibi birçok modern veritabanı sağlayıcıları bu modeli kullanır.

Anlatılan ilişkisel modelin yanı sıra bu biçimde tutulan verinin üzerinde işlemler yapılabilmesi için bir dil de ortaya konmuştur. SQL, Yapısal Sorgu Dili (Structured Query Language), programlama bilmeyen insanların bile rahatlıkla anlayabilecekleri bir yapıda geliştirilmiştir.

SQL aslında üç adet dilin birleşiminden oluşmuştur:

1. DML (Data Manipulation Language): Veri üzerinde ekleme, silme, düzeltme gibi işlemlerin yapılmasını sağlar.

2. DDL (Data Definition Language): Verinin hangi alanlara sahip olacağını ve nasıl tutulacağına ilişkin işlemlerin yapılmasını sağlar.

3. DCL (Data Control Language): Verinin kimler tarafından ve nasıl erişileceğine ilişkin güvenlikle ilgili işlemlerin yapılmasını sağlar.

Tez çalışmasında askeri birimlere ait konum-zaman raporlarını MS SQL Server kullanarak bir veritabanında kaydedilip, bu veriler üzerinde çeşitli sorgular yapan bir arayüz geliştirilmiştir.

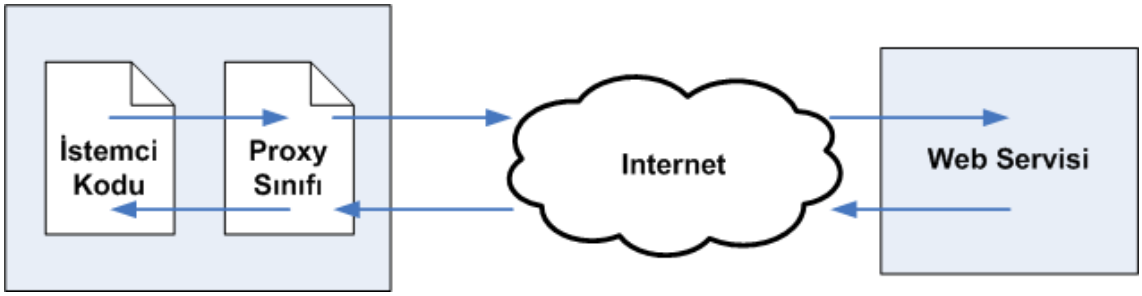
2.5 Web Servisleri

Ortak bir ağ üzerinde çalışan bilgisayarlar, üzerlerindeki hesaplama veya veri işleme yükünü sorunu başka bilgisayarlara aktararak azaltabilir. Web servisleri uzak bir makinedeki yordamların XML ve HTTP gibi ortak veri biçimleri ve protokoller aracılığıyla çağırılmasını sağlayan sınıflardır.

Günümüzde birçok sağlayıcı farklı web servisi uygulamaları sunmaktadırlar, bu şekilde Internet bağlantısı olan kurumlar var olan sorunlarına bu ağ üzerinden yaptıkları yordam çağrılarıyla cevap bulmaktadırlar.

.NET'te bu ağ üzerinden yapılan çağrılar SOAP (Basit Nesne Erişim Protokolü, Simple Object Access Protocol) ile yapılmaktadır. Bir endüstri standardı olarak benimsenen SOAP farklı platformlar arasında çalışmayı sağlar. Bu gücünü de HTTP ve XML gibi standartların üzerine kurulu olmasından alır.

Web servisinin bulunduğu makine *uzak makine* olarak adlandırılır. İstemci ağ üzerinden bir yordama çağrı yapar, bu yordam uzak makine üzerinde çalışır ve sonuçları XML biçiminde alır. İstemci ile web servisinin arasında bir *Proxy sınıfı* bulunur. Bu sınıf SOAP mesajlarının oluşturulması ve Internet üzerinden iletilmesi için gerekli işlemleri yapar.



Şekil 2.14 Web Servislerin çalışması

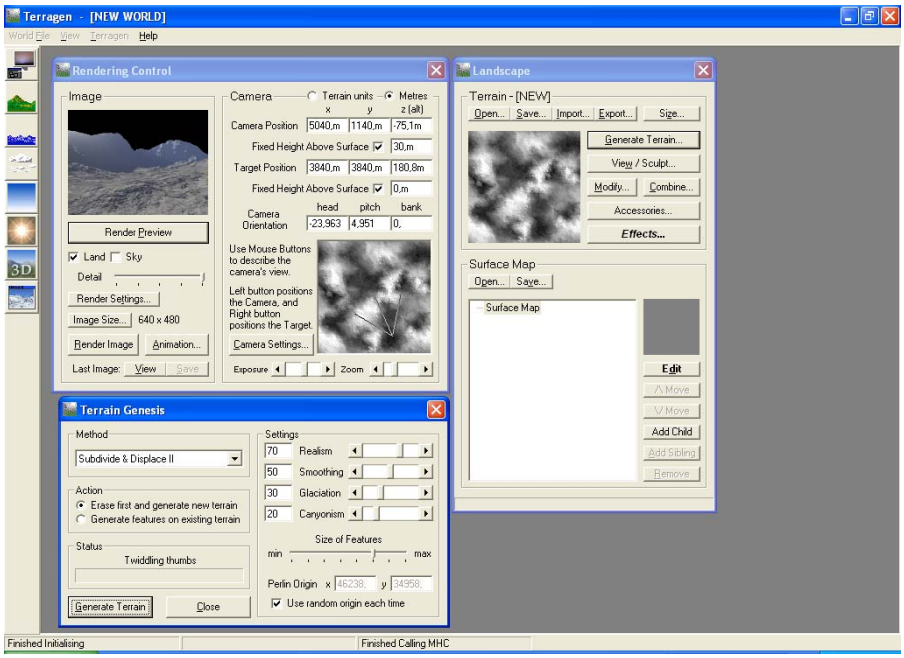
Şekil 2.14'te de gösterildiği gibi istemci bir yordam çağrısı yaptığında aslında *Proxy sınıfı*nda karşılık gelen bir yordama çağrı yapılmış olur. Bu yordam yapılan çağrı ile aynı ismi taşır fakat çağrının isteğini bir SOAP mesajı olarak iletir. Web servisi, yapılan isteği SOAP mesajı olarak alır, kodu çalıştırır ve sonuçları başka bir SOAP mesajı olarak geri gönderir. *Proxy sınıfı* gelen mesajı açar ve istemci koda iletir.

Tez kapsamında geliştirilen uygulamada web servisleri askeri birimlerin konum bilgilerinin iletilmesinde kullanılmıştır.

3. KULLANILAN YAZILIM VE ARAÇLAR

3.1 TerraGen

TerraGen farklı manzaralar oluşturmak için kullanılan tanınmış bir sahne oluşturma paketidir. Yazılımda, yükselti aralığı, arazi boyutları ve hatta yazılım içindeki rendering özelliği için güneş yüksekliğinin kullanıcılar tarafından ayarlanabildiği çok sayıda konuşma kutusu bulunmaktadır. Şekil 3.15'de yazılımdan bir görüntü verilmiştir:



Şekil 3.15 TerraGen yazılımından görünüm

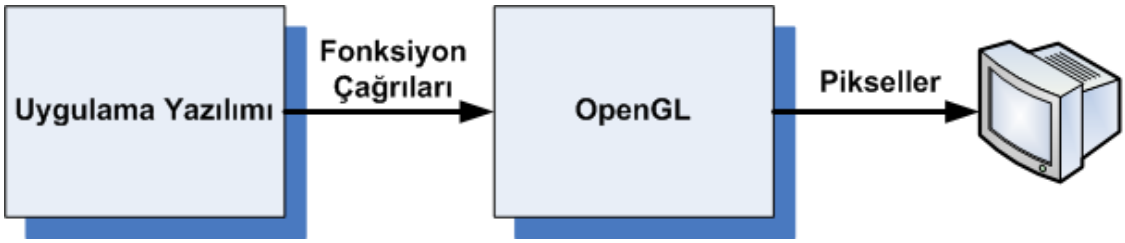
3.2 OpenGL

OpenGL, Silicon Graphics tarafından satıcıdan bağımsız 2B ve 3B grafiklerin geliştirilmesini sağlayan bir API'dir. Bu API kullanılarak geliştirilen programlar destekleyen bilgisayarlar arasında taşınabilirdir. Neredeyse tüm donanım ve işletim sistemleri için uygulamaları bulunmaktadır. Taşınabilirliğine ek olarak, OpenGL ile yazılan programların ömürleri de uzun olur. Bunun da sebebi yeni donanım özelliklerine yanıt verecek şekilde sürekli olarak geliştirilmesidir.

Programcı açısından OpenGL:

- Render edilecek nesne kümesini,
- Bu nesnelerin özelliklerini,
- Nesnelerin nasıl görüneceklerini belirledikten sonra elde etmek istediği görüntüleri oluşturabileceği bir kütüphanedir.

Bunun yanı sıra başka bir açıdan bakıldığında, OpenGL girdileri ve çıktıları olan bir durum makinesi olarak da görülebilir. Girdiler; doğru, poligon, resim gibi farklı geometrik nesneleri tanımlayacak olan fonksiyonlardır. Çıktılar ise bilgisayarın ekranında beliren görüntüdür. Arada çalışan bu makine nesnelerin tanımlarını alıp, işler ve resimlere çevirir. Bu çevirme işi de makinenin bulunduğu duruma göre değişir. Örneğin bir doğru parçası çizildiğinde, bunun rengi makineye verilen son renk bilgisi olacaktır. Yeni bir renk belirtilmesi, makinenin durumunu değiştirmektir. Verilen girdinin nasıl işleneceği de makinenin durumu ile belirlenir. Şekil 3.16 bu durum makinesini göstermektedir.



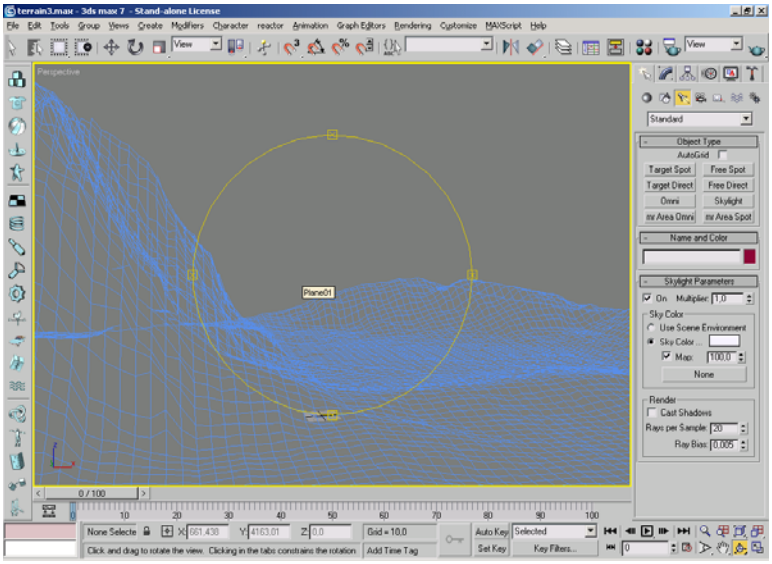
Şekil 3.16 OpenGL kütüphanesin çalışması

3.3 3D Studio Max

AutoDesk firmasının hazırladığı 3D Studio Max yazılımı güçlü bir entegre modelleme, animasyon ve rendering aracıdır. Kısa zamanda etkileyici grafikler oluşturmak için birçok oyun geliştiricisi, film yapımcıları ve birçok 3B ortam meraklısı tarafından kullanılmaktadır.

Hazır olarak sağlanan birçok basit nesnenin yanı sıra AEC modülünde sunulan ağaç modelleri gibi araçlar modelleme sürecini etkili bir biçimde kolaylaştırmaktadır. Bunun yanı sıra sağlanan poligon modelleme ve doku atama araçları da daha gerçekçi

modellerin oluşturulmasında kullanılmaktadır. Şekil 3.17'deki program görüntüsünden de anlaşılacağı üzere modelleme ve animasyon için birçok araç ve menü bulunmaktadır.



Şekil 3.17 3D Max yazılımı ve oluşturulan bir arazinin görünümü

Yapılan tez çalışmasında bu araç arazinin 3B olarak modellenmesi ve askeri modellerin işlenmesinde kullanılmıştır. İlerleyen kısımlarda bu işlemler detaylı olarak anlatılacaktır.

3.4 XNA

Microsoft firmasının isteyen herkesin rahat bir biçimde tasarladığı oyunları geliştirmelerinin sağlayan bir kütüphane olarak XNA kısa zamanda geniş bir kullanıcı kitlesine yayıldı. Bu platform sayesinde geliştiriciler hem Windows hem de Xbox 360 platformları için oyun geliştirebilmektedirler.

Tamamen oyun geliştirmek üzere tasarlandığından basit bir XNA programının yapısı genel olarak aşağıdaki gibidir:

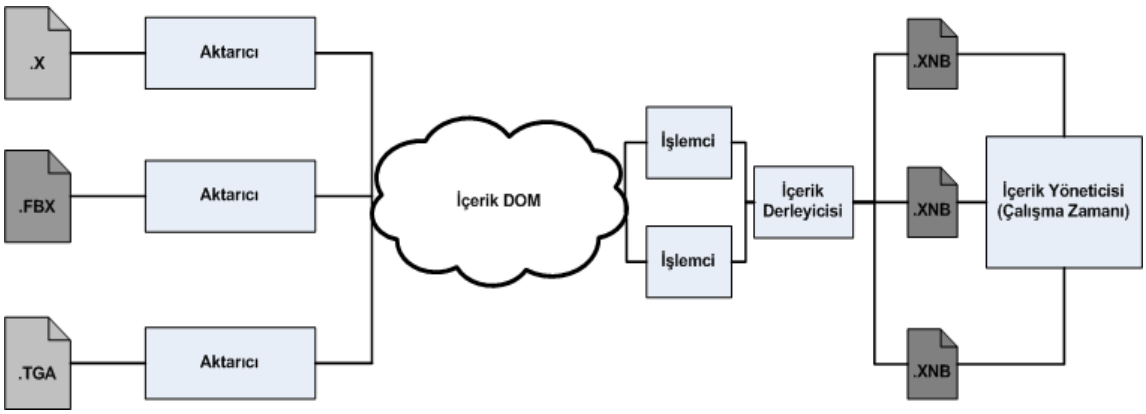
1. Grafiklerin, giriş araçlarının tanımlanması
2. Kaynakların belleğe yüklenmesi
3. Oyun döngüsünün başlatılması, her döngüde:
    a. Kullanıcı girdisinin alınması
    b. Gerekli hesaplamaların yapılması (konumlar, durumlar…)

c. Oyunun bitme şartının sağlanıp sağlanmadığının kontrol edilmesi
d. Çizimin yapılması

4. Oyunun bitirilmesi
5. Alınan kaynakların sisteme geri verilmesi

İlk adımda çizim yapılırken kullanılacak aygıt tanımlanır. Bu genel olarak bilgisayarın ekran kartıdır. XNA donanımsal kısımlarla ilgili bağlantıları otomatik olarak ayarlayıp geliştiriciye bir arayüz olarak *GraphicsDevice* nesnesi döndürür.

İkinci adım kullanılacak resim, doku, 3B modeller gibi kaynakların belleğe yüklenmesidir. Yine XNA bu işlemi, alt düzey işlemler için var olan *ContentManager*'ı kullanarak halleder.



Şekil 3.18 XNA ContentManager'ın yapısı

Şekil 3.18'de *ContentManager*'ın şeması gösterilmiştir. Verilen kaynağın türü ne olursa olsun XNA bunu işleyerek kendi türlerinden birine çevirir. Bu işlem kütüphanenin yapısında olan aktarıcılar tarafından yapılır. Aktarıcılar özel bir biçimdeki dosyanın içindeki bilgiyi okumak için hazırlanmış programcıklardır. Bu işlem sayesinde pek çok farklı türdeki dosya birer kaynak olarak kullanılarak geliştirme süreci hızlanır.

Üçüncü adımda oyun döngüsü başlar. Bu döngü içinde ilk olarak kullanıcının girdileri alınır. Bu girdiler, metin veya fare tıklamaları olabilir. Bu girdiler için örnek vermek gerekirse metin girdileri bir nesnenin özelliğini tanımlamak için kullanılırken, fare tıklamaları seçim yapma, yeni konum belirleme gibi işlemler için kullanılabilir. Alınan

girdiye göre, oyun içindeki nesnelerin yeni konumları gibi değişiklikler hesaplanarak son durum ekrana çizilir. Bu döngü oyunun bitme şartı sağlanana kadar işletilir.

Dördüncü adım oyunun bitirilmesi gerektiğinde yapılacak işlemleri kapsar. Genel olarak bakıldığında kullanıcıya sonuçların gösterilmesi, oyunun kaydedilip edilmeyeceğin veya tekrar başlamak istenip istenmediğinin sorulması bu adım içerisinde gerçekleştirilir.

Son adımda ise eğer yeniden başlatılmayacaksa kullanılan kaynaklar için ayrılan bellek bilgisayara geri iade edilir ve program sonlandırılır.

Tez kapsamında bu kütüphane Sandbox yazılımın savaş alanının ve bu savaş alanı üzerinde askeri birimlerin gösterildiği kısımda kullanılmıştır.

3.5 NATO-APP-6A

Bu standart gösterim ve çizim için gerekli uyumluluğu sağlayan ve NATO (Kuzey Atlantik Antlaşması Teşkilatı, North Atlantic Treaty Organization) C4I (Kontrol, Komuta, İletişim, Bilgisayar ve Haberalma) (Command, Control, Communications, Computer and Intelligence) sistemlerinin birlikte geliştirilebilerek çalışabilmelerini sağlayan bir harekât sembolojisidir.

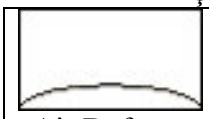
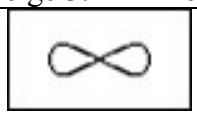
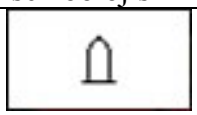
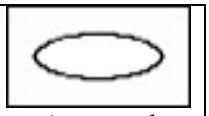
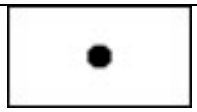
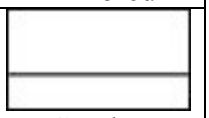
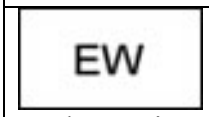
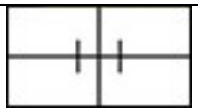
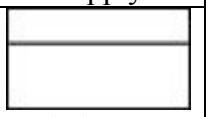
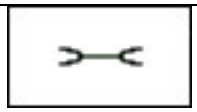
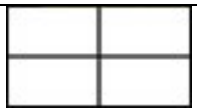
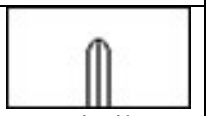
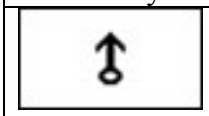
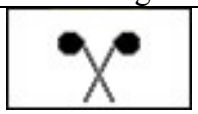
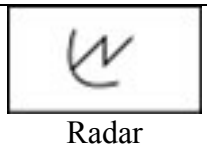
APP-6-A standardı, APP-6'nın yerini almıştır. Bu standardın içeriği, kara-tabanlı birimler için hem otomatik görüntü sistemleri hem de klasik harita işaretleme yöntemlerinde kullanılabilecek ortak bir askeri semboloji oluşturur.

Standart, NATO kara birimleri harekâtları kapsamında C4I işlemleri ile doğrudan veya dolaylı ilgisi olan tüm birimlere uygulanabilir. Harita çizimi, kokpit veya hava göstergeleri ve kullanılabildiği ölçüde mühendislik tasarım sembolojisinde kullanılabilir. Tüm gelecek 2B ve elektronik C4I görüntüleme sistemlerinde uygulanabilir.

APP-6-A, gelecekte yapılabilecek değişiklik ve geliştirmelere ayrıca kullanıcı ve operatörlerden gelen bilgilere uyum sağlayabilecek esneklikte tasarlanmıştır. Verilen sembolojide yapılacak değişiklikler ve eklenecek yeni sembol kümeleri NATO yönetmeliklerine göre düzenlenecektir.

Çizelge 3.2'de kullanılan semboller anlamlarıyla birlikte verilmiştir. Semboller orijinal isimlerine göre alfabetik sırada gösterilmektedir.

Çizelge 3.2 APP-6-A sembolojisinin kullanılan altkümesi

 Air Defence	 Air Force	 Ammunition	 Anti Tank	 Armored
 Army Aviation	 Artillery	 Bridging	 Combat Service	 Combat Supply
 Electronic Warfare	 Engineer	 EOD	 Hospital	 HQ Support
 Infantry	 Maintenance	 Medical	 Meteorological	 Missile
 Mortar	 MP	 Navy	 NBC	 Ordnance
 PHYSOPS	 Radar	 Reconnaissance	 Refuel	 Signals
 Special Forces	 Special Operations Forces	 Topographical	 Transportation	 Unmanned Air Recon

Yapılan tez çalışmasında anlatılan askeri semboloji standardının küçük bir alt kümesi, askeri birimlerin grafik alanında gösterilmesinde kullanılmıştır. Yazılımda yeni bir askeri birim eklendiğinde birime ait sembol belirtilen konumda görüntülenecektir. Ayrıca askeri birimler için gerekli olan bilgiyi tutan kısımların tasarımında da yol gösterici olmuştur.

3.6 3B Askeri Modeller

Bir önceki kısımda verilen sembolojiyle birlikte askeri birimlerin 3B modelleri de yazılıma eklenmiştir. Çizelge 3.3 kullanılan modelleri göstermektedir.

Çizelge 3.3 Kullanılan askeri modeller

 ABEJET	 APACHE	 ARTILLERY
 B52	 BRDM3	 F5F
 F15E	 GMC	 HEMTT
 M2A2	 M60	 M730A
 V100	 WILLYS	 WMCVUN

4. GELİŞTİRİLEN UYGULAMALAR

4.1 Arazi Görselleştirme (TerraVis)

Arazi Görselleştirme; oyun geliştirme, uzaktan algılama ve savaş alanı benzetimi gibi birçok uygulamada önemli bir kavramdır. Bu çalışmada güçlü arazi oluşturma ve modelleme araçları olan TerraGen, 3D Max ve OpenGL kullanılarak bir arazinin görselleştirilmesi amaçlanmıştır.

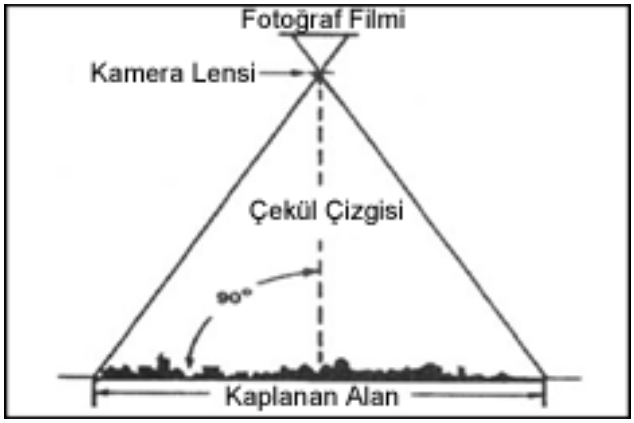
Geliştirme süreci iki ana kısma ayrılabilir. İlk kısım arazi ve arazide görüntülenecek olan ağaçların modellenmesi için kullanılan yöntemleri içerir. İkinci kısım ise 3B modelleri OpenGL'e yüklemeyi ve ekranda bu modellerin çizilmesi için geliştirilen algoritmayı ve programlamayı içerir.

4.1.1 Modelleme

Araziyi oluşturmak için TerraGen ve 3D Max araçları birlikte kullanılmıştır. Gerçekçi bir arazi verisinin etkili bir biçimde oluşturulması için yazılımların birçok farklı özellikleri bir araya getirilmiştir.

4.1.1.1 Yükselti verisinin oluşturulması

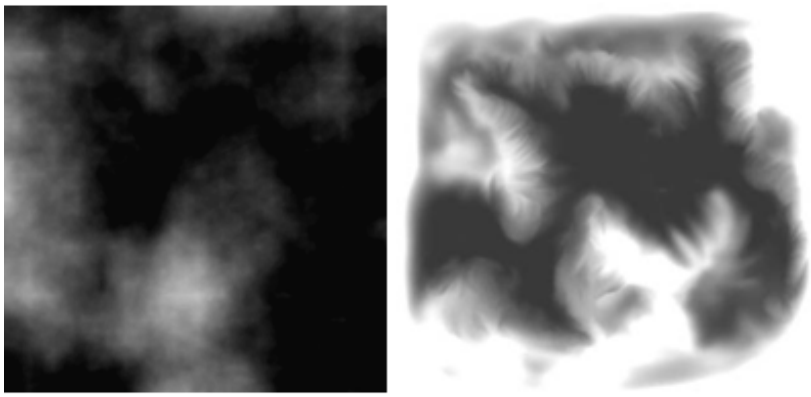
Çalışmada, TerraGen yükselti haritası oluşturmak için kullanılmıştır. Yükselti haritası, farklı gri tonlarının arazideki yükseltiyi belirlediği gri renkli bir resimdir. Bu uygulama uzaktan algılamada kullanılan yöntemlere benzer. Dikey fotoğraflar (Campbell, 2002), havadan direk olarak yere yönlendirilmiş bir kamera ile çekilir. Bu yöntem Şekil 4.19'daki gibi yeryüzünün harita benzeri bir görümünü verir.



Şekil 4.19 Havadan dikey fotoğraf

4.1.1.2 Arazi modelinin oluşturulması

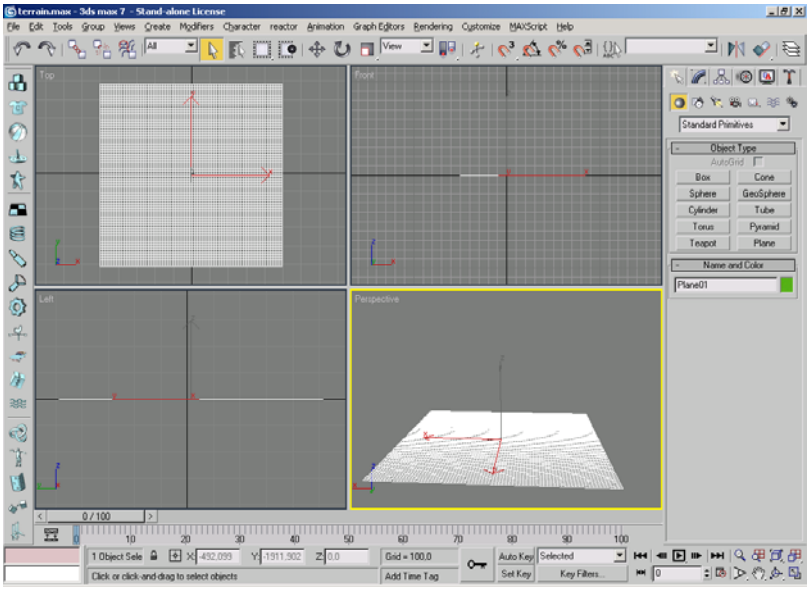
Arazi elde edilen yükselti haritası kullanılarak modellenmiştir. Yükselti haritasında farklı gri tonlar farklı yükselti seviyelerini belirler. Beyaz görünen kısımlar arazinin yüksek kısımlarını oluştururken, siyah kısımlar da en alçak yerleri oluşturur. Bu iki seviye arasında kalan gri tonları ise iki limit arasındaki boşluğu gri rengin tonuna göre farklı eğimlerde doldurur. Çalışmamızda Şekil 4.20'dekilere benzer yükselti haritaları kullanılmıştır.



Şekil 4.20 Kullanılan örnek yükselti haritaları

Verilen yükselti haritasından bir arazi oluşturmak için, 3D Max yazılımında ilk önce Şekil 4.21'deki gibi bir düzlem oluşturmak gerekir. Düzlem 3D Max menüsündeki temel ilkeller kısmından oluşturulur. Arazi için düzlem oluştururken dikkat edilmesi

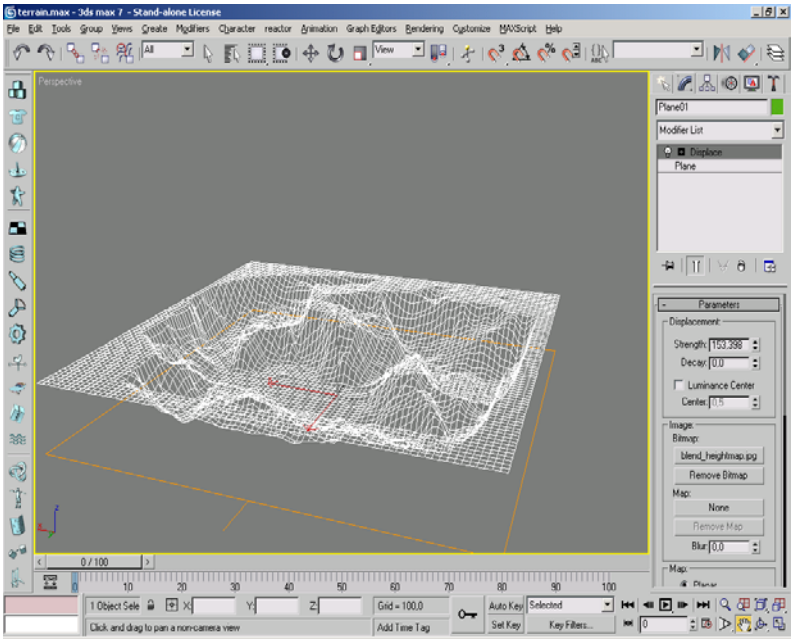
gereken bir nokta da dikey ve yatay uzunluk parçalarının sayısının belirlenmesidir. Bu parçalar harita kullanıldığında nokta sayısını belirleyecektir.



Şekil 4.21 3D Max'te düzlemin oluşturulması

Daha gerçekçi bir arazi modeli elde etmek için daha çok sayıda nokta kullanılarak tüm detaylar çizilebilir. Fakat nokta sayısı çok fazla arttırıldığında, render zamanı kötü bir biçimde etkilenerek uzayacaktır. Bu nedenle, asıl arazi verisindeki önemli detaylardan fedakârlık yapılmayacak şekilde en az sayıda nokta sayısı seçilmelidir. Bu sorun, daha sonra anlatılacağı gibi 3D Max'te bulunan özel bir değiştirici ile çözülmüştür.

3B arazi modelini oluşturmak için *Displace* (Kanbur, 2005) değiştiricisi kullanılmıştır. Bu değiştirici, önceden oluşturulan düzlem için yükselti haritasını 3. boyut (yükseklik) olarak kullanır. Bu işlemden sonra arazi Şekil 4.22'deki gibi elde edilir:

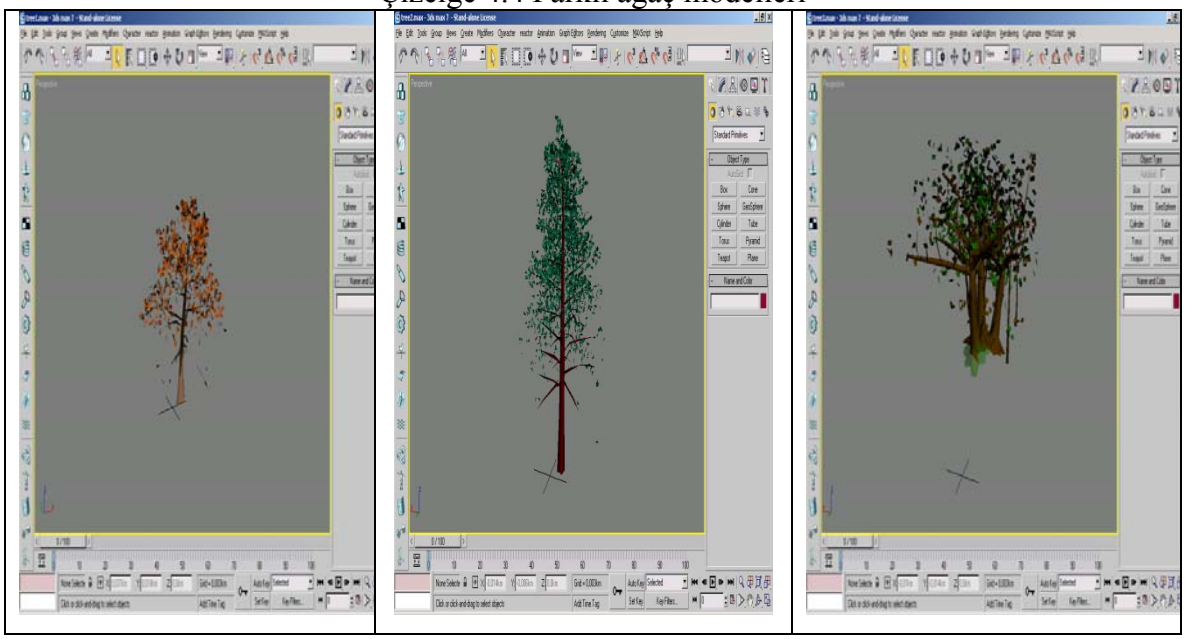


Şekil 4.22 Displace değiştiricisi ile yükselti haritasının düzleme uygulanması

4.1.1.3 Ağaçların modellenmesi

Geliştirilen yazılımda üç farklı ağaç tip kullanılmıştır. Bu ağaç modelleri 3D Max'in genişletilmiş AEC kısmında bulunmaktadır. Ağaç modelleri Çizelge 4.4'da verilmiştir. Bu ağaç modelleri arazi üzerine yerleştirilerek daha gerçekçi ve doğal görünümlü bir arazi elde edilmiştir.

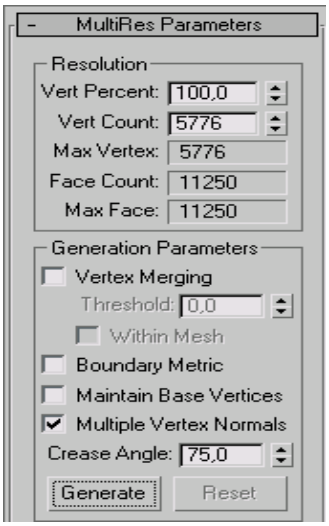
Çizelge 4.4 Farklı ağaç modelleri



4.1.1.4 Eniyileme işlemi ve aktarım

Arazi ve ağaç modelleri oluşturulduktan sonra, modelleri tanımlamak için kullanılan nokta sayısını azaltmak için eniyileyen bir değiştirici kullanılmıştır. Bu değiştirici aşağıdaki gibi çalışır:

1. Modele MultiRes (Kanbur, 2005) değiştiricisi uygulanır.
2. Değiştirici nokta ve yüzey sayısını hesaplar (Şekil 4.23).
3. Modeli yapan kişi uygulamaya göre nokta sayısını ayarlar.



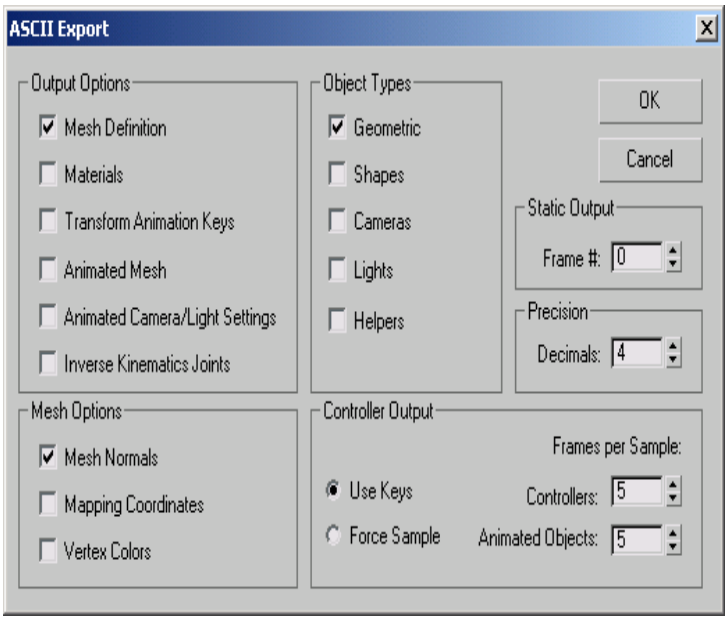
Şekil 4.23 Eniyileme işlemini yapan değiştiricinin parametreleri

3D Max'te oluşturulan modellerin kullanılabilmesi için, modellerin basit bir algoritma yardımı ile okunabileceği bir biçime aktarılması gerekir. İşlem sonunda oluşacak dosyanın basit ve anlaşılır olması için ASCII tipinde bir aktarma işlemi seçilmiştir. Bu biçimde oluşturulan dosya, yüzey ve nokta verisi içerir. Nokta bilgilerinin birer belirleyici numarası ve nokta için (x, y, z) koordinatları bulunmaktadır. Yüzey (uygulamada üçgensel yüzey seçeneğini kullanılmıştır) bilgisi bir nokta listesinden elde edilecek üç farklı noktayı temsil eden üç adet belirleyici numara içermektedir.

Örnek birer nokta ve yüzey bilgisi aşağıda verilmiştir:

*MESH_VERTEX 0 -504.0609 -513.7056 153.3978
*MESH_FACE 0: A: 76 B: 0 C: 77

Bu bilginin dosyadan okunması, modellerin yüklenmesi kısmında açıklanacaktır ama önce dosyanın bu biçimde elde edilmesi için gereken aktarma işleminin tamamlanması gerekmektedir. Aktarma menü seçeneği, ana menüden seçildiğinde Şekil 4.24'deki gibi bir ekran çıkar:



Şekil 4.24 Aktarma için gerekli parametrelerin seçilmesi

4.1.2 Gösterim

Oluşturulan modelleri OpenGL kullanarak çizmek için bu modellerin yüzey ve nokta listeleri olarak belleğe yüklenmesi gerekir, daha sonra bu nokta ve yüzeyler kullanılarak üçgenler çizilecek ve modeller ekranda görüntülenecektir.

4.1.2.1 Model bilgisinin yüklenmesi

Yükleme işlemi, aktarma sonrasında oluşturulan dosyadaki etiketlerden veri okunarak yapılır. Verinin yüklenmesinden önce, verinin tutulmasında kullanılan veri yapıları oluşturulmalı ve ilk değer ataması yapılmalıdır. *ModelType* adında bir veri yapısı iki tane dinamik dizi içerir. İlk dizi noktaları uzunluğu 3 olan vektörlerde tutar. İkinci dizi de modeldeki yüzeyleri saklar. Kullanıcı tanımlı bir *Triangle* türü bu yüzeyleri saklamak amacıyla oluşturulmuştur.

Yükleme işleminde ilk önce model dosyası yordam tarafından okunur. Dosyadaki nokta ve yüzey sayısı bu okuma işlemi sırasında elde edilir ve bu sayılar veri yapılarının dinamik olarak tahsis edilmesinde kullanılır. Farklı modellerde farklı nokta ve yüzey sayısı bulunduğundan tahsis etme dinamik olarak yapılmıştır.

Yüzey ve nokta sayısını dosyadan okuyan farklı yordamlar yazılmıştır. Okuyucu yordam uygulamada iki seçenek içerir: ya yüzey verisi ya da nokta verisi okuyacaktır. Yordam bu iki seçeneğe göre dallanarak okuma işlemini tamamlar.

Okuma işlemi sırasında önemli bir nokta da y ve z eksenlerinin OpenGL ve 3D Max'te yer değiştirmiş olmasıdır. Bu sorun basitçe okuma sırasında iki değerin yeri değiştirilerek çözülmüştür.

4.1.2.2 Çizim

Ağ yüzey (mesh) verisi yüklendikten sonra, yazılımda yüzey ve nokta listeleri tam olarak bulunmaktadır. Sıradaki adım, ağ yüzeyi oluşturmak için üçgenleri çizmektir. *ModelType* türünde bir nesneyi çizen yordam aşağıda verilmiştir:

```
Listedeki her yüzey için
{
  üçgenÇiz(yüzey->nokta1, yüzey ->nokta2, yüzey ->nokta3);
}
```

Verilen algoritmada gösterildiği gibi, yordam modeldeki nokta ve yüzeylere göre üçgenler çizmektedir.

4.1.2.3 Görünüm

Arazi klavye kullanılarak farklı açılardan görüntülenebilir ve dört ana yönde çevrilebilir. Ölçekleme de bir özellik olarak eklenmiştir ve bu sayede kullanıcıların araziyi farklı ölçeklerde görüntüleyebilmesi sağlanmıştır.

Arazideki yükseklikler her yükselti seviyesine göre renklendirilerek gösterilmiştir. En alçak kısım mavi, daha yüksek kısımlar yeşil daha yüksek kısımlar mor ve en yüksek kısımlar kırmızı ile belirtilmiştir. Bu özellik klavyeden ‘h’ tuşuna basılarak etkinleştirilir.

Bir tarama etkisi oluşturmak için sahneye hareket eden bir ışık eklenmiştir. Bu ışık tuş takımı kullanılarak arazinin farklı yerlerini aydınlatmak için hareket ettirilebilir. Işık için parametreler aşağıdaki biçimde ayarlanmıştır:

```
lightDiffuse[4] = {1, 1, 0, 1};
lightSpecular[4] = {1, 1, 1, 1};
lightAmbient[4] = {1, 1, 1, 1};
```

Işığın yanı sıra arazi görüntüsüne sis özelliği de eklenmiştir. Bu özellik, kırpmanın (clipping) etkisini azaltır ve sahneye sonbahar günbatımı gibi bir hava verir. OpenGL’deki doğrusal sis modeli (Neider, 1993) kullanılmıştır. Doğrusal modelde sis miktarı aşağıdaki gibi hesaplanır:

$$F = \frac{bitiş - z}{bitiş - başlangıç}$$

Verilen formülde z kamera ile nokta arasındaki uzaklığı belirlerken, başlangıç ve bitiş parametreleri sis etkisinin *başlama* ve *bitiş* noktalarını gösterir.

4.2 Karar Destek Aracı (MDT)

Bu çalışmada farklı askeri durumlarda kullanılmak ve karar vermede yardımcı olmak üzere matematiksel modellerin etkileşimli olarak kullanılabildiği bir yazılım geliştirilmiştir. Yazılımın temel unsurları daha önce anlatılan karar vermede kullanılan matematiksel modellere dayandırılmıştır.

4.2.1 Matematiksel modellerin uygulanması

Matematiksel modelleri uygulayan yazılım C++ Builder Bütünleşik Geliştirme Ortamı ile hazırlanmıştır. Kullanıcılar değerleri değiştirerek bu modelleri etkileşimli bir biçimde test edebilirler.

Karar matrisinde kullanıcılar ağırlıkları ve değerlendirmeleri verilen alanlara girerek değiştirebilirler. Kullanıcı gerekli bilgileri alanlara girdikten sonra "Score" tuşuna bastığında, Formül 2.2 ile hesaplanır. Bu hesaplama basit bir matris çarpımıdır. Sonuçlar görsellik amacıyla bir grafikte gösterilir.

Yazılımın ikinci kısmı da Lanchester Savaş Modelleri kısmıdır. Bu kısımda, kullanıcılar iki takım için ilk değerlerini girer ve kısım 2.2.3'de verilen parametreleri düzenleyebilirler.

İki takım için de aşağıdaki algoritma uygulanmıştır:

```
for(n=1 to numberOfDays)
{
 if(team[n-1] == 0)
          team[n]=0;
     else if(enemyTeam[n-1] == 0)
      team[n] = team[n-1];
     else
     {
```

```
      value = team[n-1] - team.battleLoses/100*enemyTeam[n-
     1] - team.operationalLoses/100*team[n-1] +
     team.reinforcements;
      if(value<1)
           team[n] = 0;
          else
           team[n] = value;
     }
}
```

Girilen her değere göre sistem yeniden hesaplanır. Kullanıcılar kuvvet sayılarındaki değişimleri hem grid'de hem de yazılımda sağlanan grafikte inceleyebilir. Bu parametreleri ayarlayarak ve sonuçları görerek, kullanıcılar farklı etkenlerin savaşın sonucunu nasıl etkileyeceğini anlayabilir.

Yazılımda uygulanan üçüncü model de Silahlanma Savaşı Modelidir. Bu kısımda kullanılan algoritma, dinamik özellikli nümerik bir sistemin basit bir uygulamasıdır. uygulanmasıdır. Bir döngü içinde, bir tarafın şu anki değeri o tarafın ve rakip tarafın bir önceki değerlerine göre hesaplanır:

```
for(n=1 to numberOfSteps)
        country[n] = rival[n-1] * countryData.coefficient +
countryData.assumption;
```

Yukarıda verilen algoritma sadece bir ülke için geçerlidir, fakat yazılımda değerler aynı algoritmayı kullanarak iki ülke için de hesaplanır.

4.2.2 XML verisinin kullanımı

Yazılımla sağlanan önemli bir özellik de XML verisinin kullanılmasıdır. Bu düşünce daha önce bazı sistemlerde (Arnett, 2004), (Hobbs, 2003) kullanılmıştır fakat tez çalışmasında yapılan katkı bunun bir karar matrisi ile birlikte kullanılmasıdır. Aşağıda sunulan Document Type Definition (DTD) ile XML verisinin yapısı gösterilmiştir:

```
<!DOCTYPE FIELDDATA
[
        <!ELEMENT FIELDDATA (DATA+)>
        <!ELEMENT DATA (OPTIONNUMBER, RATIOOFFORCES, TERRAIN, LOGISTICSUPPORT, RISK)>
        <!ELEMENT OPTIONNUMBER (#PCDATA)>
        <!ELEMENT RATIOOFFORCES (#PCDATA)>
        <!ELEMENT TERRAIN (#PCDATA)>
        <!ELEMENT LOGISTICSUPPORT (#PCDATA)>
        <!ELEMENT RISK (#PCDATA)>
]>
```

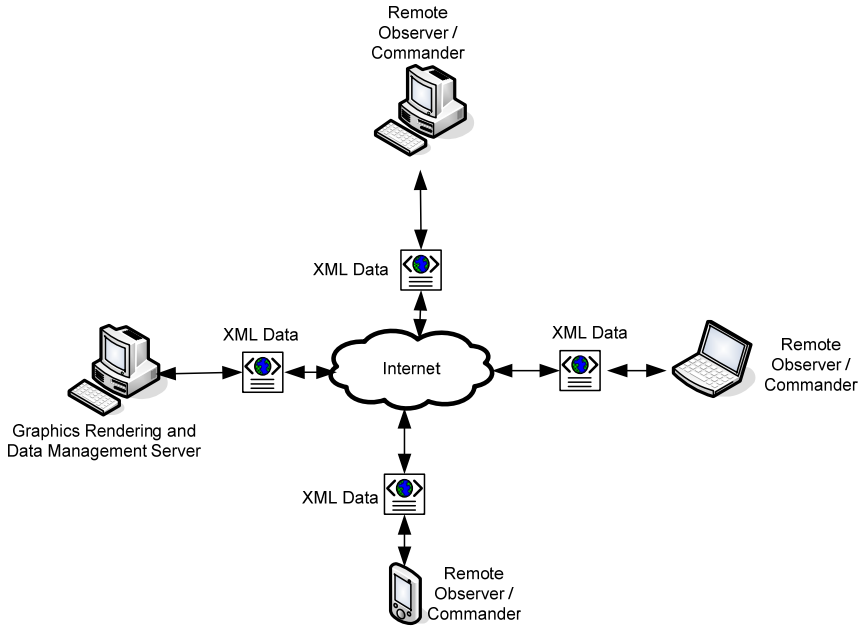
XML tanımında gösterildiği gibi, belge savaş alanı verisini kök eleman olarak tutmaktadır. Bu kök eleman belirli bir durumda düşünülebilecek birçok seçenek verisini çocuk eleman olarak tutar. Her seçenek kuvvet oranları, arazi, lojistik destek ve risk hakkında bilgi taşır.

Kullanıcı ilgili tuşa bastığında XML verisi dosyadan okunur. GetXMLData() yordamı bu amaçla çağrılır. Bu yordam basitçe XML belgesinin köküne erişir ve düğümleri (ve iki seviyeli bir hiyerarşi kullanıldığından bu düğümlerin çocuklarını da) tek tek gezer, değerleri alır ve kullanıcıya gösterir.

4.3 Sanal Kumsandığı Projesi (Sandbox)

Tez konusunun temelini oluşturan “Sandbox” adlı yazılım, .NET ortamında XNA kütüphanesi kullanılarak geliştirilmiştir. Yazılım ile kumandanların bir savaş alanını görüntüleyebilmesi ve uzak muhabere ekiplerinden alınan güncel bilgiler ile daha etkili kararlar vermesi sağlanmıştır.

Geliştirilen sistem, daha esnek bir yapı sağlamak ve dağıtık karar vericiler arasında bilgi iletişimini kolaylaştırmak amacıyla XML tabanlı veriler üzerinde çalışır. Sistemde hedef alınan mimari Şekil 4.25’de belirtilmiştir.



Şekil 4.25 Çalışmada temel alınan sistem mimarisi

Verilen mimariye göre birçok farklı yerden alınan rapor bilgileri verilen rendering sunucusunda işlenerek ekranda görüntülenir. Aynı zamanda sunucuda var olan birimlere ait veriler de diğer uzak kullanıcılara aktarılır.

Sistemin geliştirilmesinde izlenen adımlar aşağıdaki alt kısımlarda geliştirme sırasına göre ayrıntılı bir biçimde anlatılacaktır.

4.3.1 Kullanıcı arayüzü ve rendering ortamı

Grafiksel kullanıcı arayüzü bir Windows formu olarak tasarlandı. Bu form içinde bilgi girişi veri görüntüleme gibi alanların yanında, 3B çizim için kullanılan bir bileşen de bulunmaktadır.

XNA kütüphanesinin bir Windows formunun içinde gösterilmesi kütüphanenin geliştiricileri tarafından henüz tamamlanmadığından, çizim için gereken bileşen (*GraphicsDeviceControl*), *System.Windows.Forms.Control* sınıfının genişletilmesiyle elde edilmiştir. Oluşturulan sınıf ekran kartıyla ilgili alt seviye işlemlerden sorumlu olan *GraphicsDevice* sınıfını kullanır. Bu kullanım *GraphicsDeviceService* sınıfının

aracılığıyla yapılır. Adından da anlaşılacağı üzere bu sınıf, GraphicsDevice sınıfına erişmek için yordamlar sağlayan bir arayüzdür.

GraphicsDevice bileşeni oluşturulduktan sonra bilinen diğer form elemanları gibi formun üzerine yerleştirilip konum, boyut gibi ayarlar aynen diğer bileşenler gibi rahat bir biçimde ayarlanabilir.

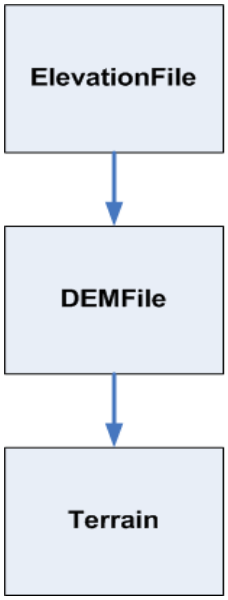
Genel XNA uygulamalarında kaynakların (modeller, resimler…) yüklenmesi bir *Content* projesi tarafından yürütülür. XNA bir form bileşeni olarak kullanıldığından bu projenin de ayrıca oluşturulup asıl projeye eklenmesi gerekmiştir.

Yapılan animasyonların da uygulanması bu *GraphicsDeviceControl* bileşeninin belirli aralıklarla sürekli olarak yenilenmesi ile sağlanmıştır. Bu konu konumları güncelleme animasyonu kısmında daha ayrıntılı olarak anlatılacaktır.

Bahsedilen işlemler yapılarak en basit anlamda çizim için gereken bir ortam sağlanmıştır.

4.3.2 Arazinin oluşturulması

Gerçek savaş alanındaki birimlerin üzerinde duracağı platform arazi olarak tanımlanır. Arazinin gerçekçi olması için örnek USGS DEM dosyalarından alınan yükselti verileri kullanılmıştır. Şekil 4.26 arazinin gösterilmesini sağlayan sınıfların yapısını göstermektedir.



Şekil 4.26 Arazinin oluşturulmasında kullanılan sınıflar

Arazi aslında üzerinde değerleri bir yükselti dosyasından alınan birçok veri noktası bulunan bir ağ yüzeyi yapısıdır. Bu yükselti dosyası DEM, DTED, yükselti haritası gibi birçok kaynaktan elde edilebilir. Bu nedenle, sistemin ilerde genişletilebileceği de düşünüldüğünden, genel bir yükselti sınıfı (*ElevationFile*) oluşturulmuştur. Bu sınıf birçok yükselti dosyasında ortak olarak bulunan dosya adı, coğrafi bölge, 2 boyutlu yükselti dizisi, arazide bulunan en düşük ve en yüksek yükselti değerleri, satır ve sütun sayısı gibi verileri içermektedir.

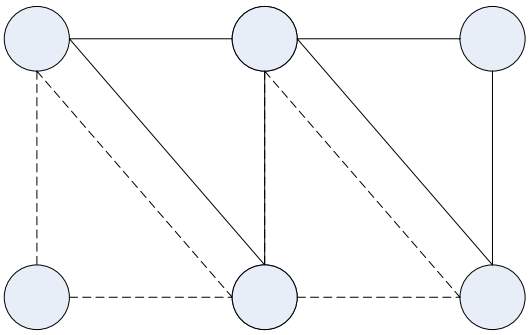
Bu genel sınıftan, DEM türü dosyalar için özel bir sınıf (*DEMFile*) türetilerek, bu tür bir yükselti verisi dosyasından yükseltiler alınmıştır. Arazi verileri gerçek dünyadaki verinin örneklenmiş hali olduğundan bu veriyi direk olarak kullandığımızda bazı sivri uçlu noktalar ortaya çıkabilir. Yükseltiler alındıktan sonra belirli bir aralığa ölçeklenerek daha yumuşak hatlara sahip bir arazi modeli elde edilmiştir. Bu ölçekleme işlemi çizim sırasında oluşabilecek sivri uçların yumuşatılmasını veya yok edilmesini sağlanmıştır.

Dosya okunduktan sonra, arazinin bilgisayarda gösterimi için gerekli veriyi tutan veri sınıf (*Terrain*) oluşturuldu.

Arazinin hem yükseltiye göre renklendirme yapılarak yükseltideki değişimler göz önüne daha açık bir biçimde konmuştur, hem de çimen, çöl ve kar gibi dokular kullanılarak yazılımda gerçekçi bir görünüm elde edilmiştir. Bu amaçla nokta pozisyon bilgisi, renk, normal ve doku bilgisi gibi bilgilerin bir arada tutulabileceği bir yapı oluşturulmuştur.

Kullanıcı tarafından bir dosya seçilerek açıldığında, seçilen dosyadan yükselti verisini okuyarak yükleyen bir yordam çağrılır. Nokta pozisyonları, renkler ve dokular hep bu veriye göre düzenlenir. Bu işlemden sonra bir indisleme işlemi yapılır.

İndisleme işlemi, çizilecek son ağ yüzeyi için gereken nokta sayısını azaltmak açısından son derece önemlidir. Şekil 4.27'de gösterildiği gibi ağ yüzeyi basit olarak üçgenler çizilerek oluşturulur. Normalde 4 tane üçgen çizmek için 4x3=12 (her üçgen için 3 tane olmak üzere) noktaya ihtiyaç duyulur. İndisleme işlemi kullanılarak bu 4 üçgen aynı noktaları iki veya daha fazla kez kullanılarak sadece 4 nokta ile de çizilebilir. Böylece grafik kartında arazi için ayrılacak bellek miktarı büyük oranda azaltılmış olur.



Şekil 4.27 Arazinin oluşturulması için veri noktaları arasındaki üçgenlerin çizimi

Bu indisleme işlemi, 2 boyutlu arazi verisi dizisinden 4'er nokta alınarak ve bu noktaları tek boyutlu bir diziye aktararak gerçekleştirilir. Aşağıda verilen kod parçası bu işlemi göstermektedir:

```
for(y = 0 ; y < height ; y++)
 for(x = 0 ; x < width ; x++)
{
 index = x+y*width;
 indices[count++] = index;
 indices[count++] = index + 1;
 indices[count++] = index+width+1;
```

```
 indices[count++] = index+width+1;
 indices[count++] = index+width;
 indices[count++] = index;
}
```

width ve *height* değişkenleri arazi boyutlarını tutarken, *indices* yukarıda bahsedilen ve çizim sırasında kullanılacak tek boyutlu diziyi gösterir. *index* değişkeni ise 2 boyutlu dizinin *x* ve *y* koordinatlarını belirlemek için kullanılan bir değişkendir.

Kullanılan sistem ayrıca basit aydınlanma veya ışıklandırma gibi etkilerin hesaplanması için her üçgenin yüzey normali bilgisine de ihtiyaç duymaktadır.

Son olarak arazi için hesaplanan nokta bilgisi, arazi verisi görüntüleme boyunca-yeni bir arazi verisi açılana kadar- sabit kalacağından grafik kartının hızlı belleklerine yazılır. Ayrıca tanımlanan bir *nokta arabelleğine* (*vertex buffer*) ve bir *indis arabelleğine* (*index buffer*) hesaplanan bilgiler kopyalanarak bu işlem tamamlanır.

Terrain sınıfı arazinin ilk çizileceği zaman veya görüntülenirken herhangi bir değişiklik yapılacağı zaman yeniden çizilebilmesi için bir çizim yordamı da sağlar. Burada görüntüleme sırasındaki değişikliklere örnek olarak kullanıcının araziyi farklı görüş açılarında görüntülemek için döndürmesi, arazinin çiziminin yükselti renklendirmeden doku kullanılarak çizime geçişi veya farklı dokular arasındaki geçişler verilebilir.

4.3.3 Birim sistemi ve semboloji

Yazılım, savaş alanındaki birimlerin görüntülenmesi amacıyla geliştirildiği için oluşturulan sanal birimlerin gerçek dünyadaki karşılıkları hakkında yeterli bilgiyi saklamaları gerekir. Ek olarak, grafik alanda çizilen her birimin birbirinden rahatça ayrılabilmesi gerekmektedir. Daha önce geliştirilen bir sistemde (Arnett, 2004) olduğu gibi herhangi bir renklendirme veya semboloji kullanılmadan yalnızca şekillerden birim hakkında bilgi çıkarılması yeterli bir yöntem değildir.

Bu nedenle, NATO-APP-6-A (Thibault, 2005) sembolojisi kullanılarak sistemdeki her birimin görüntülenmesi sağlanmıştır. Bu gösterim biçimi, hem tüm birimler hakkında

yeterli ve gerekli bilginin tutulması hem de grafik alanında birimler arasında dost veya düşman gibi ayrımların yapılabilmesine olanak sağlamıştır.

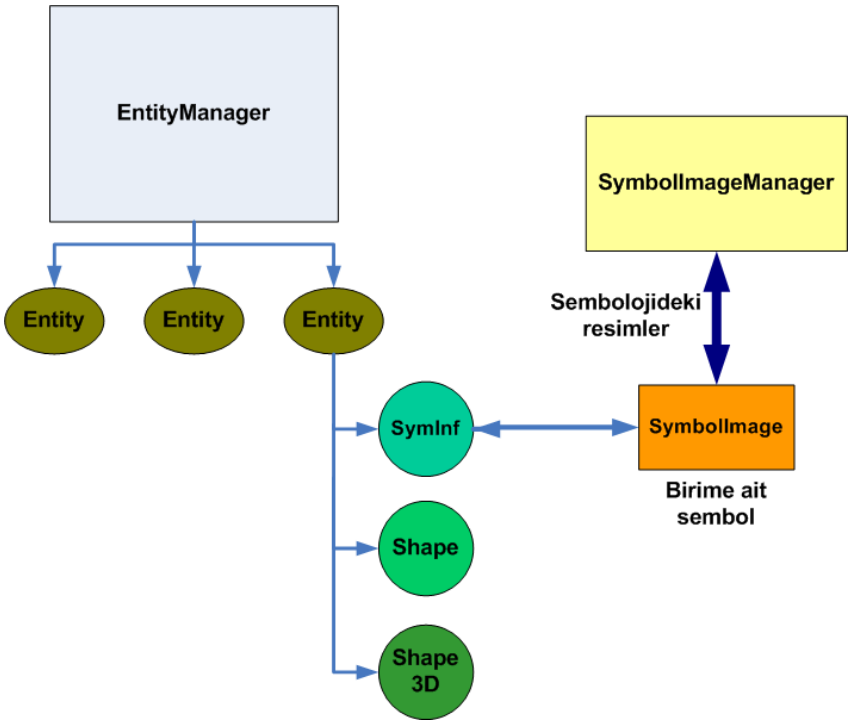
Yukarıda belirtilen sembolojiye bağlı kalınarak geliştirilen gösterim biçimi ile birimler etkili bir biçimde ifade edilmiştir. Aşağıda bir birim için saklanan bilgi yer almaktadır:

```
<Entity>
     <ID />
     <Affiliation />
     <Type />
     <Quantity />
     <Reinforcement />
     <StaffComments />
     <AdditionalInformation />
     <EvaluationRating />
     <CombatEffectiveness />
     <ReportDateTime>
          <Date>
               <Day />
               <Month />
               <Year />
          </Date>
          <Time>
               <Hour />
               <Minute />
               <Second />
          </Time>
     </ReportDateTime>
     <Position>
          <X />
          <Y />
          <Z />
     </Position>
</Entity>
```

Gösterilen yapıda, *ID* birimlere has bir belirleyicidir. *Affiliation* dost, düşman, bilinmeyen veya nötr değerlerinden birini alabilir ve grafik alanda sırasıyla mavi, kırmızı, yeşil ve sarı renklerle gösterilir. *Type* birimin piyade, top veya keşif gibi türünü gösterir. Birim türü ayrıca sembolojiye bağlı kalınarak basit bir kutunun üzerine doku olarak atanmıştır ve grafik alanda gösterilir. Sayı, yorumlar, ek yorumlar gibi bilgiler de sembolojide bulunduğundan bu alanlar da birim verisine eklenmiştir. *Evaluation rating* gelen birim bilgisinin doğruluğunu gösteren bir ölçüdür ve savaş alanında bulunan

belirsizliğin gösterilmesi açısından önemli bir unsurdur. Bir kumandanın gerekli tüm bilgileri göreceli doğruluk paylarıyla bilmesi gerektiğinden ve tüm bir sistemde matematiksel modellerin de eklenmesi gerektiği düşünüldüğünden bu unsur gerekli bilgi kısmına eklenmiştir.

Combat effectiveness alanı birimin çalışma kapasitesini gösterir. Rapor tarihi ve zamanı da son gelen raporun güncel olup olmadığını kontrol etmek için eklenmiştir. Son olarak, birimlerin koordinatlarını gösteren pozisyon bilgisi de eklenmiştir.



Şekil 4.28 Birimlere ait yapıların gösterimi

Birim nesnesinin *SymbolInformation* türünde bir nesnesi vardır. Bu nesne, içerisinde sembolojiye ait bilgileri tutar. Bu bilgiler daha önce bahsedilen sembolojideki alanlardır. Nesnenin içinde *Affiliation* türündeki nesneler birimin hangi taraftan olduğu bilgisini tutar. Benzer şekilde, birimin görüntülenmesinde kullanılan resim bilgisi *SymbolInformation* sınıfında tutulur. Ayrı bir sınıf olarak tasarlanan *SymbolImageManager*'da sembolojide kullanılan tüm resimler tutularak, kullanıcının çalışma zamanında bir birimin sembolünü değiştirmek istediğinde resim bilgisinin yüklenmesi için ayrıca zaman kaybetmesi önlenmiştir.

Uygulamada birim nesnesi grafik alanında görüntülenmek üzere *Shape* türünde bir nesneye sahiptir. Bu nesne, üzerinde sembolün doku olarak atanacağı basit bir kutuyu göstermek içindir. Bu nesnenin içinde dönüşüm bilgileri de bulunmaktadır. Dönüşüm bilgisi model için öteleme, ölçekleme ve dönme gibi bilgileri içerir. Her birimin render edilmesi için gerekli olan *view*, *projection* ve *world* matrisleri de bu sınıfta tutulur.

Benzer olarak bir *Entity* nesnesinin içinde *Shape3D* türünde bir nesne bulunmaktadır. Bu da *Shape* nesnesine benzer şekilde çalışır ama bu nesne basit bir kutu saklamak yerine modele ait 3B nesneyi tutar. Bu iki sınıf sayesinde aynı konumda istenirse sembolün istenirse de 3B modelin gösterilmesi sağlanmıştır.

Kullanıcı tarafından ve gelen bilgi ile oluşturulan birimlerin düzenli bir biçimde tutulması için *EntityManager* adında kapsayıcı bir sınıf oluşturulmuştur. Bu sınıf, seçilen bir birimi etkin hale getirmek için gereken işlemleri yapar ve bu bağlamda yazılımda bahsedilen "Birim Sistemi"ni oluşturur.

Bu kapsayıcı sınıfta (container class), seçilen birimin hızlı bir biçimde döndürülmesi için *SortedList* türünde bir veri yapısı kullanılmıştır. Bu sıralı listede, istenen birime herhangi bir biçimde ulaşmak gerektiğinde, örneğin kullanıcı fare ile ekranda bir birimi seçtiğinde veya gelen rapora göre sıra ile konum güncellemesi yapılacağı zaman, birim *ID* numarasına göre aranıp ve bulunduğunda etkin birim olarak belirlenmesi sağlanmıştır.

Bahsedilen sınıflar sayesinde askeri semboloji yazılımla bütünleşik bir hale getirilirken, sistemdeki askeri birimlerin de etkili bir biçimde tutulması sağlanmıştır.

4.3.4 Konumları güncelleme animasyonu

Geliştirilen yazılımda önceleri, seçilen bir birim için konum güncellemesi yapıldığında, birim hemen yeni konumunda beliriyordu. Bu gösterim, son konumların görüntülenmesi açısından kabul edilebilir olmasına rağmen, birimin hareket yönünün kestirilebilmesi açısından yetersiz kalmıştır. Buna ek olarak, yazılımın görselliğinin daha ön plana

çıkarılması açısından da verilen konum değişikliğinin bir öteleme animasyonu şeklinde gösterilmesi gerekmiştir.

Güncelleme işlemi daha önce açıklanan ve birimlerin tutulmasından sorumlu olan *EntityManager* sınıfının *UpdateEntities()* yordamı aracılığıyla yapılır. Bu yordam sıra ile tüm birimleri kontrol ederek, eğer herhangi bir konum güncellemesi gerekiyorsa aşağıda belirtilen şekilde güncelleme yapar.

Bir form içinde gösterilen birimlerin hareket etmeleri (öteleme animasyonu), formdaki çizim alanının sürekli olarak güncellenmesi ile sağlanmıştır. Normalde XNA uygulamalarında bu güncelleme hazır olarak yer almaktadır ve yapılacak olan güncelleme işlemleri programcıya sağlanan *Update()* yordamı içinde yapılmaktadır.

Geliştirilen uygulamada, çizim alanı bir form içine gömüldüğü için güncelleme işleminin otomatik olarak yapılması aşağıda gösterilen kod parçası ile sağlanmıştır:

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

Yapılan bu atama ile, uygulama çalışırken, işletilecek bir olay olmadığı sürece çizim alanındaki görüntü yeniden çizilecektir. Her çizim esnasında da önce yukarıda açıklanan *UpdateEntities()* yordamı çağırılır. Birimlerin konumlarını animasyon olarak değiştirecek olan bu yordam, her birimin ilk ve son konum bilgilerine ulaşır. Bu konumlar arasındaki ara değerleri doğrusal interpolasyon ile bularak her yenilemede bu değeri yeni konum olarak ayarlar.

Doğrusal interpolasyon için XNA tarafından sağlanan *MathHelper* sınıfının *Lerp(...)* yordamı kullanılmıştır. Bu yordam üç adet parametre alır. Bu parametreler sırasıyla birinci değer, ikinci değer ve artış miktarını gösteren değerdir.

Belirtilen yordam aşağıdaki formülü kullanarak iki değer arasında doğrusal interpolasyon yapar:

$$ilkKonum + (yeniKonum - ilkKonum) * artış$$

Verilen formülde *artış 0* ve *1* arasında değerler alır ve her seferinde ikinci değere, bizim durumumuzda sonKonum'a ne kadar yaklaşılacağını gösterir. Formülde *artış* olarak *0* değeri verildiğinde *ilkKonum*'un, *1* değeri verildiğinde ise *sonKonum*'un elde edileceği görülmektedir. Aşağıdaki kod parçası formülün uygulanmasını göstermektedir:

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

Burada *calculatedPosition*, her hesaplamada elde edilen konum bilgisini tutmaktadır. Yenileme sürekli yapıldığından bu değer de sürekli olarak küçük artışlarla son konuma doğru yaklaşır ve son konuma ulaştığında da durur.

4.3.5 3B modellerin eklenmesi

Yazılımda gerçekçiliğin arttırılması için kullanılan sembolojinin yanı sıra, bazı askeri birimlere ait 3B modeller de gösterime eklenmiştir. Bu sayede hem sembolojinin öğrenimi kolaylaştırılmış hem de yazılımın görselliği arttırılmıştır.

Modellerin kullanılabilmesi için *Shape3D* adında ayrı bir sınıf oluşturulmuştur. *Entity* sınıfının bir elemanı olarak çalışan bu sınıfta modelin dönüşüm bilgileri tutularak birimin olduğu pozisyonda gösterilmesi sağlanmıştır.

Yazılım kısmında modellerin yüklenmesinden önce tüm modeller 3D Studio Max yazılımında düzenlenerek, poligon sayıları azaltılmıştır. Bu işlemde daha önce belirtilen yöntemler kullanılmıştır. MultiRes değiştiricisi ile en uygun poligon sayıları belirlenerek modellerdeki poligon sayıları azaltılmıştır.

Daha sonra modeller, XNA'de kullanılabilen biçimlerinden biri olan FBX biçimine dönüştürülmüştür. Bu biçime çevirmek için aşağıdaki adımlar izlenmiştir:

1. Orijinal model projenin Content klasörüne atılır.
2. Ardından model bu klasörden seçilerek 3D Max programıyla açılır
3. Daha önceden yazılıma eklenti olarak yüklenmiş olan Panda Exporter aracılığıyla .fbx biçiminde dışa aktarılır.
4. Aktarma işlemi sırasında modele ait doku resimleri (texture) 2^n x 2^n olarak belirlenir.
5. Son olarak da elde edilen dokular modelin proje içindeki klasörüne eklenir.

Bu biçimde olan dosyalar doğrudan aktarılarak XNA içinde kolaylıkla kullanılabilir.

Elde edilen modellerin görüntülenmesinden önce, modellerin arasındaki modellemeden kaynaklanan birçok farklılıklar mümkün olduğunca azaltılmaya çalışılmıştır. Bu farkların başında modelin ölçeği gelir. Örneğin; bir tank modeli, bir jeep modelinden kat kat küçük olabilmektedir. Bu sorun modellerin farklı kişiler tarafından ve belirli bir ölçeğe bağlı kalınmayarak hazırlanmış olmasından kaynaklanır. Bu gibi gerçeğe uygun olmayan durumların önüne geçilebilmesi için modeller bir araya getirilerek göze en uygun gelen ölçekler belirlenmiştir. Farklı modeller için ayrı ayrı yapılan bu işlem zahmetli olduğu gibi, işlem sonunda kesin bir sonuç edilememiştir. Buna rağmen en uygun oranlar kullanılmaya çalışılmıştır.

Ölçeklendirmenin yanı sıra, modellerin hareketinde kullanılacak olan model doğrultuları $+z$ ekseni $(0, 0, 1)$ yönünde belirlenmiştir. Bu referans noktasından yola çıkılarak, daha sonra yapılacak olan güncellemeler sonucunda hareket edecek modeller varılacak olan nokta doğrultusunda o yöne bakacak şekilde hareket edeceklerdir. Bu işlem sonunda da modeller için başlangıç dönme açıları belirlenmiştir. Bu açılar daha sonra yapılacak yönlendirme hesapları için başlangıç değeri olarak kullanılacaktır.

Anlatılan işlemlerden sonra modeller için en uygun olarak belirlenen dönüşüm bilgileri aşağıda Çizelge 4.5'de verilmiştir.

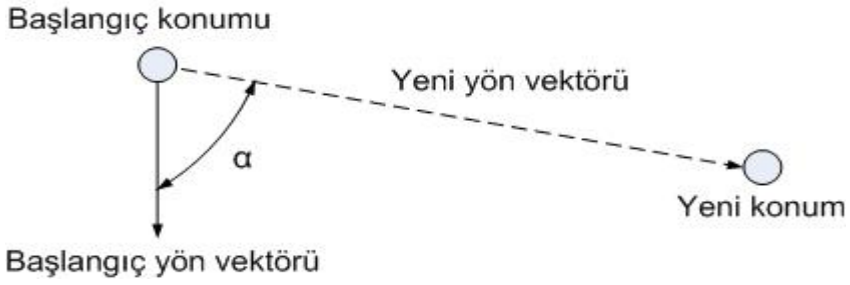
Başlangıç ölçekleri ve dönme yönleri belirlenen modeller, arazi üzerine eklenen askeri sembol türüne göre gösterilir. Kullanıcı açılan bir menü ile sembolü veya 3B modeli görüntüleyebilir.

Çizelge 4.5 Kullanılan modeller için varsayılan dönüşüm değerleri

Model	Ölçek			Dönme		
	X	Y	Z	X	Y	Z
ABEJET	0.5	0.5	0.5	0	0	0
APACHE	40	40	40	0	Π/2	0
ARTILLERY	80	80	80	0	Π/4	0
B52	8	8	8	0	0	0
BRDM3	1	1	1	0	0	0
F5F	0.5	0.5	0.5	0	0	0
F15E	0.5	0.5	0.5	0	0	0
GMC	7	7	7	0	0	0
HEMTT	0.5	0.5	0.5	0	0	0
M2A2	0.2	0.2	0.2	0	0	0
M60	0.8	0.8	0.8	0	0	0
M730A	0.8	0.8	0.8	0	0	0
V100	50	50	50	0	0	0
WILLYS	0.2	0.2	0.2	0	0	0
WMCVUN	1.3	1.3	1.3	0	0	0

Askeri birimlerin pozisyon bilgisi değiştiğinde bir önceki kısımda anlatılan birimlerin öteleme hareketi 3B modellerde de uygulanmıştır. Fakat bu durumda direk olarak sembolü hareket ettirmediğimizden elimizdeki modelin yönünün de hareket edilen yöne doğru dinamik bir biçimde değiştirilmesi gerekmektedir.

Hareket yönüne doğru modelin baş kısmını döndürmek için gereken açı iki vektör arasındaki açı hesaplanarak bulunmuştur. Bu durum Şekil 4.29'da gösterilmiştir:



Şekil 4.29 Konum ve yön vektörleri

Verilen şekilde başlangıç yön vektörü a ile ve yeni yön vektörü de b ile gösterilirse aradaki açı, iki vektörün nokta çarpımı formülünden hesaplanabilir:

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

Formülde $|a|$ ve $|b|$, a ve b vektörlerinin uzunluğunu gösterir. α ise aradaki açıdır. Bu formül aşağıdaki gibi yeniden düzenlenirse, α açısı kolay bir biçimde bulunabilir.

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

Döndürme için gereken açının bulunması için ilk önce birimin gideceği yön belirlenir. Yön vektörü iki konum arasındaki fark vektörü olarak tanımlanır. Bu işlemler yazılımda aşağıdaki şekilde yapılmıştır:

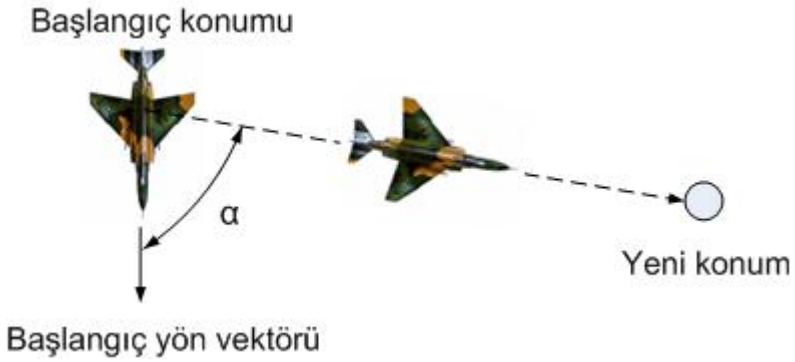
```
position2D = new Vector2(position.X, position.Z);
newPosition2D = new Vector2(newPosition.X, newPosition.Z);
newDirection2D = Vector2.Subtract(newPosition2D,
position2D);

cosAlpha = Vector2.Dot(newDirection2D, heading2D) /
(newDirection2D.Length() * heading2D.Length());
alpha = MathHelper.ToDegrees((float)Math.Acos(cosAlpha));
```

Döndürme açısı da bulunduktan sonra bilinmesi gereken tek şey döndürmenin hangi yönde yapılacağıdır. Buna da yeni konumun *x* değerine bakılarak karar verilir. Bu değer sıfırdan büyükse saat yönünde, değilse saat yönünün tersinde bir dönme yapılacaktır. Aşağıda bu işlem gösterilmiştir:

```
if(newDirection2D.X<0)
     entity.SetRotation(new Vector3(0, -1 * alpha, 0));
else
     entity.SetRotation(new Vector3(0, alpha, 0));
```

Verilen kod parçasında *newDirection*, birimin gideceği yönü gösterir. Açı yukarıdaki formüle göre hesaplandıktan sonra hangi yönde dönüleceği belirtilerek modele dönme uygulanmış olur. *SetRotation(...)* yordamı birimin dönüşüm matrisindeki bu değeri

verilen değer olarak atar. Bu işlem de yapıldıktan sonra 3 boyutlu model aşağıda Şekil 3.20 gösterildiği gibi önce bir dönme yapar ve daha sonra adım adım yeni verilen konuma doğru ilerler.



Şekil 4.30 Modelin yeni konuma doğru yönelmesi

4.3.6 Senaryo yapısı ve XML veri işleme

Bir savaş alanı benzetim yazılımında kullanıcıların daha önceki bir zamana dönerek önceden kullanılan bir stratejiyi ve sonuçlarını inceleyebilmesi gerekir. Bu şekilde kumandanlar yanlış bir stratejinin sonuçlarını görerek bu stratejiden kaçınmanın veya onu geliştirmenin yollarını ararlar.

Sunulan senaryo yapısı yukarıda belirtilen amaçlar için tasarlanmıştır. XML verisi kullanılarak, Sandbox yazılımı bir durumu kaydeder ve kullanıcı bir senaryo dosyasını açmak istediğinde geri getirir.

(Julier, 1999) ve (Durbin, 1998)'de yapılan çalışmalarda zaman bilgisi sistemde yer almıyordu ve sistemin önceki durumları kaydedilmiyordu. Yapılan çalışmada zaman bilgisi de senaryo yapısına katılarak, sistemin belirli zamanlardaki kaydının getirilmesi sağlanmıştır.

Sistemde XML verisinin işlenmesi, .NET tarafından sağlanan *XMLDocument* sınıfı kullanılarak yapılmıştır. Bu sınıf bir XML belgesinin açılması, elemanlara eleman adıyla ulaşılmasının sağlanması, okuma ve dosyadaki elemanları içeriğinin güncellenmesi gibi işlemler için yordamlar sağlamaktadır.

Aşağıda gösterilen kod parçası birimlere karşılık gelen elemanların okunmasını gösterir:

```
entitiesNode =
xmlDocument.DocumentElement.SelectSingleNode("Entities");
XmlNodeList nodeList =
entitiesNode.SelectNodes("Entity");
foreach(XmlNode entityNode in nodeList)
   {
      ReadEntity(entityNode);
   }
```

Gösterilen kod parçasında, *entitiesNode* sistemde kaydedilen tüm elemanlara karşılık gelir. Bu değişken, *SelectNodes()* yordamı ile "Entity" isimli elemanların tümü ile doldurulur. Bu ana düğümün tüm çocukları *ReadEntity()* yordamı ile okunur.

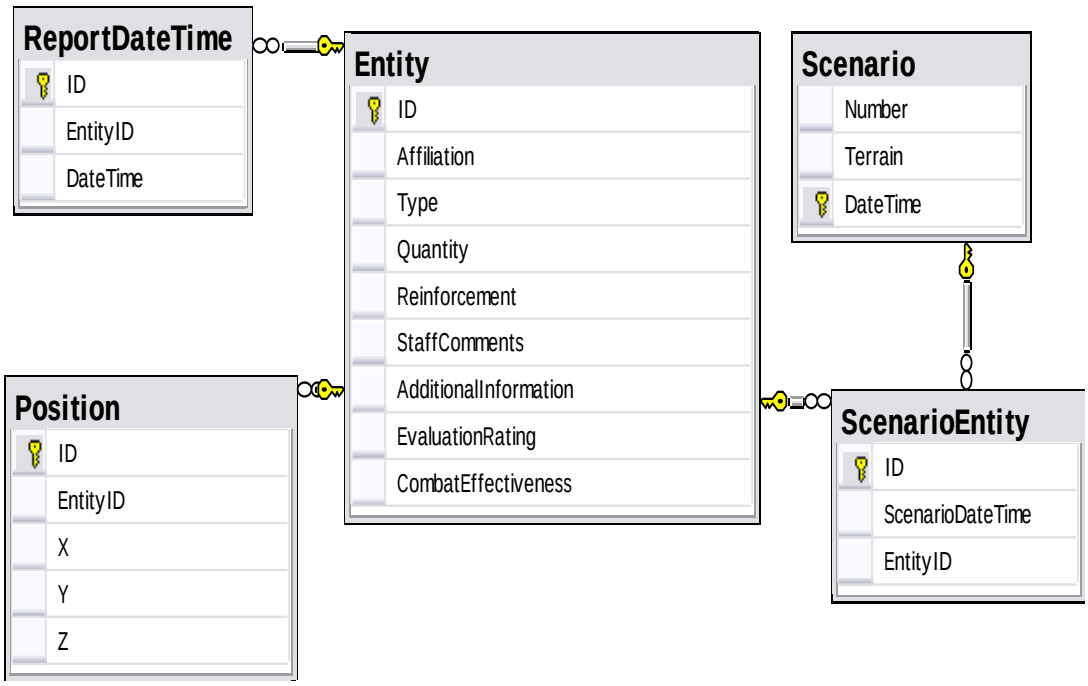
Benzer şekilde, o anki *EntityManager*'da bulunan tüm elemanlar XML senaryo dosyasına *WriteEntity()* yordamı kullanılarak yazılır.

4.3.7 Veritabanı

Geliştirilen Sandbox benzetim yazılımı askeri birimlere ait bilgilerin bir veritabanı içinde saklanmasını da sağlamıştır. Bu sayede kullanıcıların, belirli askeri durumları uzun süreli kayıt için tutabilmeleri ve daha sonra nesne tanımlarını (ID) gibi bazı kriterleri kullanarak belirli nesneler hakkında bilgiye ulaşmaları sağlanmıştır. İlişkisel veritabanı Microsoft SQL Server 2005 ile tasarlanıp geliştirilmiştir.

Veritabanı tasarlanırken daha önce bahsedilen ve yazılım için bütünleşik bir halde yer alan senaryo yapısı göz önünde bulundurulmuştur. Her askeri durum bir senaryo olarak görülmüş ve veritabanına kaydedilmiştir. Tasarımda izlenen yöntem aşağıdaki gibi açıklanabilir:
Herhangi bir askeri durum, bir savaş alanı ve üzerindeki askeri birimleri kapsar. Görselleştirme için bu askeri birimlerin gerçek dünya verilerini tam bir biçimde tutması gerekir. Bu nedenle veritabanında ilgili tablo (*Entity*) tasarlanırken daha önce anlatılan askeri sembolojideki alanlar oldukları gibi eklenmiştir.

Özel bir askeri durum Senaryo tablosunda a) *DateTime* tablo sütununda verilen o senaryoya özgü zaman bilgisi, b) *Terrain* sütununda verilen arazi bilgisi tutulur. *Entity* tablosu ise askeri birimlere ait verilerin tutulduğu tablodur. *Scenario* ve *Entity* tablolarının arasındaki bağlantı *ScenarioEntity* tablosunda tutulmaktadır. Şekil 4.31'de tablolar arasındaki ilişkiler verilmiştir.



Şekil 4.31 Veritabanındaki tablolar arasındaki ilişkilerin gösterimi

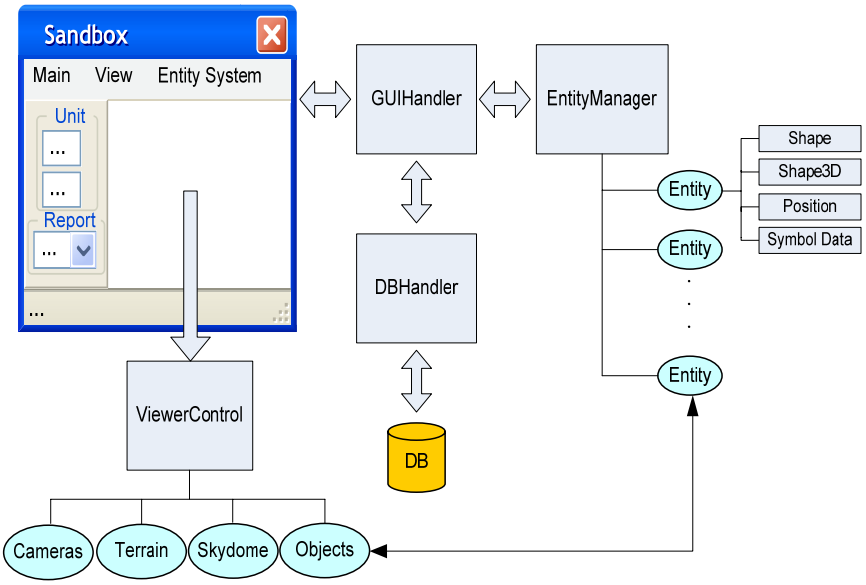
Veritabanına erişim ADO.NET API'si kullanılarak yapılmıştır. Bu teknoloji veritabanına yazılımsal olarak erişip sorgular yapmayı sağlar.

Saklanan askeri birimlerin gösterileceği ayrı bir form tasarlanmış ve kullanıcıların bu form üzerinden çeşitli sorgular yaparak veritabanının içeriğini görmeleri sağlanmıştır. Burada izlenen yöntem veri bağlama (data binding) olarak da bilinir. Bir *DataGridView* nesnesi kullanılarak veritabanından getirilen bilgi tablo halinde gösterilmiştir. Bu nesne veri kaynağını belirten bir *DataSource* nesnesi kullanır. Uygulamada farklı sorgular bir araya getirilmiştir. Veri tabloları ve veri kaynakları arasındaki bağlantılar *TableAdapter* sınıfı üzerinden gerçekleştirilmiştir.

Üç adet *TableAdapter* kullanılarak *Entity*, *Position* ve *ReportDateTime* tablolarına erişim sağlanmıştır. Her uyarlayıcının (adapter) işletecek bir veya birden fazla sorgusu bulunur. Yazılımda kullanılan örnek bir sorgu aşağıda verilmiştir:

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

Verilen SQL tümcesinde, *ReportDateTime* ve *Entity* tablolarından alınan veri sorgu sonucu olarak döndürülür. *ID*, *Entity* tablosunun birincil anahtarıdır ve ReportDateTime tablosu için ikincil anahtardır (foreign key). Bu nedenle verilen *entityID* değişkeni ile karşılaştırılır. "@" sembolü *entityID*'inin bir değişken olduğunu ve değerini, ilgili tablo uyarlayıcısının yordamından çalışma zamanında alacağını göstermektedir.

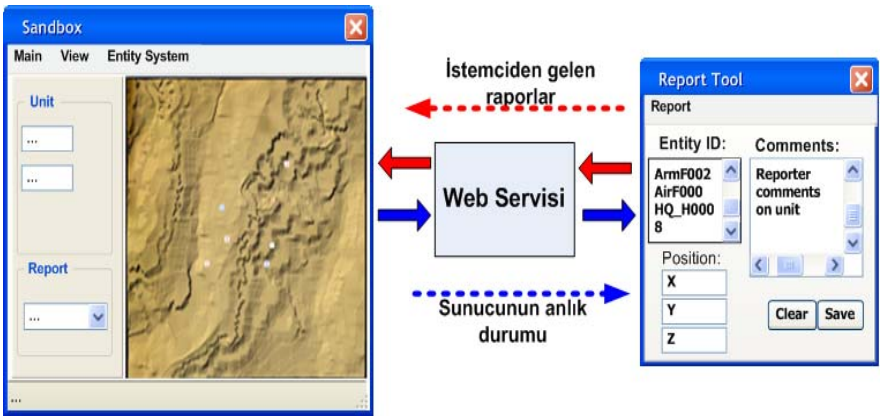


Şekil 4.32 Veritabanı ile birlikte yazılımın kısımlarının gösterimi

Şu ana kadar eklenen kısımlarla birlikte yazılımın genel şeması Şekil 4.32'de verilmiştir. Burada kullanıcı arayüzü ve Birim Yöneticisi arasındaki yardımcı sınıflar (*GUIHandler*) sayesinde metin halinde alınan değerler işlenerek askeri birimlerin bilgileri olarak atanır. Benzer şekilde veritabanına bağlanma için gerekli işlemler de *DBHandler* sınıfı yardımıyla yapılır. Bu yardımcı sınıflar gösterim (representation) ve verinin kendisini (storage) ayırmak için son derece önemli sınıflardır.

4.3.8 Web servislerinin kullanımı

Tez çalışmasında yalnızca bir bilgisayardan girilen bilgilerin değil bunun yanında uzak gözlemleyicilerden gelen raporların da sistemi etkilemesi sağlanmıştır. Web servisleri kullanılarak biri sunucu diğeri istemci olmak üzere iki taraf arasında iletişim kurulmuştur. Bu mimari Şekil 4.33'deki gibi gösterilebilir.



Şekil 4.33 Web servisi aracığıyla iki uygulamanın haberleşmesi

Uygulamada web servisi ana uygulamanın çalıştığı makinede bulunmaktadır. Aradaki iletişim *CommunicateService* isimli sınıf aracılığıyla sağlanır. Bu servis iki yönlü bir iletişim sağlar. İstemciden gelen veriler sayesinde gözlemleyicilerden gelen saha bilgisi ana uygulamaya gönderilirken, sunucu sürekli olarak istemcilere ana uygulamadaki son durumu hem nesne tanımları hem de gösterilen savaş alanının bir görüntüsü halinde gönderir.

Askeri birim bilgisinin bu servis aracılığıyla gönderilmesi için kullanıcı tanımlı *EntityData* ve *Report* sınıfları oluşturulmuştur. İlk sınıf nesne tanımı, konum bilgisi, taraf ve gözlemleyici yorumları gibi bilgiler içerirken, ikinci sınıf birçok farklı birimin tek bir raporda gönderilmesi için birimlerin tutulduğu bir nesne tutar. Bu listeye ek olarak raporun gönderilme zamanı da rapora eklenir.

İki yönlü iletişim için gerekli birçok yordam, web servisi tarafından sağlanmıştır. Şekil 4.34'de raporların XML biçiminde gönderilmesi örneklendirilmiştir. *ReportList* etiketi, farklı askeri birim bilgisi için ana bir düğüm olarak görev alır. Her birimin konumu, ait

olduğu taraf ve o birime ait yorumlar saklanmaktadır. Listeden sonra ise rapora ait zaman bilgisi eklenmiştir.

```
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>
```

Şekil 4.34 Gönderilen rapora ait SOAP cevabı

Ana uygulama ise sıklıkla (uygulamada her 10 saniyede bir) hâlihazırdaki sistem durumunu gözlemleyicilere gönderir. Resim olarak gönderilen bu durum bilgisi, rendering işleminin varsayılan rendering hedefi olan kullanıcı ekranı değil de bir resim olarak tanımlanan bir rendering hedefi üzerinde yapılması ile elde edilir. Bu sayede gösterimdeki anlık bir görüntü resim üzerine alınmış olur. Bu resim de web servisi aracılığıyla istemcilere gönderilir.

Web servisinin de eklenmesiyle Sandbox yazılımının uzak bilgisayarlardan aldığı verileri de işleyerek en güncel durumu göstermesi sağlanmıştır.

5. TEST ve UYGULAMA SONUCU

Bu bölümde geliştirilen yazılımların uygulanması ve elde edilen sonuçlar incelenecektir.

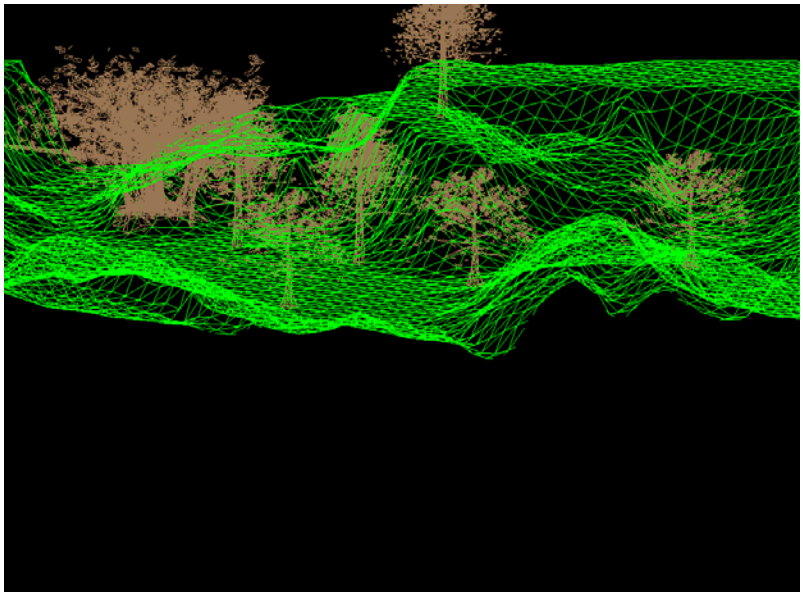
5.1 TerraVis

Okunan bir yükselti verisinden elde edilen araziyi gösteren bu yazılım geliştirilirken daha önce de anlatıldığı gibi nokta sayısı eniyileme işlemi sonucunda azaltılmıştır. Bu işlem sonucunda elde edilen nokta sayıları ve bu noktalar kullanılarak çizilen üçgen sayıları aşağıdaki Çizelge 5.6'de gösterilmiştir. Çizelge işlemin farklı model türleri için sonuçlarını göstermektedir.

Çizelge 5.6 Eniyileme işleminden sonra nokta ve yüzey sayıları

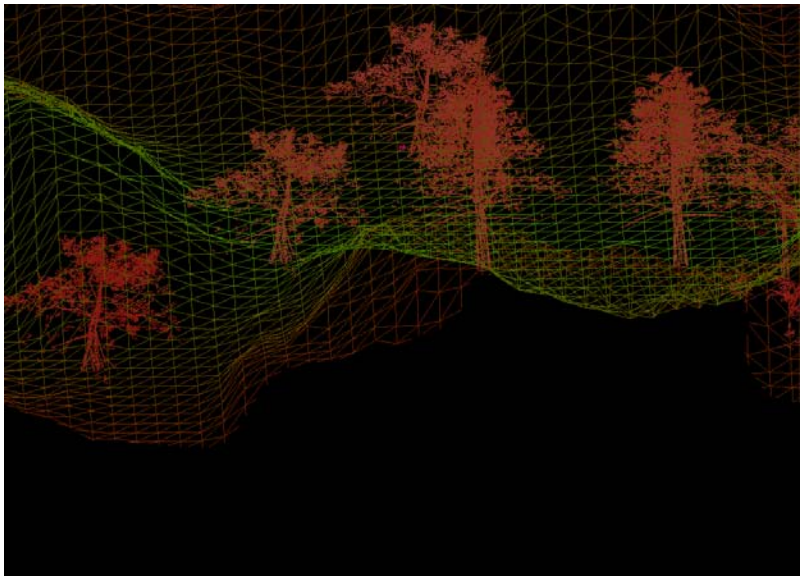
Model: Arazi1	**Değiştirici Uygulanmadan Önce**	**Değiştirici Uygulandıktan Sonra**	**İyileştirme Oranı (%)**
Nokta Sayısı	5776	3038	47.4
Yüzey Sayısı	11250	6063	46.1
Model: Arazi2			
Nokta Sayısı	10201	6850	32.8
Yüzey Sayısı	20000	13426	32.9
Model: Ağaç			
Nokta Sayısı	21385	3713	82.6
Yüzey Sayısı	21945	1476	93.3

Geliştirilen yazılım bir arazi modelini ve üzerine çizilecek ağaç modellerini okur ve Şekil 5.35'deki gibi ekranda gösterir. Yapılan çizim farklı yaklaşım oranlarında veya farklı yönlerde görüntülenebilir. Yine çizim yapılırken sadece noktalar, çizgiler veya poligonlar gösterilebilir.



Şekil 5.35 Yazılımın tam ekran görünümü

Farklı etkiler oluşturmak ve kırpmanın etkisini azaltmak için kullanılan sis etkinleştirildiğinde Şekil 5.36'daki görüntü elde edilir.

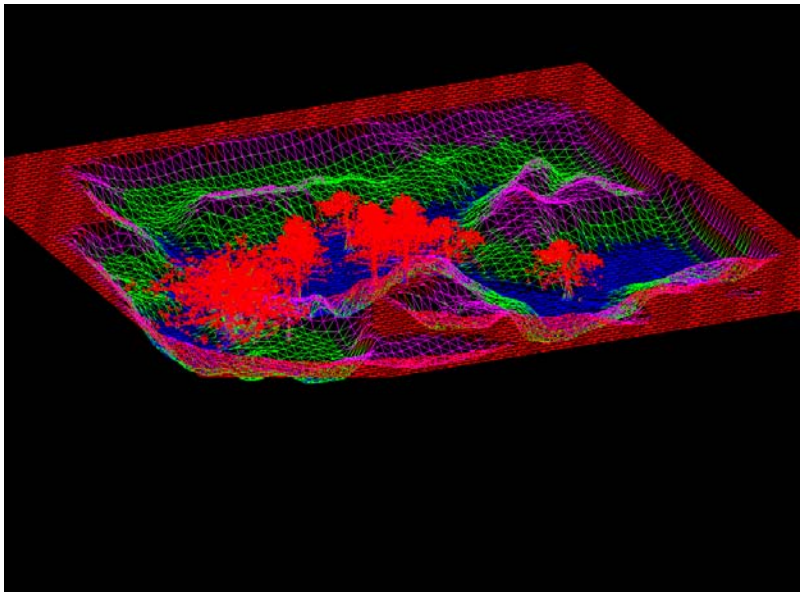


Şekil 5.36 Sisin etkinleştirilmesi

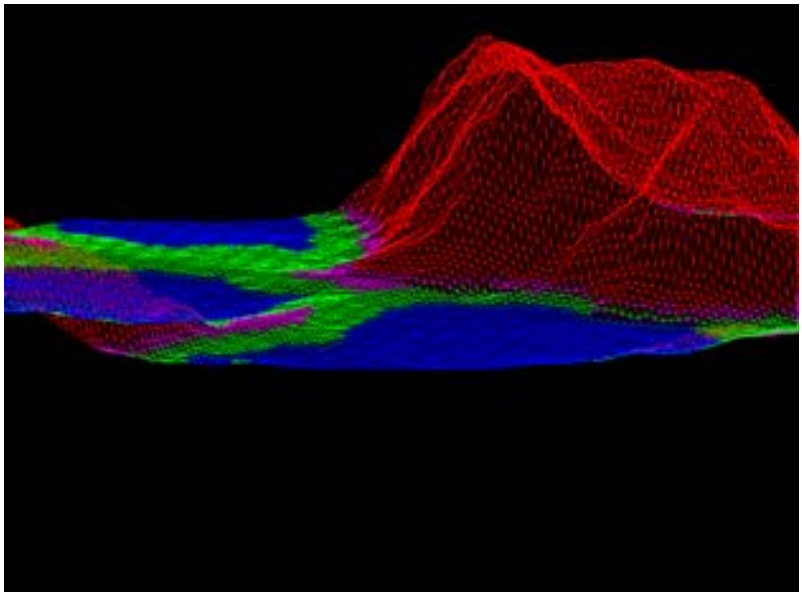
Sisin ardından hareketli ışık da etkinleştirildiğinde Şekil 5.37'deki gibi bir çıktı oluşur. Burada hareketli ışık klavyeden kontrol edilerek yer değiştirebilmektedir.

Şekil 5.37 Hareketli ışığın etkinleştirilmesi

Arazi üzerindeki farklı yükseltilerin farklı renklerle gösterilmesi etkinleştirildiğinde Şekil 5.38 ve 5.39'daki görüntüler oluşur. Böylece arazi üzerindeki yükselti farkları daha belirgin bir biçimde ortaya konmuş olur. Şekil 5.39'da farklı bir arazi modeli kullanılmıştır.



Şekil 5.38 Yükselti renklendirmenin etkinleştirilmesi

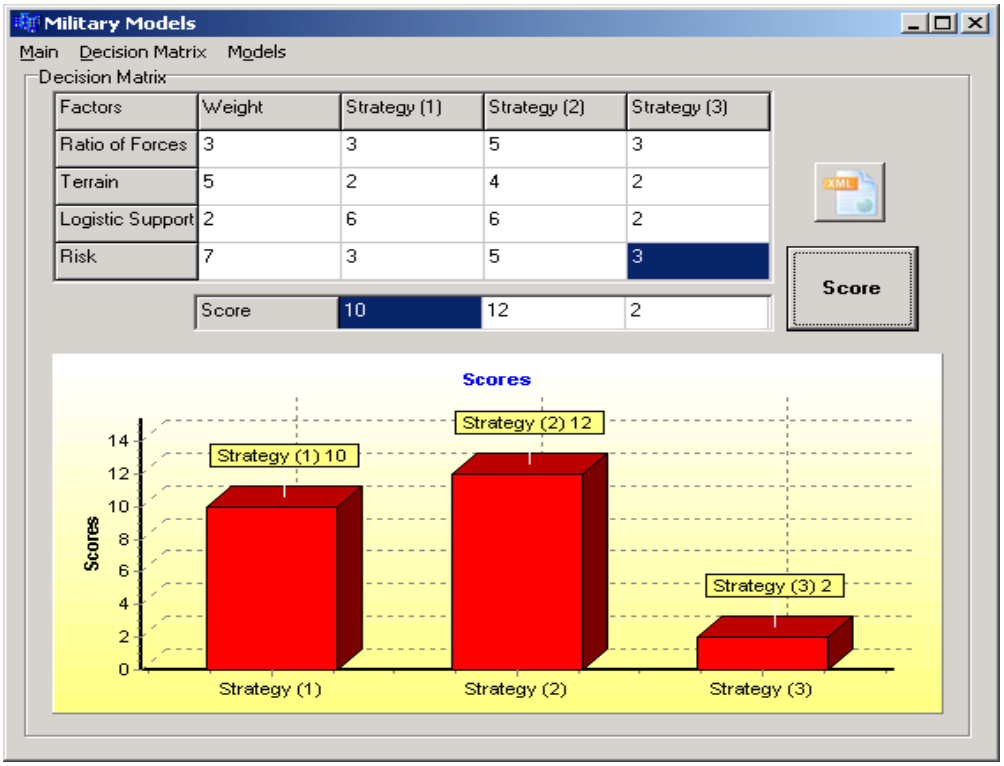


Şekil 5.39 Başka bir arazi modelinin gösterimi

5.2 MDT

Geliştirilen MDT yazılımı ile basit bir karar verme aracı prototipi oluşturulmuştur. Kullanılan modellerden ilki olan karar matrisi açıldığında Şekil 5.40'taki ekran kullanıcının karşısına çıkar. Bu ekran üzerinde kullanıcılar belirli bir durumda farklı stratejilerin farklı etkenlerle birlikte önemlerini belirlerler ve sonunda en uygun strateji seçilir.

Elde edilen sonuçlar aşağıda grafik olarak verilir. Bu kısmın önemli bir özelliği de XML verilerini okuyarak bunlar ile işlem yapabilmesidir. Böylece, uzak kullanıcılardan gelen verilerin de değerlendirilmesi sağlanacaktır.

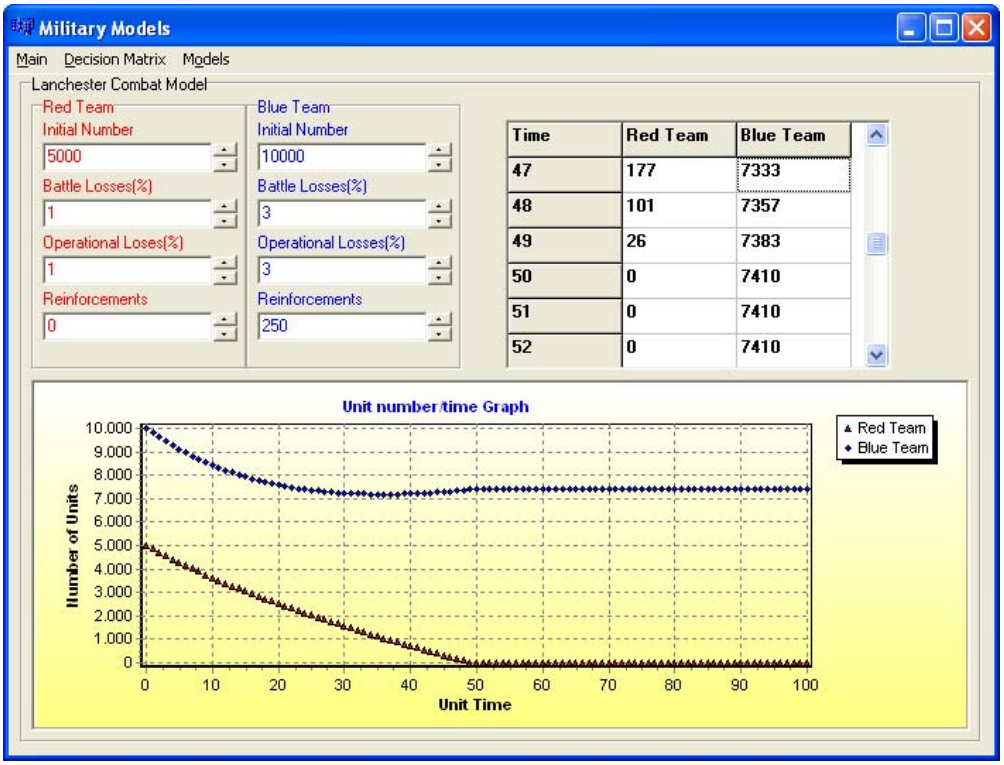


Şekil 5.40 Karar matrisinin gösterimi

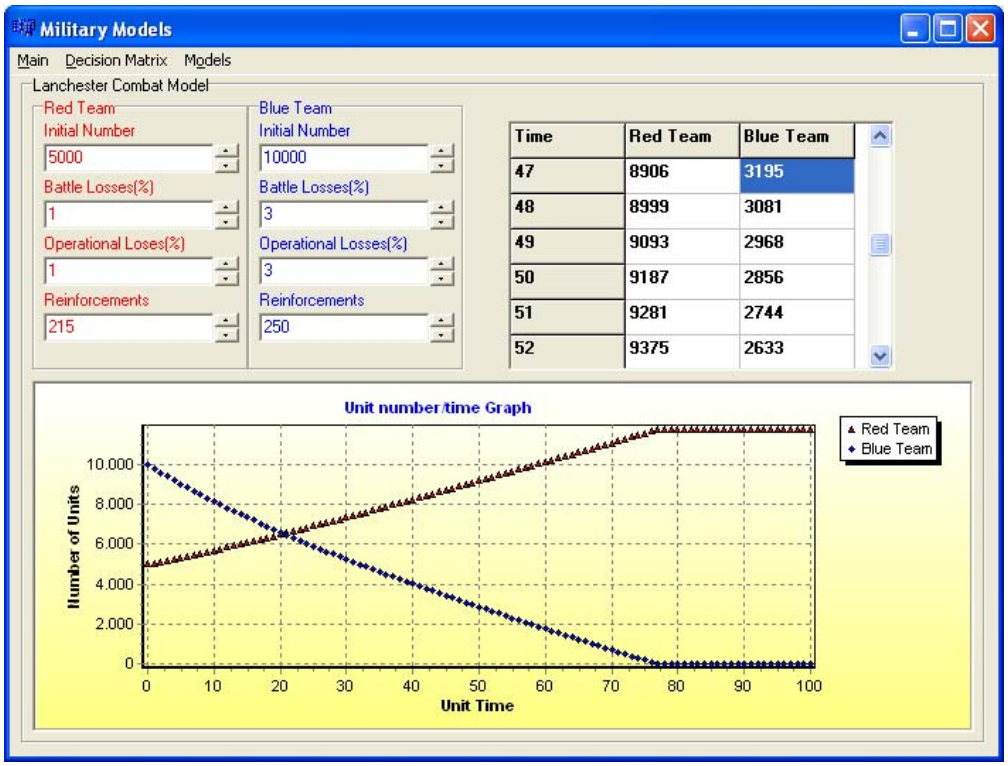
Lanchester Savaş Modellerinin de kullanılması ile belirli bir durumda iki taraf için verilen başlangıç değerleri ve savaşın durumuna göre verilen harekât kayıpları, harekât dışı kayıplar ve destek kuvvetler gibi parametrelerin savaşın sonucunu nasıl etkilediğinin incelenmesi sağlanmıştır.

Şekil 5.41'de mavi takımın başlangıçta sayıca üstünlüğü olmasına rağmen harekât ve harekât dışı kayıpları kırmızı takımdan fazladır. Buna karşılık, kırmızı takım hiç destek kuvvet alamamaktadır. Sonuç olarak kırmızı takımda hiç birim kalmamaktadır.

Şekil 5.42'de ise kırmızı takımın destek kuvvet alması gösterilmiştir. Görüldüğü gibi kırmızı takımın destek kuvvet sayısı diğer takıma göre az olsa da savaşın sonucu tamamen değişmektedir.



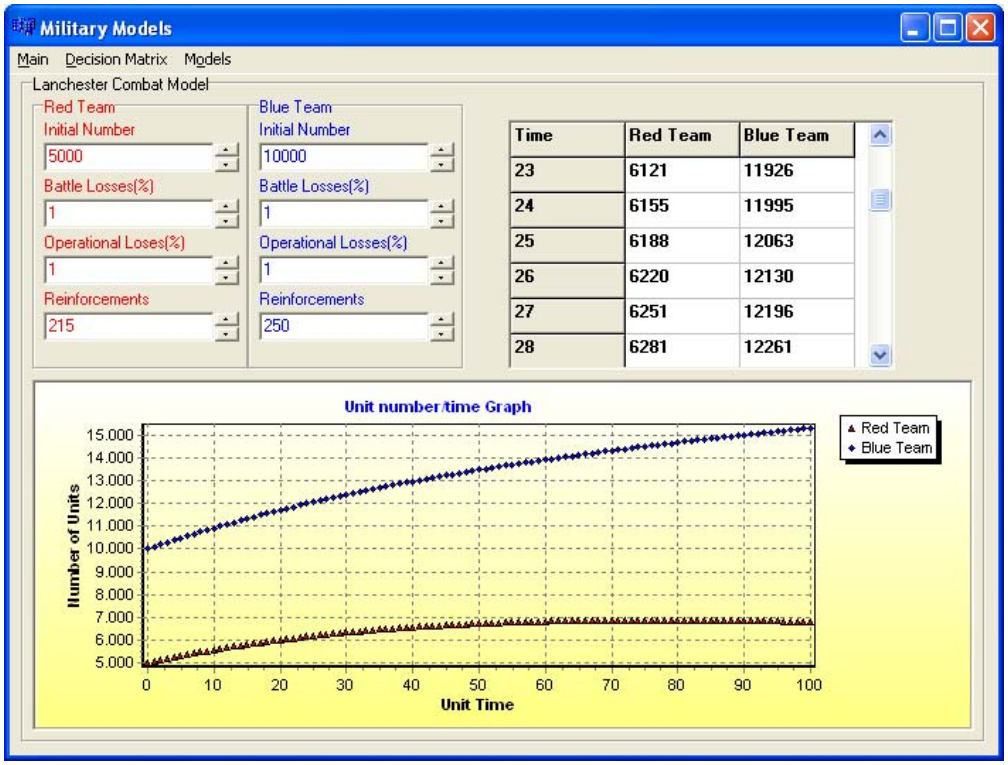
Şekil 5.41 Mavi takım için destek ekiplerin sonucu etkilemesi



Şekil 5.42 Kırmızı takıma destek kuvvet verildiğinde savaşın sonucu

Mavi takımın harekât ve harekât dışı kayıplarını azaltması durumu Şekil 5.43'te verilmiştir. Buradan mavi takımın silah teknolojisinde bir iyileştirme yapıldığı sonucu

çıkarılabilir. Sonuç olarak savaşın ilk 100 birim zamanı içinde mavi takımın galip geldiği ve sonrasında da kırmızı takımın tamamen yok edileceği görülmektedir.



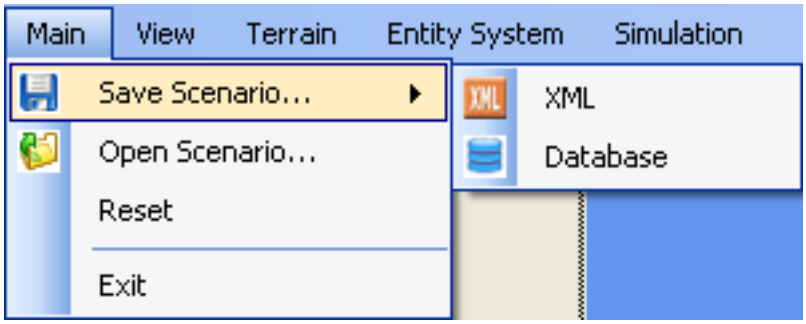
Şekil 5.43 Mavi takımın savaş dışı kayıpları azaltıldığında alınan sonuç

5.3 Sandbox

Tez konusunun temeli olan Sandbox yazılımı ilk olarak açıldığında yalnızca üst kısımdaki menüler ve boş bir ekran görünür. Sol taraftaki panel kullanıcının seçimine göre gerek yeni veri girme işlemi, gerekse güncelleme işlemlerinde kullanılmak üzere çeşitli bileşenlerle doldurulacaktır.

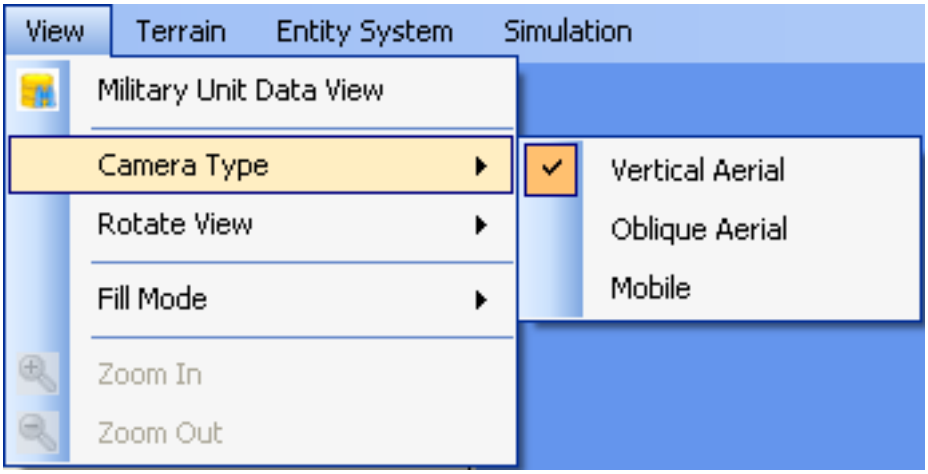
Ana menüde *Main*, *View*, *Terrain* ve *Entity System* seçenekleri bulunmaktadır. Bu seçeneklerin tümünün bir yerde toplanması kullanıcıların sistematik bir biçimde yazılımın farklı özelliklerini kullanabilmelerini sağlamıştır. Ayrıca menü elemanları kullanım sıklıklarına göre düzenlenerek yazılımın kullanımı kolaylaştırılmıştır.

İlk menü elemanı olan *Main* menüsünün içinde yazılımı yeniden başlatmak veya kapatmak gibi temel işlemlerin yanında Şekil 5.44'de gösterildiği gibi bir senaryoyu kaydetmek veya kayıtlı bir senaryoyu açmak gibi işlemler sunulur.



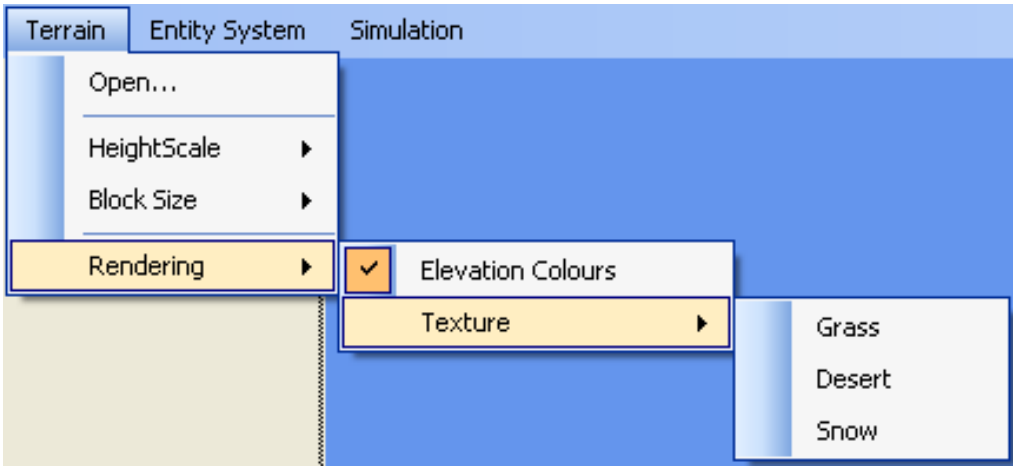
Şekil 5. 44 Main menü elemanı

View menüsü grafik alanın farklı açılardan ve bakış noktalarından gösterilmesini sağlar. Kullanıcı çeşitli menü seçenekleri ile savaş alanını farklı yönlere çevirebilir ve yakınlaşıp uzaklaşabilir. Veritabanı bu kısımdan erişilebilir. Bu menünün alt elemanları Şekil 5.45'te gösterilmiştir.



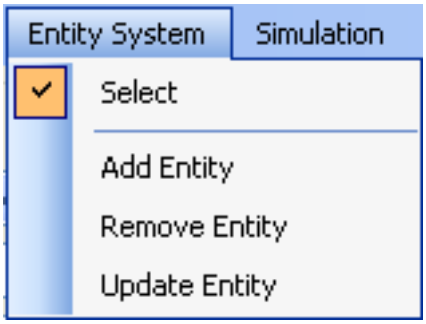
Şekil 5.45 View menü elemanı

Terrain seçeneği arazi ile ilgili işlemlerin yapılmasını sağlar. Bu eleman sayesinde kullanıcılar farklı bölgelere ait DEM biçimindeki yükselti verilerini açarak savaş alanı olarak kullanabilirler. Ayrıca arazi noktaları arasındaki uzaklıkları ve yükselti değerlerini değiştirek araziyi farklı biçimlerde görüntüleyebilirler. *Rendering* kısmında ise araziyi yükselti renklendirme veya farklı dokularla görüntüleme seçenekleri sunulmuştur.



Şekil 5.46 Terrain menü elemanı

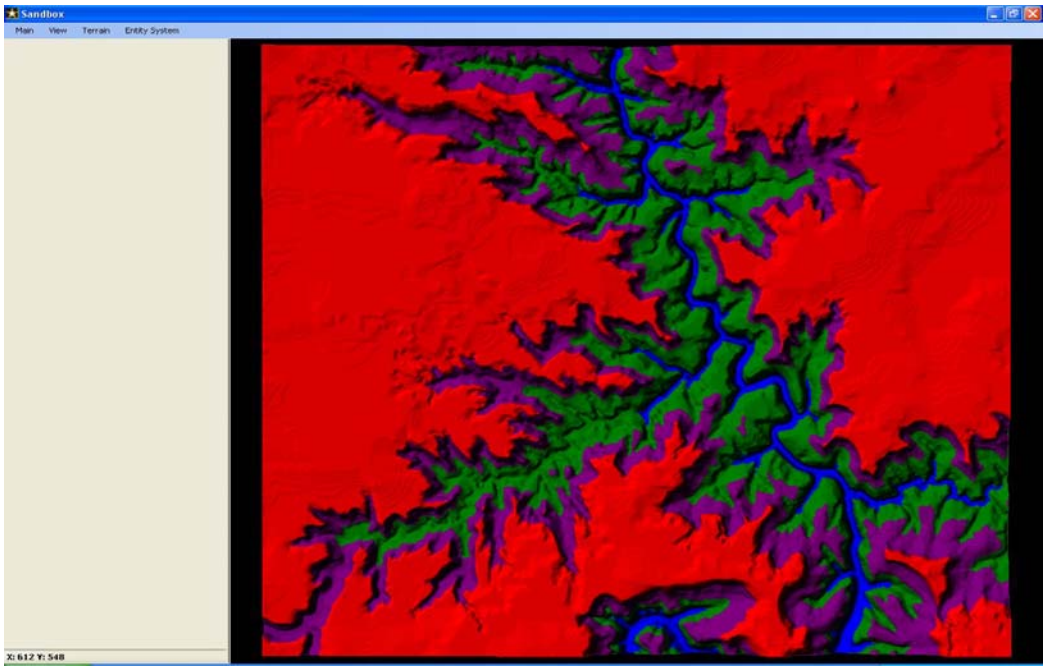
Askeri birimlerle ilgili işlemler *Entity System* menüsü kullanılarak yapılır. Şekil 5.47'de gösterildiği gibi burada herhangi bir birimi seçme, yeni birim ekleme, güncelleme veya silme gibi işlemler yapılabilir.



Şekil 5.47 Entity System menü elemanı

Son olarak *Simulation* menü elemanı kullanılarak Lanchester Savaş Modellerinin gösterildiği form açılır.

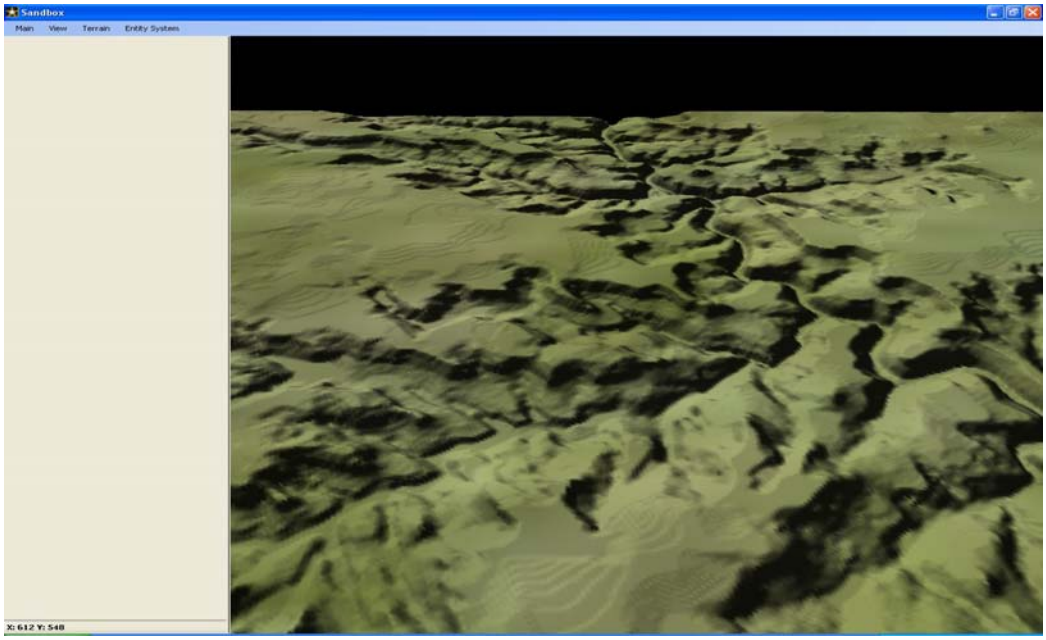
Şekil 5.48'de arazi ilk olarak açıldığında kullanıcıya karşısına gelecek olan görüntü verilmiştir.



Şekil 5.48 Arazi yükselti verisi yüklendiğinde programın görünümü

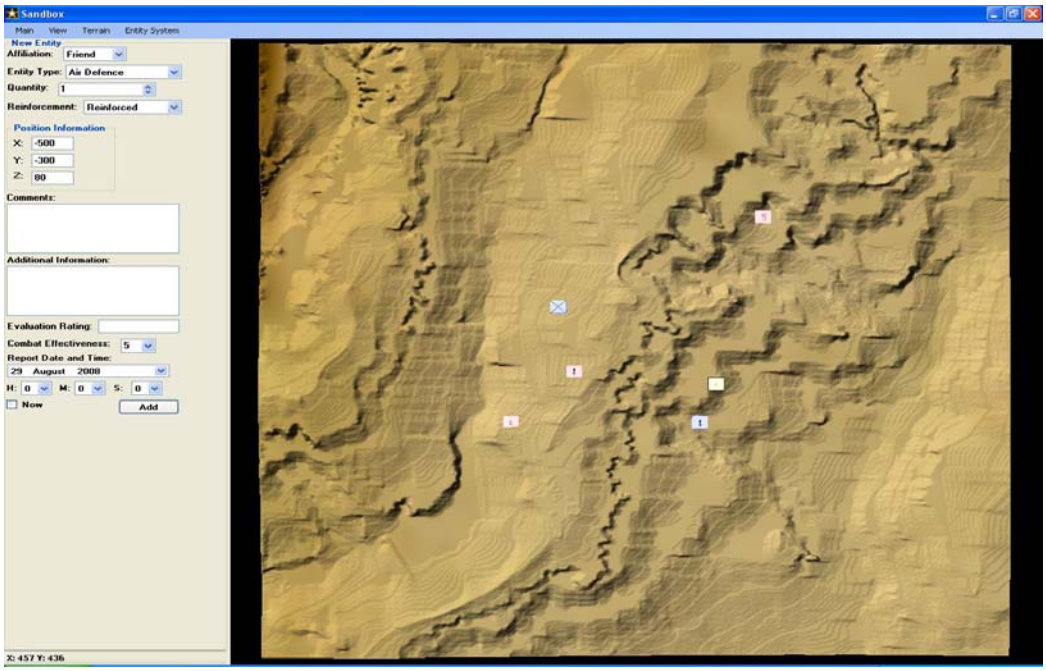
Arazideki farklı yükselti aralıklarının değişik renklerle gösterilmesi kullanıcıların yükselti değişimlerini daha açık görmelerini sağlayacaktır. Sağlanan dikey görünüm ise tüm savaş alanının bir bütün halinde gösterilmesini sağlayarak en genel durumun anlaşılmasında yardımcı olacaktır. Dikey görünüme ek olarak sunulan ortografik görünüm ise eğik bir açıdan görüntülenmesini sağlar. Bu görünümde tüm alan görüntülenememesine rağmen birimler arazi yükseltisiyle birlikte daha rahat görünür.

Terrain menüsünün altında ayrıca arazinin dokusunun seçilebilmesi de sağlanmıştır. Böylece kullanıcılar durumu en iyi ifade edebilecek biçimde çimen, çöl ve kar dokularından birini seçebilir. Araziye farklı dokuların uygulanması Şekil 5.49, 5.50 ve 5.53'te gösterilmiştir.



Şekil 5.49 Ortografik görünüm ve çimen dokusunun uygulanması

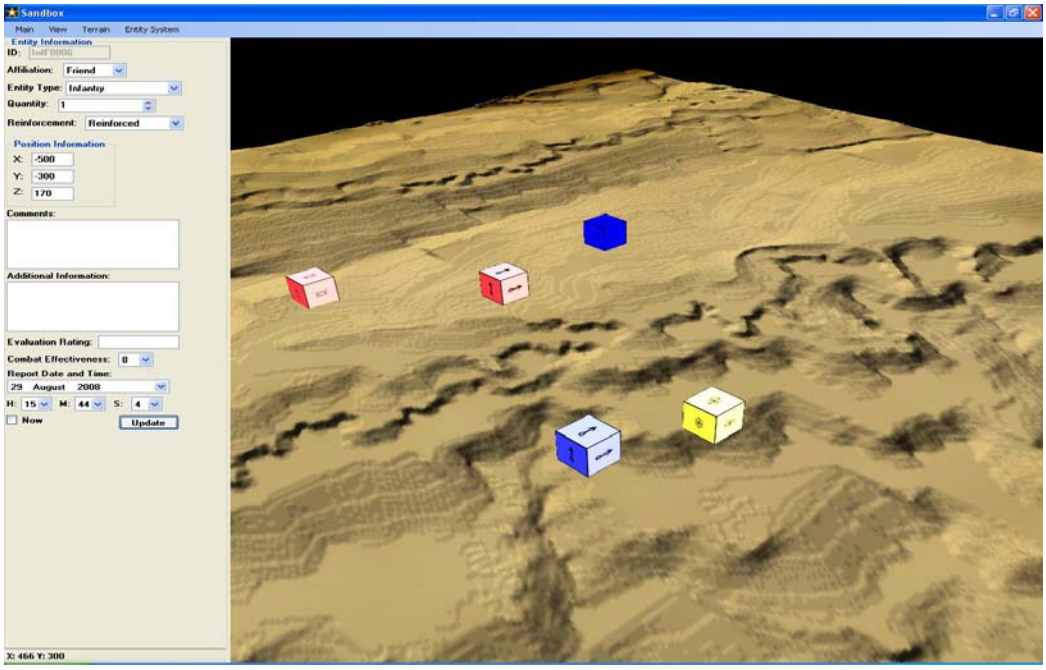
Entity System menüsü aracılığıyla yazılımdaki askeri birimlerin kontrol edilmesi sağlanmıştır. Bu kısımda yazılıma farklı birimler eklenebilir, istenen bir birim güncelleme için seçilebilir veya sistemden silinebilir. Farklı taraflardan birimlerin eklenmesi Şekil 5.50'de gösterilmiştir.



Şekil 5.50 Savaş alanına askeri birimlerin eklenmesi

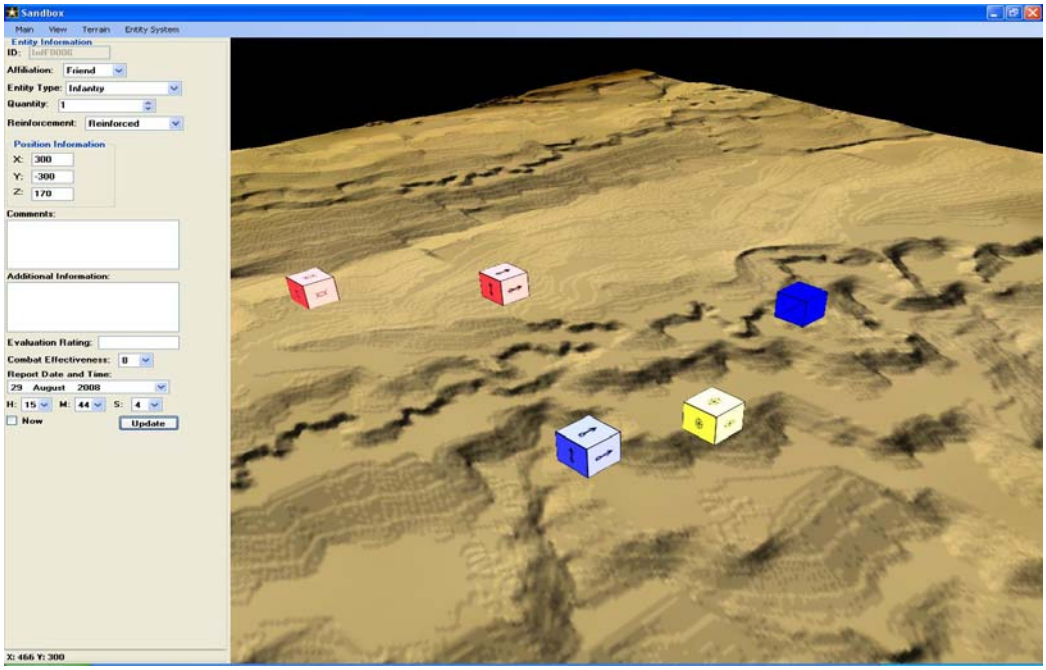
Bir askeri birim seçildiğinde rengi diğerlerinden rahatlıkla ayrılabilecek bir biçimde belirginleştirilir (Şekil 5.51). Aynı birim seçiliyken bir kere daha tıklandığında birim

normal rengine döner. Herhangi bir askeri birim seçildiğinde sol taraftaki panelde o birime ait bilgiler görünür. Bu panel aracılığıyla birime ait bilgiler değiştirilebilir.

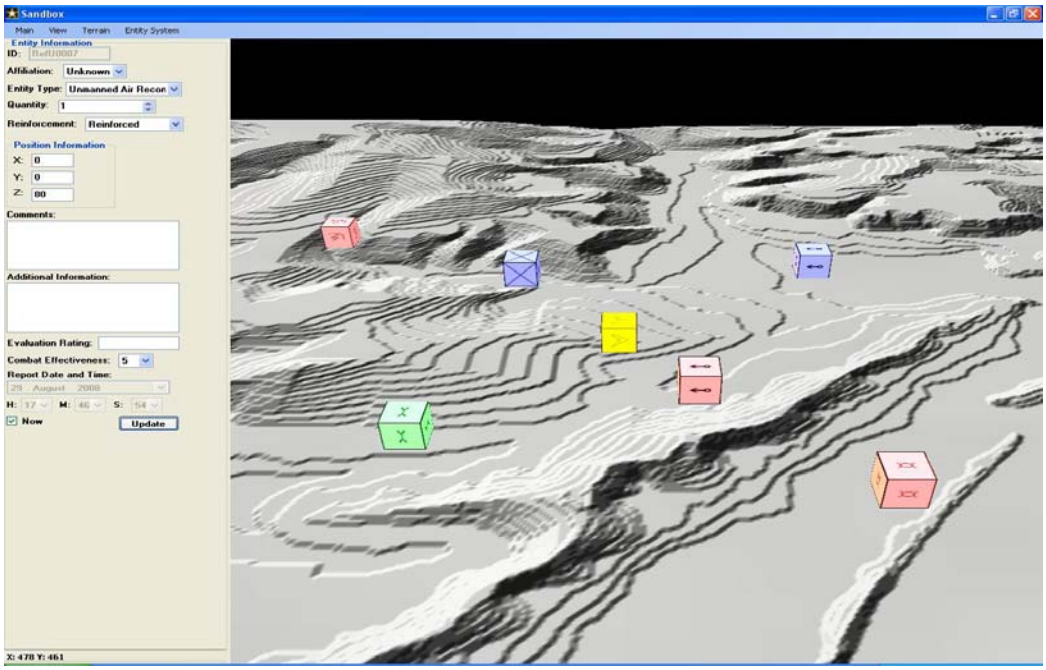


Şekil 5.51 Savaş alanının farklı açılardan görünümü ve bir birimin seçilmesi

Seçili birimde konum bilgisi güncellenmek istendiğinde seçili birim yeni konumuna doğru yavaş bir biçimde hareket ederek gider.

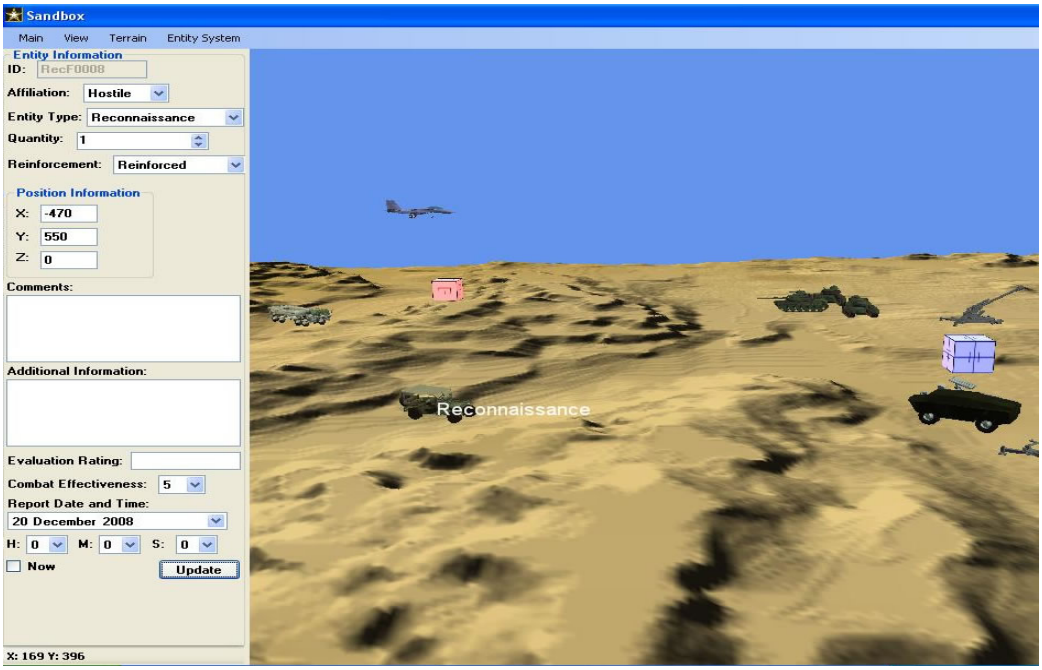


Şekil 5.52 Seçilen birim için konum güncellemesi yapılması



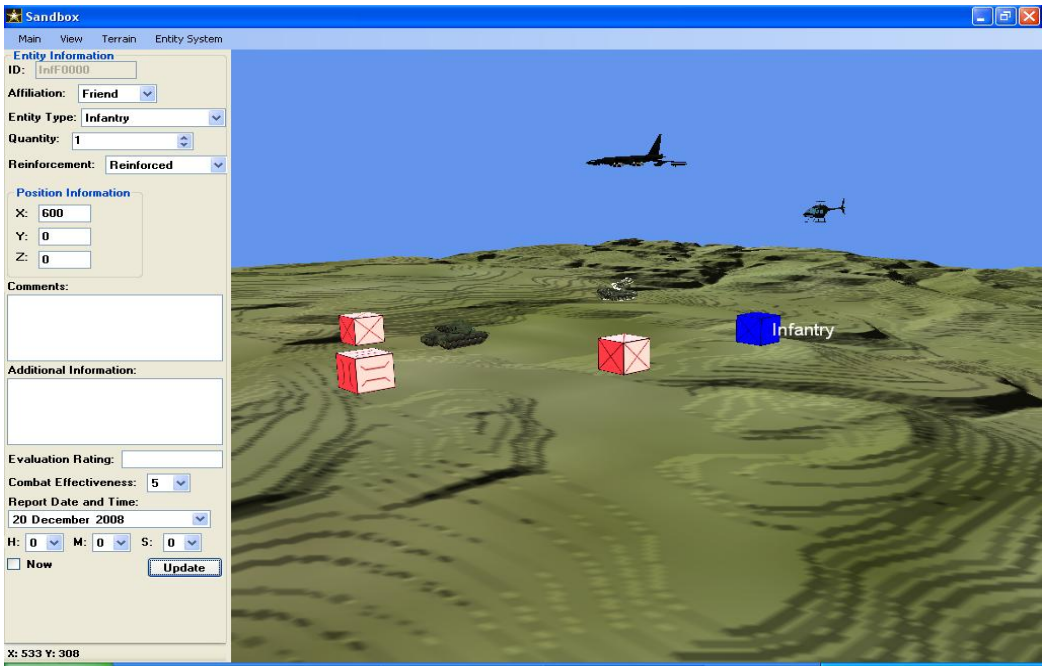
Şekil 5.53 Farklı birimlerin eklenip çıkarılması

Birimlerin semboller halinde gösterilmesine ek olarak bazı birimlere ait 3B modellerin de yazılıma eklenmesi ile daha gerçekçi bir gösterim elde edilmiştir. Kullanıcılar birimleri ister 3B model olarak isterlerse de sembol olarak görebilirler. Bu kullanım sayesinde sembolojinin de öğrenimi kolaylaşacaktır. Şekil 5.54 ve Şekil 5.55 bu kullanıma örnek olarak verilmiştir.



Şekil 5.54 3B modellerin semboloji ile birlikte gösterilmesi

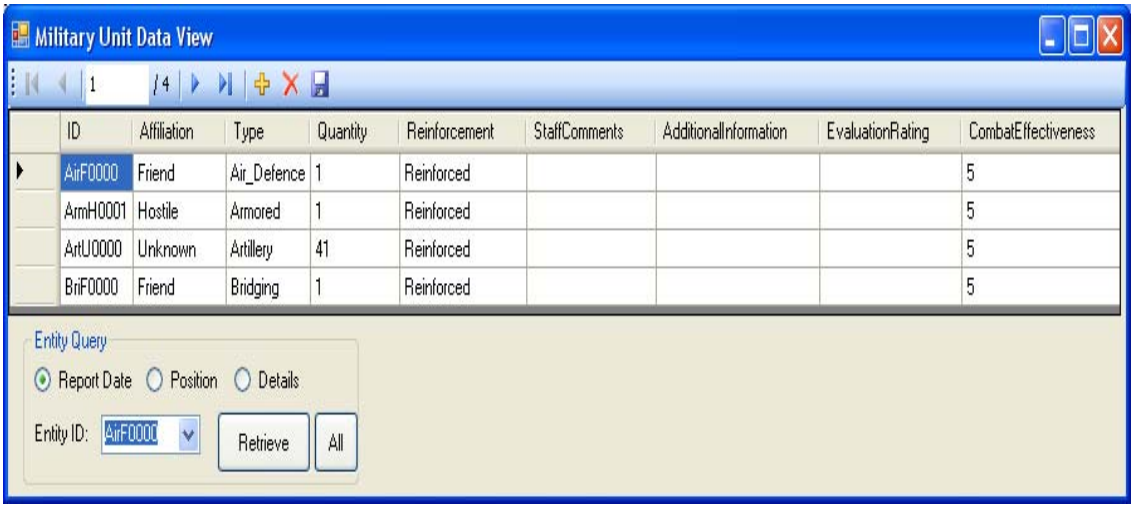
Şekil 5.54'da kuzeydoğu yönünde dost bir tank taburu konuşlandırılmışken, düşman uçağı üzerlerine doğru gelmektedir. Doğuda uzun menzilli toplar bulunmakta ve batıda da bir muhabere jeepi savaş alanı hakkında bilgi toplamaktadır.



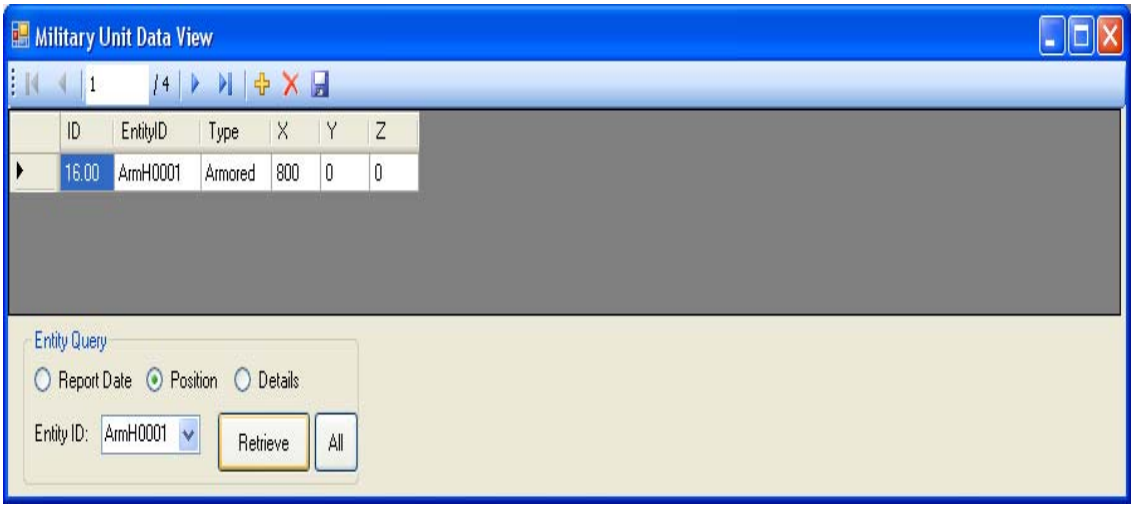
Şekil 5.55 Farklı bir senaryoda 3B modellerin semboloji ile birlikte gösterilmesi

Şekil 5.55'te ise farklı bir senaryo verilmiştir. Bir hava savunma birimi destek uçağını hedef almıştır ve bir helikopter de bu uçağı korumak için destek ateşi sağlamaktadır. Piyadeler semboller halinde gösterilmiştir (üzerlerinde çarpı işareti olanlar). Bir piyade sembolünün üzerindeki yazı seçili elemanın türünü göstermektedir.

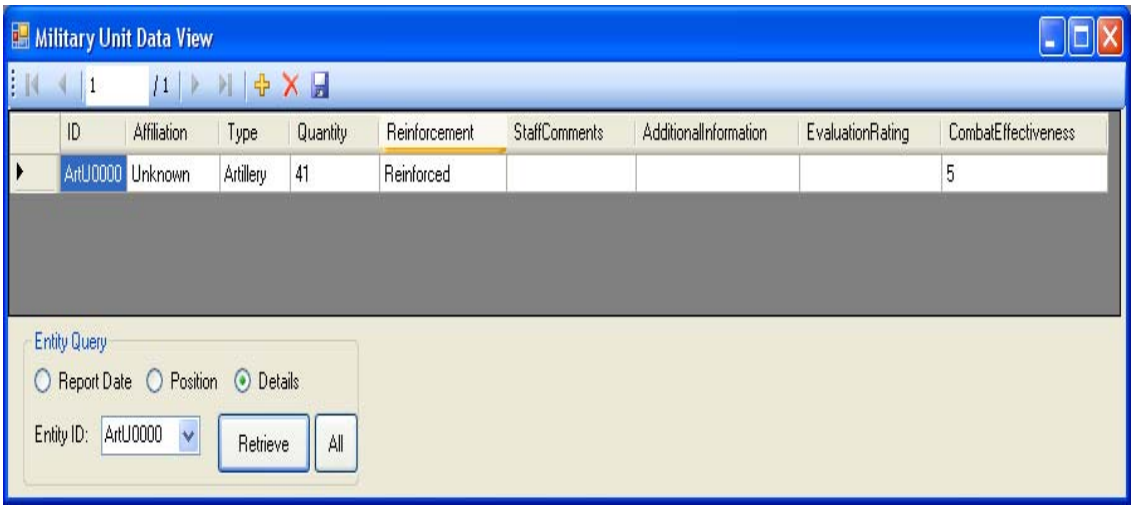
Yazılım içinde veritabanına erişim Şekil 5.56, 5.57 ve 5.58'de verilen form ile sağlanmaktadır. Bu form üzerinden farklı sorgular çalıştırılabilir. Örneğin seçilen bir birime ait konum bilgisi, rapor zamanı ve ayrıntılı bir biçimde veri getirmek için sorgular düzenlenebilir.



Şekil 5.56 Kayıtlı tüm birimlerin gösterilmesi

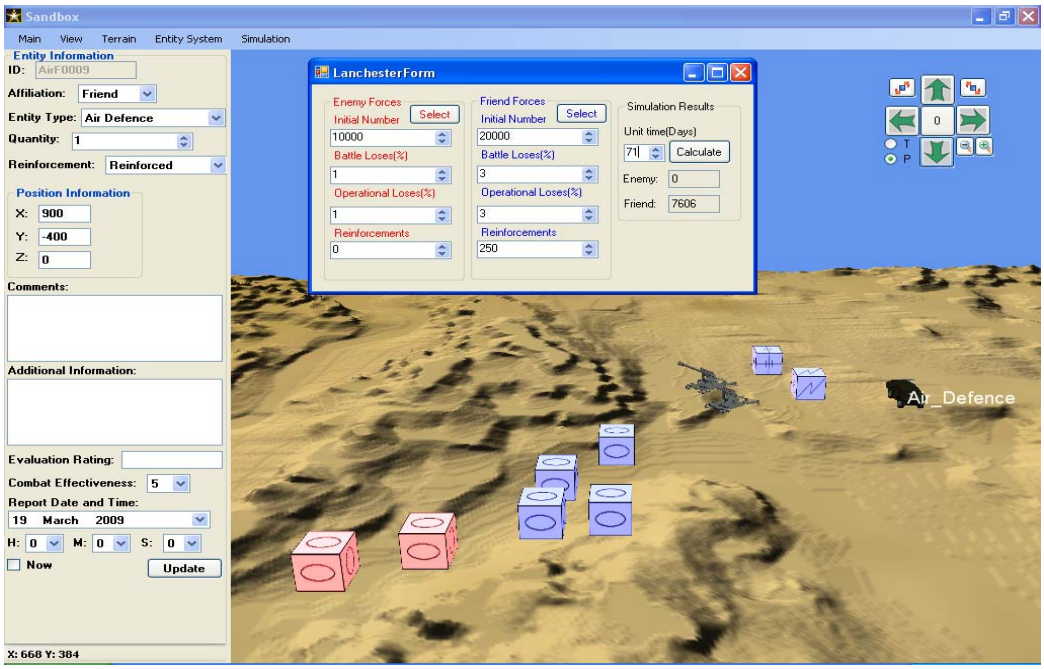


Şekil 5.57 Tek bir birime ait konum sorgusu



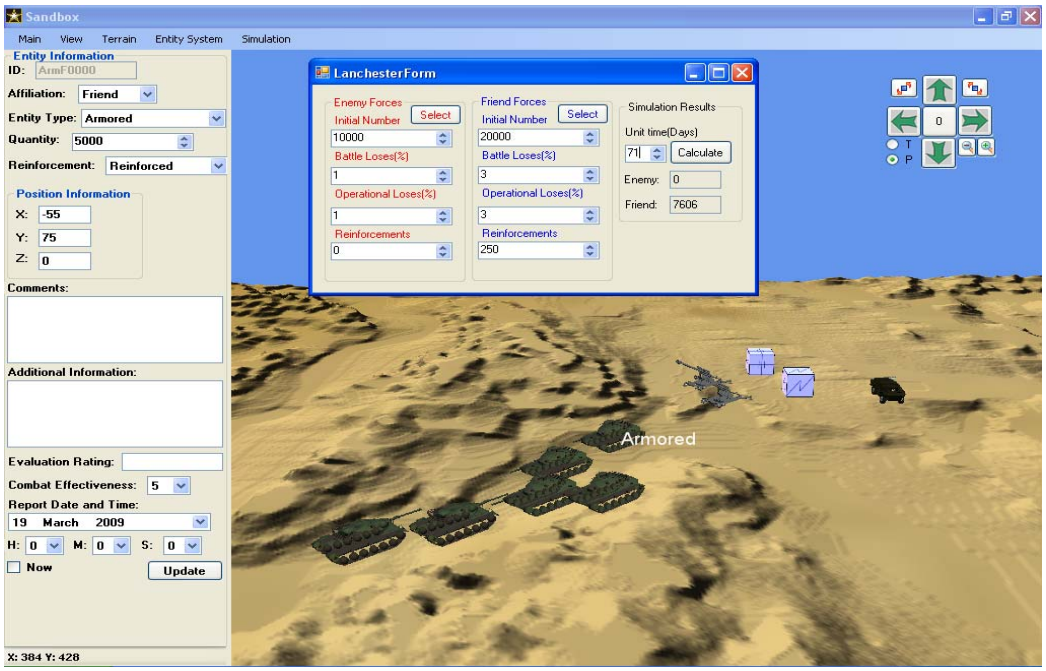
Şekil 5.58 Tek bir birime ait detaylı sorgu

Daha önce ayrıntılı bir biçimde anlatılan Lanchester Savaş Modelleri de yazılımla bütünleşik bir hale getirilmiştir. Şekil 5.59'da verilen örnekte semboloji ile gösterilen iki düşman tankı müttefik bir kampa girmeye çalışmaktadır. Her bir sembolün 5000 tanka karşılık geldiği düşünüldüğünde buna karşılık gelen veriler gösterilen formda verilmişlerdir. Savaşın gidiş durumuna göre komutanlar çeşitli parametreleri ayarlayarak sonucun nasıl olacağını kestirmeye çalışırlar.



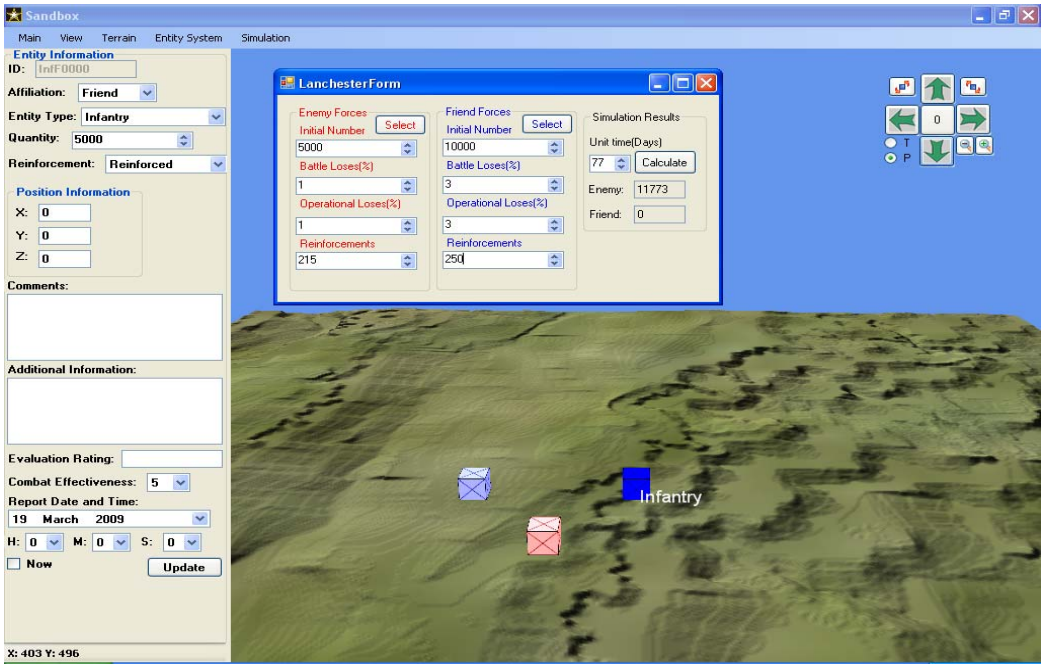
Şekil 5.59 Semboloji birlikte Lanchester modellerinin kullanımı

Aynı senaryonun 3B modellerle gösterilmesi de Şekil 5.60'ta verilmiştir.



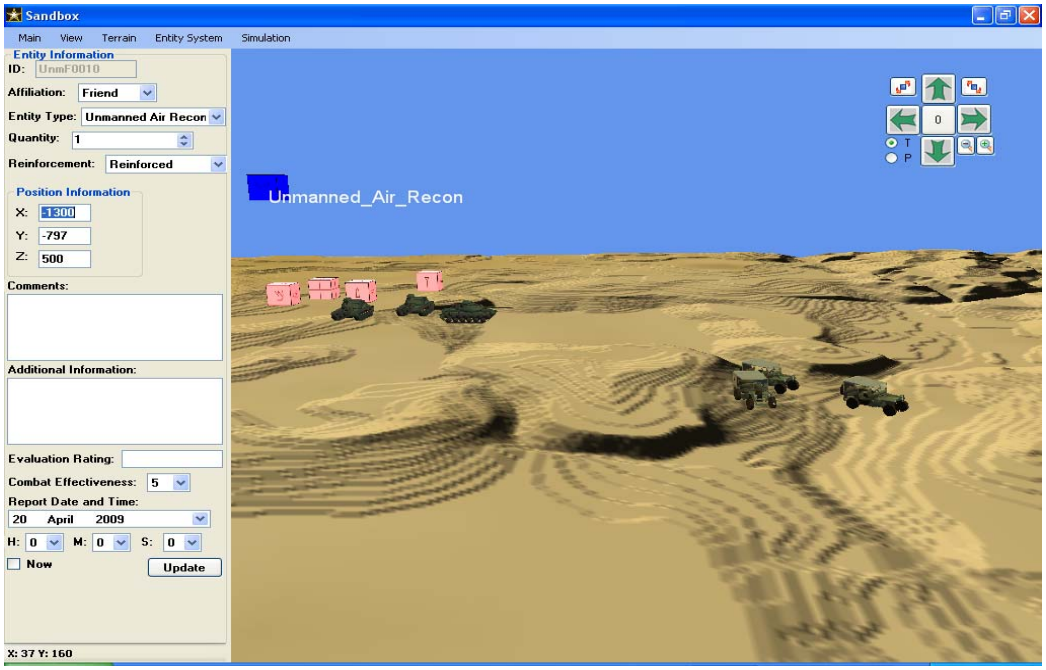
Şekil 5.60 3B görünümlerle birlikte Lanchester modellerinin kullanımı

İkinci bir senaryo da Şekil 5.61'de verilmiştir. Burada düşman piyade bölüğü iki dost piyade bölüğü tarafından sarılmıştır. Buna rağmen sürekli destek alarak 77 gün sonra savaşı kazanması simülasyon sonucu olarak gösterilmektedir.



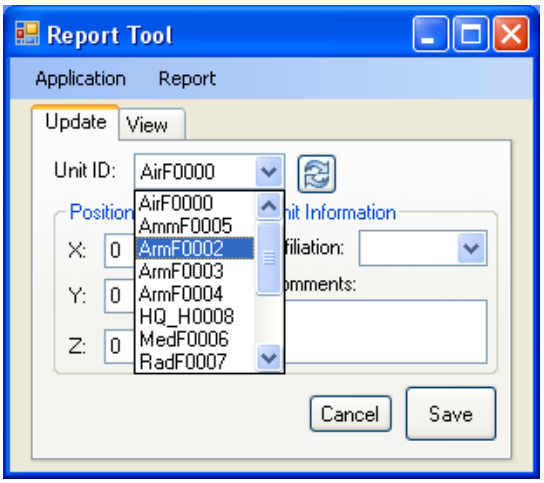
Şekil 5.61 Bir diğer senaryo için simülasyon sonuçları

Yazılımın en önemli kısımlarından biri de şüphesiz web servislerini kullanarak uzak gözlemleyicilerden gelen raporlar ile sistemin güncellenmesidir. Bu kullanıma örnek olması için bir başlangıç senaryosu Şekil 5.62'de verilmiştir. Burada üç adet muhabere jeep'i savaş alanı hakkında bilgi toplarken düşman hattına fazla yaklaşmışlardır.

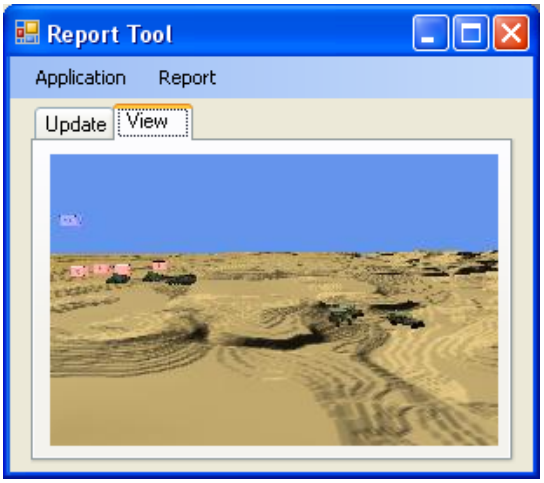


Şekil 5.62 Raporlama aracının kullanımı için örnek bir başlangıç senaryosu

Raporların oluşturulması için hazırlanan raporlama aracı Şekil 5.63 ve 5.64'te gösterilmektedir. Bu araç sürekli olarak güncellenmektedir. Aracın iki farklı görünümü bulunmaktadır: rapora ait verilerin olduğu görünüm ve Sandbox yazılımındaki görünüm.

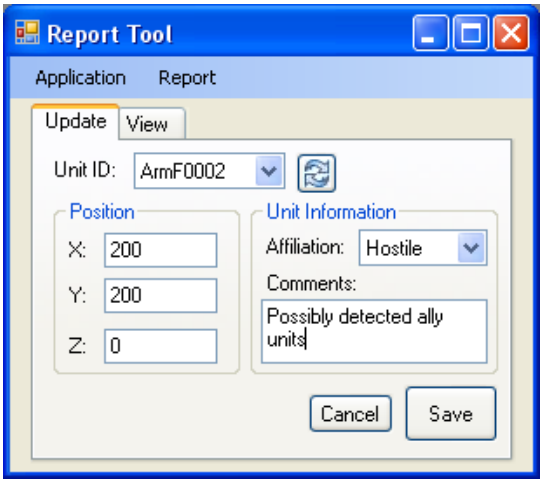


Şekil 5.63 Sandbox'ta eklenen birimleri raporlama aracında gösterilmesi



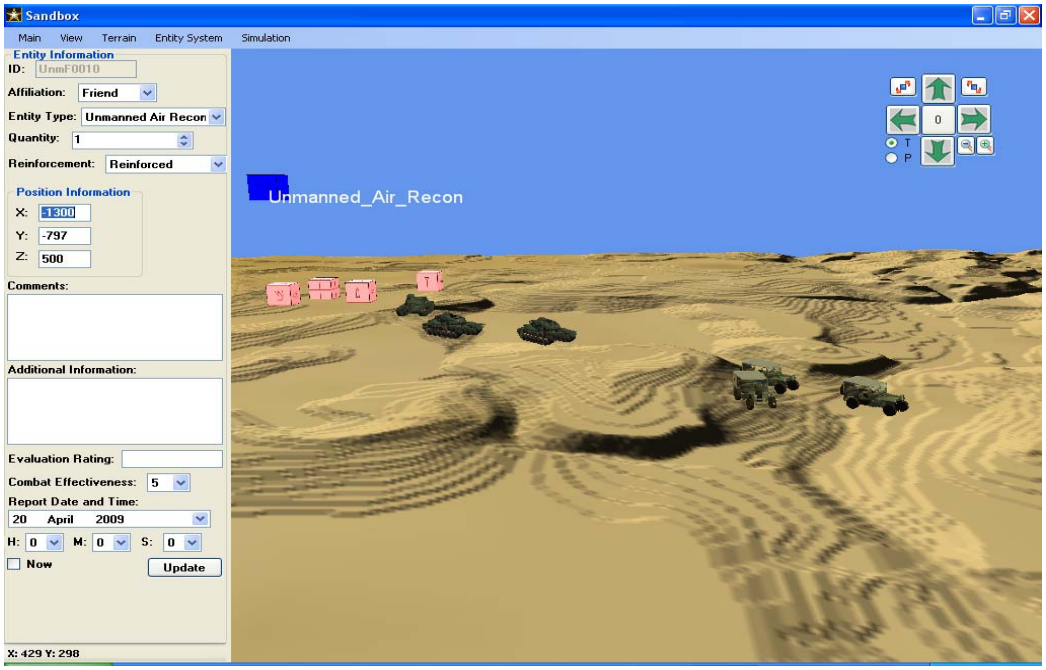
Şekil 5.64 Sandbox'taki son durumun raporlama aracında gösterilmesi

Düşman tanklarının, jeeplerimizi fark ettiğini gören gözlemleyici hemen bu durumu Şekil 5.65'deki gibi raporlama aracına girer. Raporlar istenen sayıda birim hakkında gönderilebilir. Her birimin verisi girildiğinde *Save* düğmesine tıklanır.



Şekil 5.65 Seçilen bir birim için yeni konum bilgisi ve yorumların girilmesi

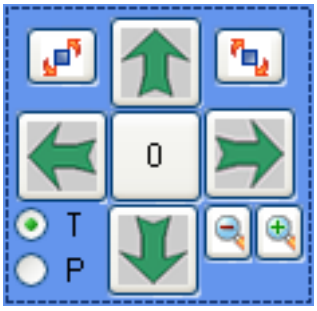
Tüm birimler hakkında raporun gönderilmesi için *Report* menüsünden *Send Report* seçeneği tıklanır. Böylece raporlar hazırlanan web servisi aracılığıyla Sandbox yazılımına iletilmiş olur.



Şekil 5.66 Gelen rapora göre Sandbox'ın güncellenmesi

Sandbox yazılımı da gönderilen raporu aldığında hemen rapor verilen tüm birimleri yeni duruma göre güncelleyerek grafik alanında gösterir. Burada yapılan tüm güncellemeler de yine animasyon biçiminde gösterilmektedir (Şekil 5.66).

Sağ üst tarafta sağlanan dolaşma paneli ise kullanıcıların savaş alanını diledikleri gibi görmelerini sağlar. Burada kameranın bakış açısı, kamera konumu değiştirilebilir. Görüntü iki yönde çevrilebilir ve savaş alanı üzerinde yakınlaşma-uzaklaşma yapılabilir. Şekil 5.67 bu paneli göstermektedir.



Şekil 5.67 Hareket paneli

6. SONUÇ KISMI

Bu kısımda tezin amacı ve sonucu tartışılmış, ardından bölümler kısaca özetlenmiş ve son olarak da gelecekte yapılabilecek çalışmalar üzerine tavsiyelerde bulunulmuştur.

6.1 Tezin Amacı

Tez çalışmasında hedeflenen amaç askeri savaş alanlarının sanal ortamda 3B olarak görselleştirilmesidir. Askeri durumlarda kullanılmak üzere kapsamlı, en güncel veriler üzerinden çalışan ve komutanların karar verme süreçlerinde yardımcı olacak ve böylece daha başarılı stratejiler izlemelerini sağlayacak bir sistemin günümüz teknolojileri kullanılarak geliştirilmesi amaçlanmıştır.

6.2 Tezin Sonucu

Çalışma sonucunda geliştirilen Sandbox, güncel ve güvenilir veriler kullanarak komutanların karar verme süreçlerini kolaylaştıracaktır. Savaş alanının gerçekçi bir gösterimi, durumun daha iyi bir biçimde algılanmasını sağlayacaktır.

Yazılımın tasarımı genel olarak basit tutulsa da gerekli askeri bilginin kapsamlı bir biçimde ele alınması sağlanmıştır. Ayrıca ilerideki gereksinimleri de karşılaması için esnek bir yapıda tasarlanmıştır.

Daha önce geliştirilen bazı sistemlerde (Arnett, 2004), askeri birimler basit geometrik şekillerle ifade edilmiştir. Her ne kadar çok sayıda geometrik şekil kullanılsa da bu kullanım birçok farklı türdeki askeri birimi göstermede yetersiz kalacaktır. Yapılan tez çalışmasında (Thibault , 2005) tarafından geliştirilen NATO-APP-6A sembolojisinin bir alt kümesi kullanılarak geniş bir yelpazede birçok askeri birimin eklenmesi sağlanmıştır.

Geliştirilen yazılım veri alışverişinde temel olarak (Hobbs, 2003)'un öne sürdüğü gibi XML biçimini kullanmaktadır. Bu biçim, farklı sistemler arasında veri alışverişini kolaylaştırdığı gibi verilerin biçiminin de kolayca değiştirilmesine olanak sağlar.

(Taylor, 2005), (Shiau ve Liang, 2007) ve (Wisher, 2001) tarafından geliştirilen sistemlerin aksine güncel bir askeri sembolojinin yanı sıra eklenen 3B modeller kullanıcıların farklı sembolleri daha rahat ayırt etmelerini sağlayacaktır. Böylelikle geliştirilen yazılım askeri eğitimlerde de etkili bir öğrenme aracı olarak kullanılabilir.

(Durbin, 1998), (Julier, 1999) tarafından yapılan uygulamalarda zaman kavramı bulunmamaktadır. Sandbox yazılımı ise zaman verisini de askeri birimlerin konum bilgilerine ekleyerek, hangi birimin ne zaman nerede olduğunu göstermiştir.

Durağan bir gösterimin yerine, kullanılan animasyonlar ile birimlerin hareketleri de yazılımda gösterilmiştir. Böylece dinamik bir ortamda daha gerçekçi bir savaş alanı benzetimi elde edilmiştir.

(Taylor , 1993) tarafından ayrıntılı bir biçimde incelenen Lanchester Savaş modelleri de geliştirilen Sandbox yazılımı ile bütünleştirilmiştir. Böylece başlangıç koşulları verilen bir savaşın nasıl sonlanacağı matematiksel olarak hesaplanarak kullanıcılara sunulur.

Web servislerinin de kullanımı ile uzak saha gözlemleyicilerinden gelen raporların da sistemi etkilemesi sağlanmıştır. Aynı zamanda savaş alanını gösteriminin de son durumu gözlemleyicilere gönderilerek gösterimin kontrolü de yapılmaktadır. Böylece Sandbox (Tolk, 2003)' da tanımlanan modelleme ve simülasyon ortamına tam anlamıyla uymaktadır.

Sonuç olarak Sandbox yazılımı, görsel özellikleri ve gelişmiş veri işleme yetenekleri ile savaş alanı yönetimi ve savaş oyunlarının simülasyonu konularında yeni adımlar atmıştır.

6.3 Bölümlerin Amacı ve Özeti

Bu tez çalışması 5 bölümden oluşmaktadır. Çalışmanın birinci kısmı özet ve giriş bölümlerinden oluşur. Özet bölümünde tez çalışması kısa ve öz bir biçimde anlatılmıştır. Giriş kısmında ise yapılan tez çalışmasının amacı ve neden sanal bir kum sandığına gerek olduğu açıklanmıştır. Ayrıca daha önce yapılan çalışmalara da değinilmiştir.

İkinci kısımda ise tez kapsamında kullanılan yöntemlerin arkasında yatan teorik bilgi verilmiştir. Bu kısımda bilgisayar grafiğinin temellerinden, karar vermek için kullanılan matematiksel modellere; veritabanlarından, web servislerine kadar geniş bir yelpazede konular kısa bir biçimde açıklanmıştır.

Üçüncü kısımda tez çalışmasında kullanılan yazılımlar anlatılmıştır. Bahsedilen askeri sembolojinin tez kapsamında kullanılan alt kümesi, sembolojinin özellikleriyle birlikte sunulmuştur. Bu bölümde ayrıca kullanılan 3B askeri modeller de gösterilmiştir.

Dördüncü kısımda, yazılımların geliştirme aşamaları ayrıntılı bir biçimde anlatılmıştır. Bu kısımda üç adet yazılım üretilmiştir. Bunlardan ilki bir arazinin görselleştirilmesidir. İkincisi askeri durumlarda kullanılabilecek karar verme modellerini içeren bir araçtır. Son ve tez çalışmasının asıl amacı olan yazılım ise askeri bir savaş alanının gösterilmesini, bu savaş alanı üzerindeki askeri birimlerin konumlandırılmasını ve savaş alanındaki güncel bilgilerle değiştirilebilmesini sağlar.

Beşinci kısım, geliştirilen yazılımların özelliklerini anlatır. Bu bölümde yazılımlara ait birçok ekran görüntüsü verilmiştir.

6.4 Gelecekteki Çalışmalar İçin Tavsiyeler

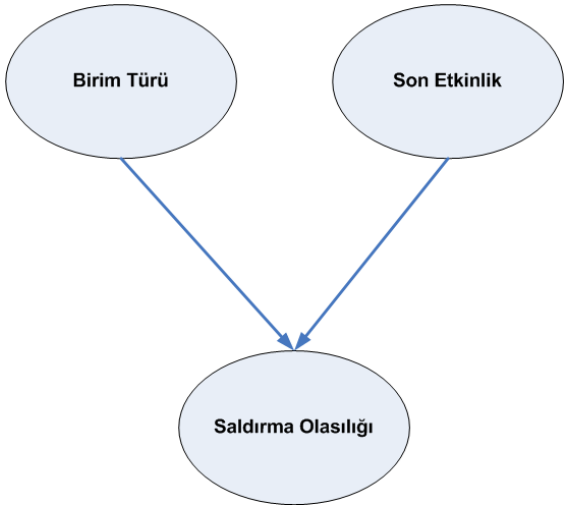
Yapılan çalışmada her ne kadar kapsamlı olmaya çalışıldıysa da savaş alanı benzetimi konusu sürekli yenilenen teknolojiye ayak uydurmak açısından önü açık bir konudur. İleride yapılacak çalışmalar için aşağıdaki tavsiyeler yeni fikirler üretilmesinde yardımcı olabilir:

• Daha gerçekçi bir sanal ortamın oluşturulması için daha fazla model sisteme eklenebilir. Bina modelleri, bazı yeryüzü şekilleri gibi modellerin eklenmesi kullanıcıların savaş alanını tüm ayrıntılarıyla görmelerine yardımcı olacaktır. Buna ek olarak, bir kurtarma operasyonunun planlanması gibi bazı özel durum senaryoları için şehrin bir kısmının tamamının modeli sisteme eklenebilir.

• Savaş alanının tam olarak canlandırılması için çatışma etkileri ve patlamalar eklenebilir. Bu etkilerin ne seviyede ekleneceği geliştiricinin amacına göre değişir. Gerçek dünya verileriyle çalışan bir strateji oyunu geliştirmek de mümkündür; strateji geliştirmek için kullanılan ve bazı sınırlı animasyon özellikleri olan bir araç da elde edilebilir.

• Tez kapsamında geliştirilen web servisinin tüm NET uygulamaları ile birlikte kullanılabilmesi, bu web servisi mobil aygıtların (akıllı telefon, PDA gibi…) da birer rapor verme aracı olarak düşünülmesi fikrini akla getirmektedir. Geliştirilecek uygulamalar ile uzak gözlemcilerin yanı sıra askerlerin de taşıdıkları mobil aygıtlar aracılığıyla son durumu sisteme bildirmeleri sağlanabilir.

• Birçok farklı alanda sebep sonuç ilişkileri kuran BBN (Bayesian Belief Networks) modelleme açısından etkili bir araçtır. Elimizde hazır olan bir bilgi varsa ve bundan sonra gelen verinin güvenilirliği şüpheliyse bu iki veriye göre bir karar verme mümkün olabilir. Örnek vermek gerekirse elimizde bir askeri birimin var olduğu biliniyor. Daha sonra bu birimle ilgili bir etkinlik raporu geldiğinde bu raporun güvenirliğine göre bir karar verme mekanizması kurulabilir. Bu duruma ilişkin BBN aşağıdaki Şekil 6.68'de de gösterilmiştir.



Şekil 6.68 Tavsiye edilen örnek bir BBN gösterimi

KAYNAKLAR

Arnett Cadet Daniel. et al. "Tactical Terrain Visualization System", Systems and Software Technology Conference, USA, 2004.

Campbell, "Introduction to Remote Sensing", Taylor &Francis, 2002.

Ceruti M. G., "Challenges in Data Management for the United States Department of Defense (DOD) Command, Control, Communications, Computers and Intelligence (C4I) Systems", Proc. 22nd Ann. Int'l Computer Software and Applications Conf., IEEE COMPSAC '98, pp. 622-629, Aug. 1998

Ceruti M. G., "Data Management Challenges and Development for Military Information Systems", IEEE Transactions of Knowledge and Data Engineering, Vol 15, No 5, 2003

Deitel and Deitel, "C# for Programmers", Prentice Hall, USA, 2006.

Durbin, J.; Swan, J.E. et al, "Battlefield visualization on the responsive workbench", Visualization apos'98. Proceedings, pp. 463-466, 1998.

Elmasri, Navathe, "Fundamentals of Database Systems", Addison-Wesley, 2004

Giordano, F. R., M. D. Weir, and W. P. Fox. "A First Course in Mathematical Modeling", CA: Books/Cole Publishing Company, 1997

Hearn D, Baker P, "Computer Graphics with OpenGL", Prentice Hall, 2003

Hobbs Reginald L.. "Using XML to Support Military Decision Making", XML Conference and Exposition Proceedings, USA, 2003.

Julier Simon 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.

Kanbur N., "3D Studio Max Visualization and Modelling", Pusula Publishing, Turkey, 2005.

Koyuncu B, Bostancı E, "A Scenario Based Virtual Military Sandbox Implementation Using Web Services", Proceedings of International Conference on Advanced Computer Control, Singapore, 2009

Lobao, Evangelista and Faris, "Beginning XNA 2.0 Game Programming", Apress, USA, 2008

Longley, Paul et al. "Geographic Information Systems and Science", Wiley, ISBN-13: 978-0470870013, 2005.

Ming Li. “Military Decision Modeling with Conflict Analysis”, IEEE International Conference on Systems, Man and Cybernetics, 1996, China.

MSDN XNA Documentation, http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.game.update.aspx (Son erişim tarihi: 30.04.2009)

Neider, M Woo, T Davis , “OpenGL programming guide”, Addison-Wesley Reading, Mass, 1993.

Pullen M., Hieb M., Sudnikovich W., Champs P., “An International Experiment in Command and Control-Simulation Interoperability Using Web Services”, Proc. 10th Int’l Symposium on Distributed Simulation and Real-Time Applications, IEEE, 2006

Russel S. and Norvig P. “Artificial Intelligence A Modern Approach”, Prentice Hall, 2003

Stewart Bart D. “An Interactive Use of Lanchester Combat Model”, JOMA.

Shiau Y, Liang S, “Real-time Network Virtual Military Simulation System”, 11th International Conference Information Visualization, IEEE, Switzerland, 2007

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

Taylor James, “An introduction to Lanchester-Type Models of Warfare”, Naval Postgraduate School, California, 1993.

Thibault D.U, “Commented APP-6A-Military symbols for land based systems”, Canada, 2005.

Tolk A., “A Common Framework for Military M&S and C4I Systems”, Spring Simulation Interoperability Workshop, 2003, Florida, USA

Williams A. P., “Richardson Arms Race Model”,
http://shakti.trincoll.edu/~pbrown/armsrace.html Erişim Tarihi: 29.12.2008

Wiper et al. “Bayesian Inference for a Lanchester Type Combat Model”, Naval Research Logistics, Volume 47, Issue 7, pp541-558, 2000

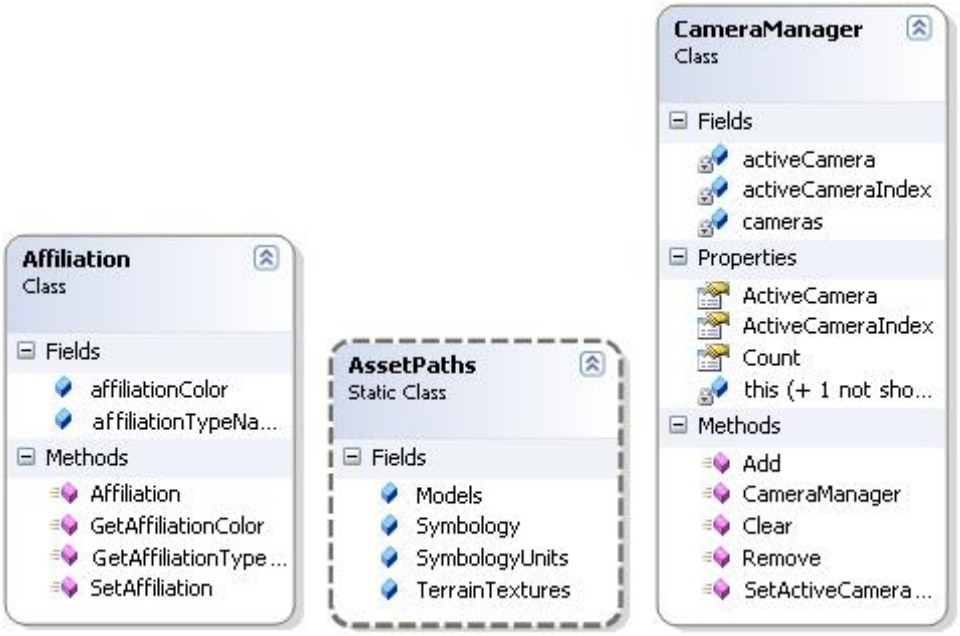
Wisher R. A. et al. “The Virtual Sand Table: Intelligent Tutoring for Field Artillery Training”, Research Report 1768, US Army Research Institute for Behavioral and Social Sciences, USA, 2001

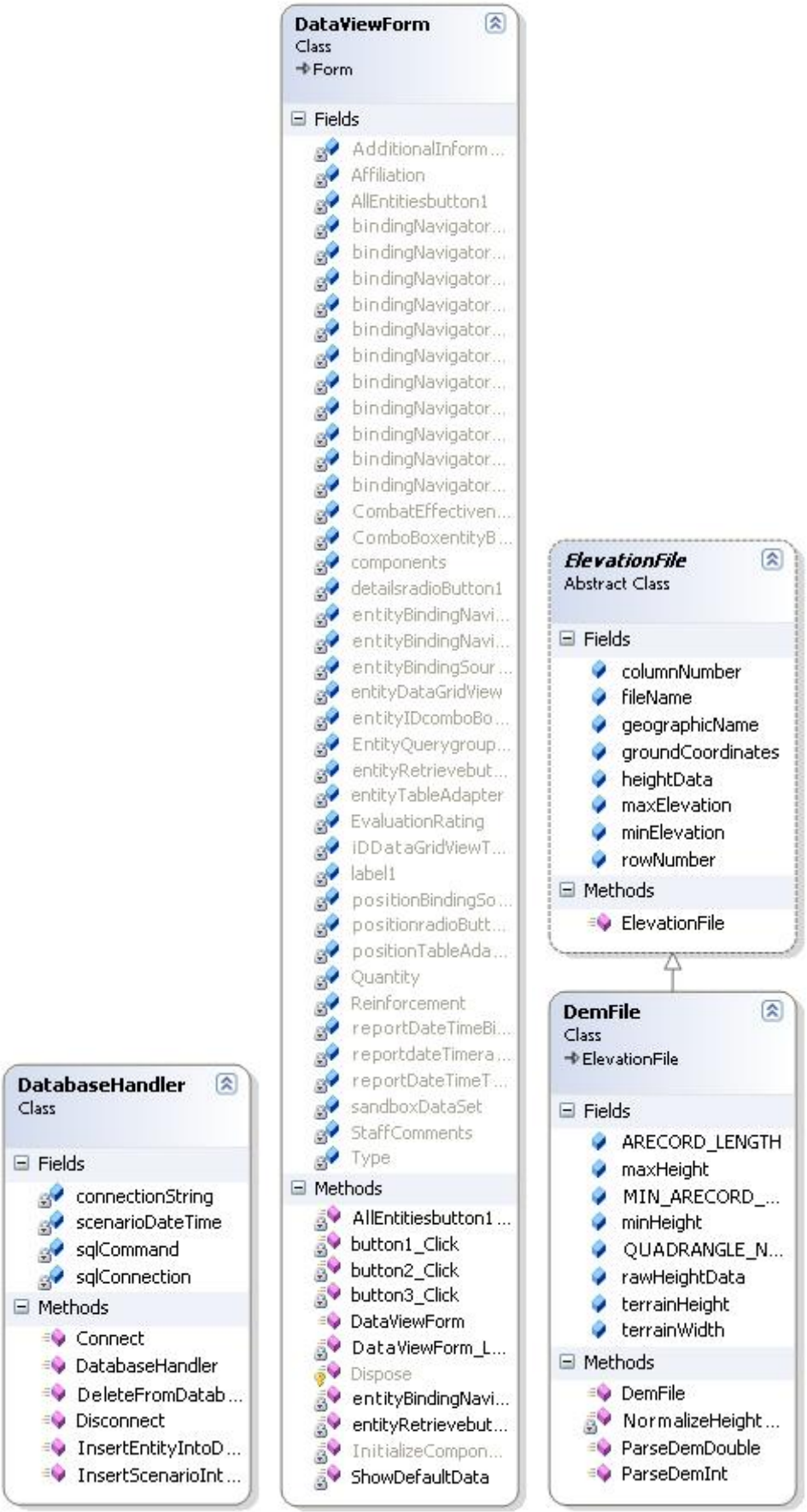
Yang J. B., Singh M. G. “An evidential Reasoning Approach for Multiple-Attribute Decision Making with Uncertainty”, IEE Trans. SMC-9, Vol.24 No.1, 1994

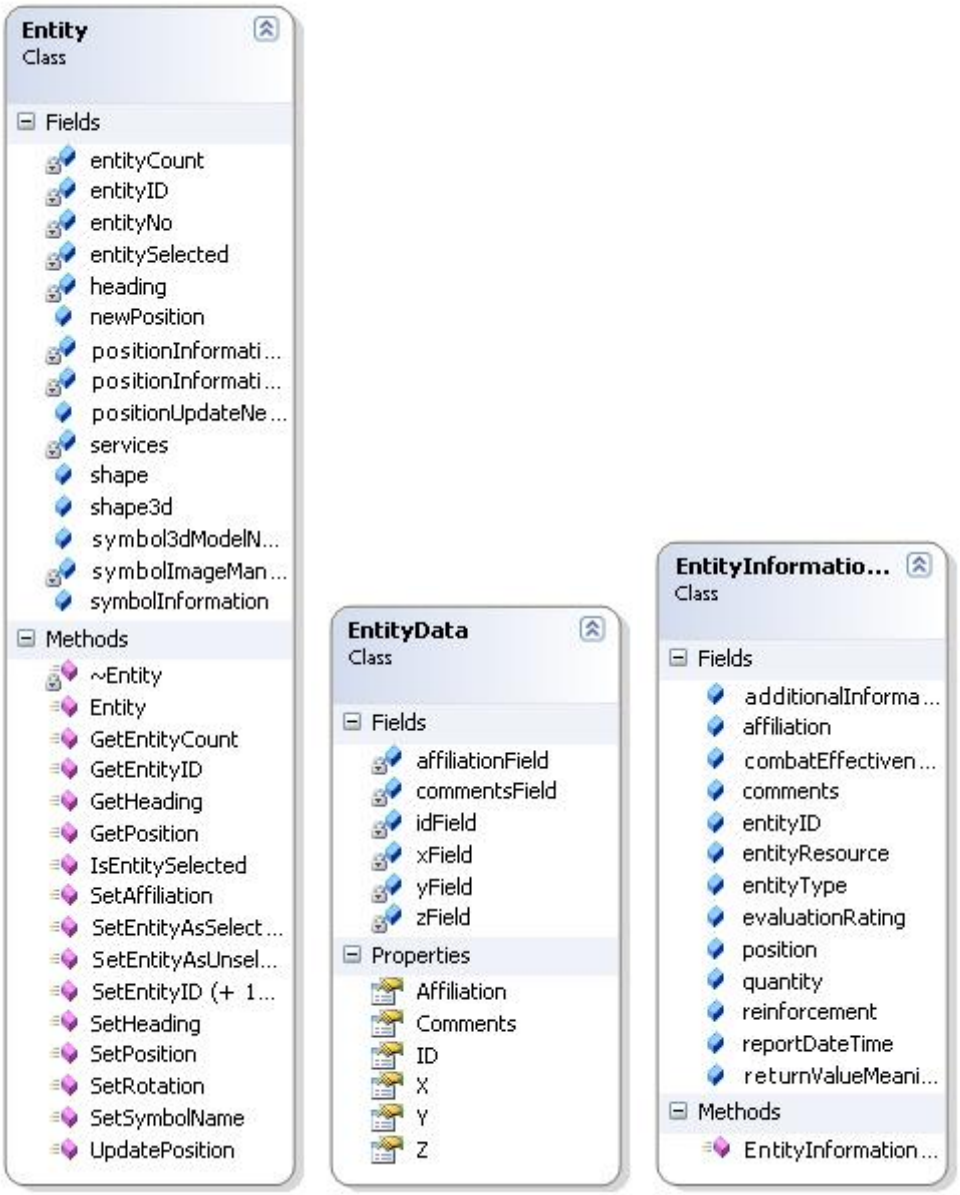
EKLER

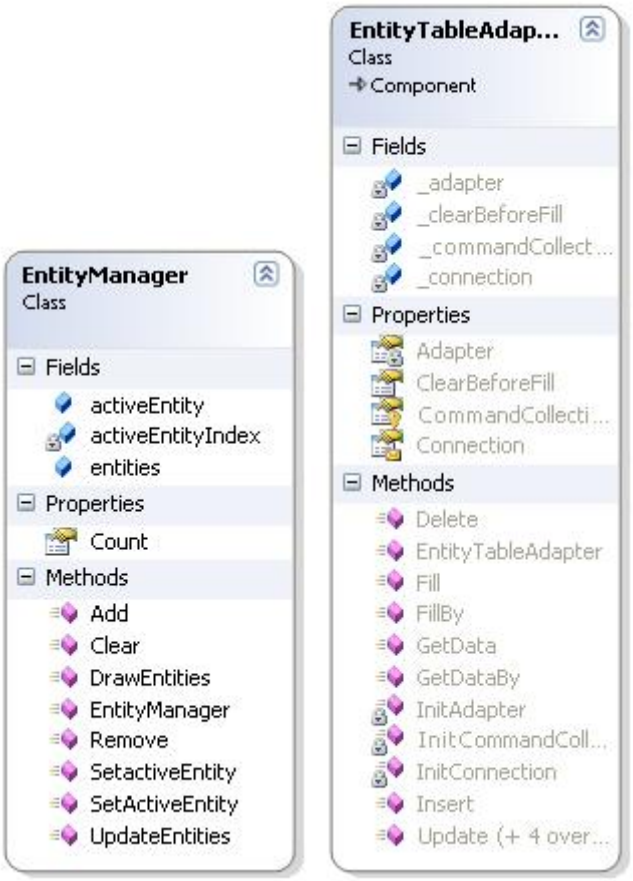
EK 1 UML Diyagramları

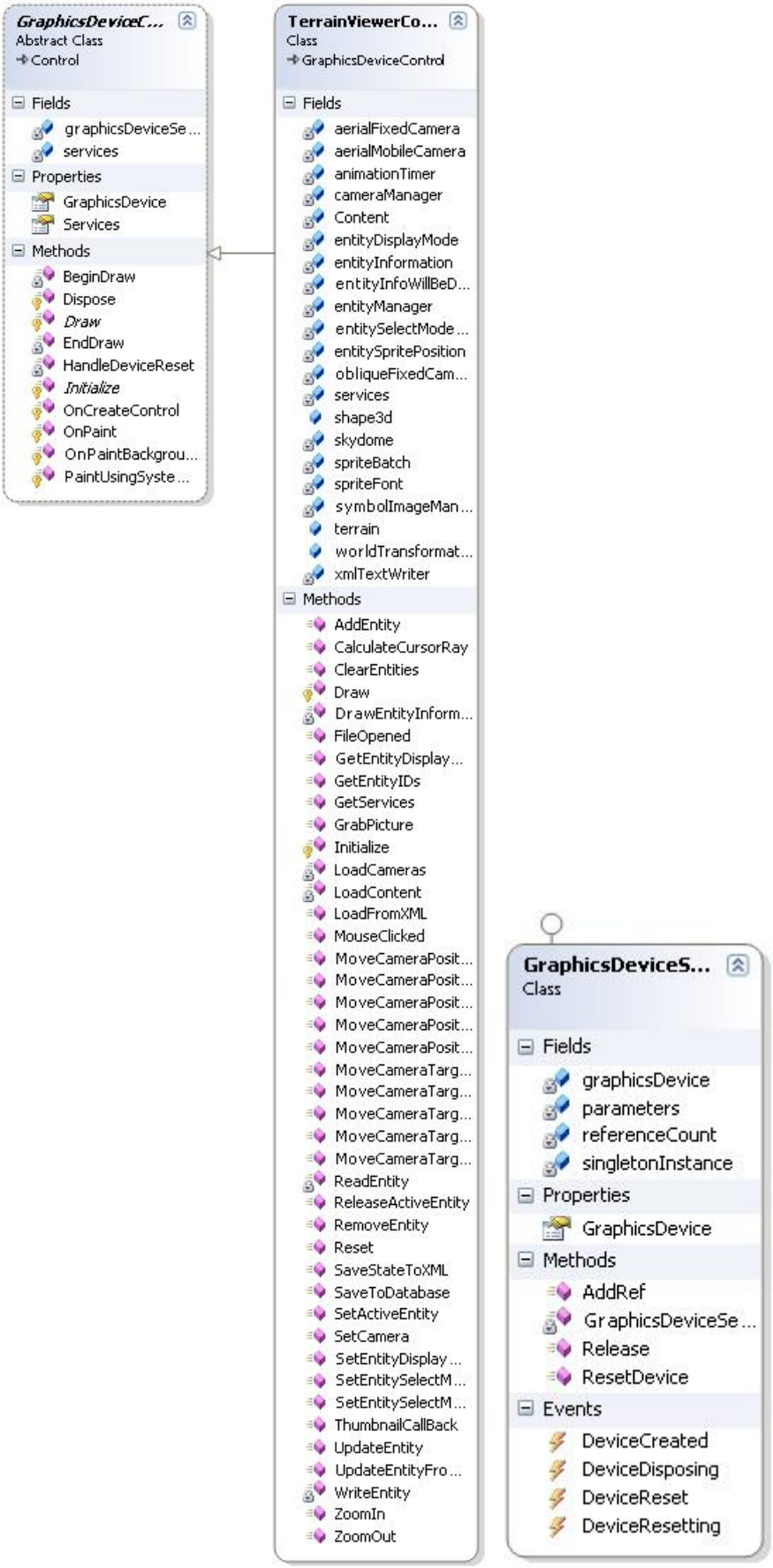
Sandbox yazılımı için geliştirilen tüm sınıflara ait UML diyagramları aşağıda verilmiştir.

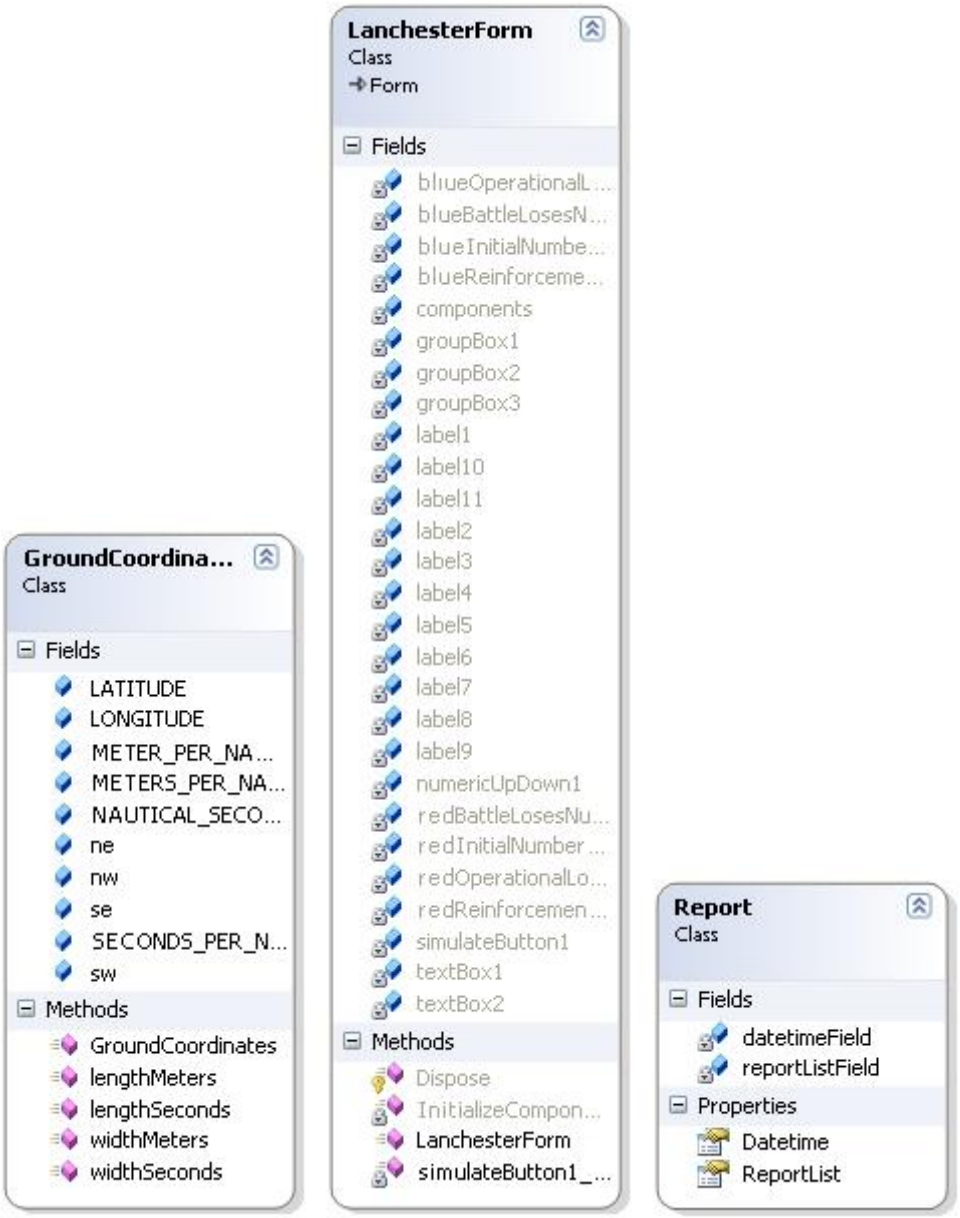


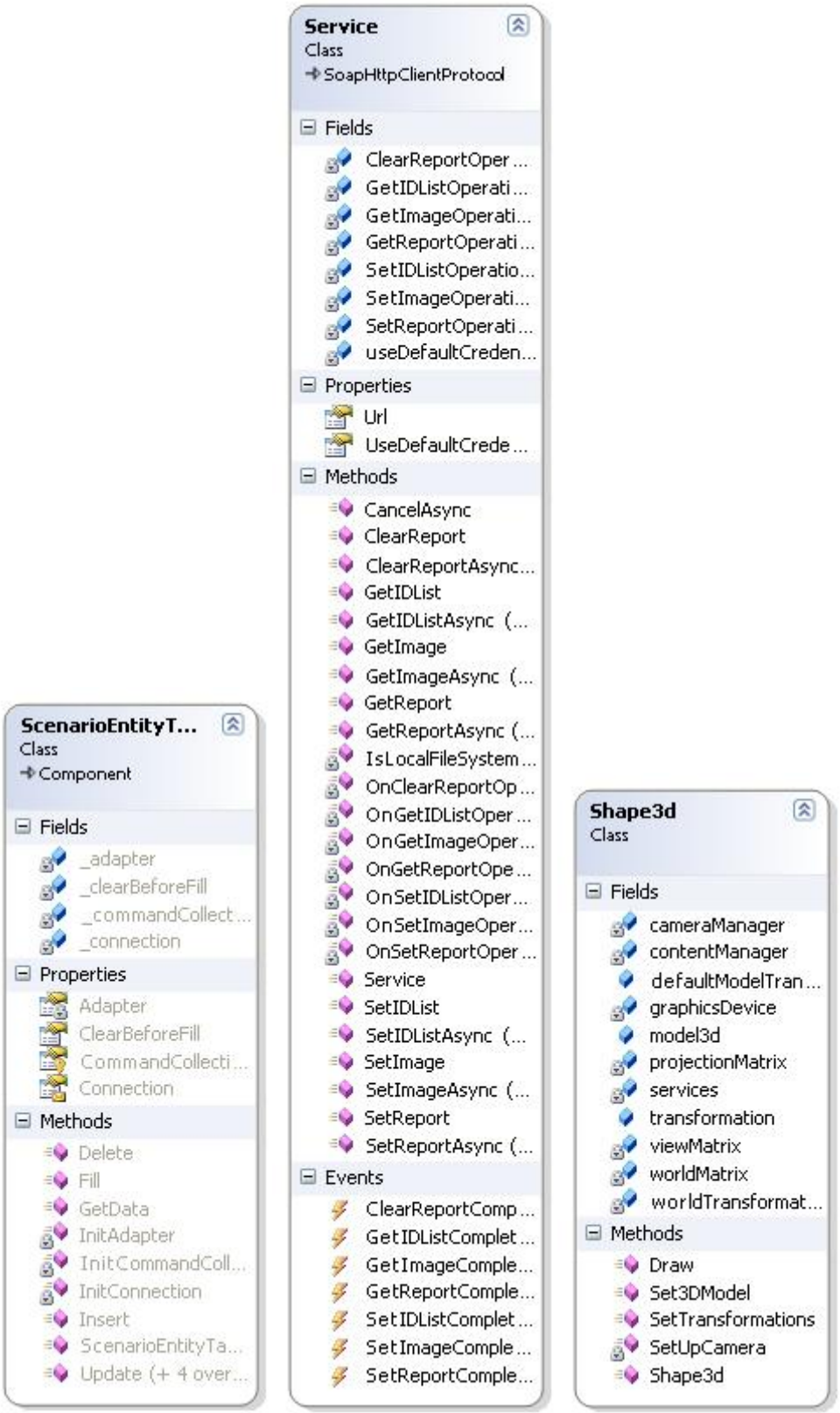

DataViewForm
Class
Form
Fields
AdditionalInform...
Affiliation
AllEntitiesbutton1
bindingNavigator...
bindingNavigator...
bindingNavigator...
bindingNavigator...
bindingNavigator...
bindingNavigator...
bindingNavigator...
bindingNavigator...
bindingNavigator...
bindingNavigator...
bindingNavigator...
CombatEffectiven...
ComboBoxentityB...
components
detailsradioButton1
entityBindingNavi...
entityBindingNavi...
entityBindingSour...
entityDataGridView
entityIDcomboBo...
EntityQuerygroup...
entityRetrievebut...
entityTableAdapter
EvaluationRating
IDDataGridViewT...
label1
positionBindingSo...
positionradioButt...
positionTableAda...
Quantity
Reinforcement
reportDateTimeBi...
reportdateTimera...
reportDateTimeT...
sandboxDataSet
StaffComments
Type
Methods
AllEntitiesbutton1...
button1_Click
button2_Click
button3_Click
DataViewForm
DataViewForm_L...
Dispose
entityBindingNavi...
entityRetrievebut...
InitializeCompon...
ShowDefaultData
ElevationFile
Abstract Class
Fields
columnNumber
fileName
geographicName
groundCoordinates
heightData
maxElevation
minElevation
rowNumber
Methods
ElevationFile
DemFile
Class
ElevationFile
Fields
ARECORD_LENGTH
maxHeight
MIN_ARECORD_...
minHeight
QUADRANGLE_N...
rawHeightData
terrainHeight
terrainWidth
Methods
DemFile
NormalizeHeight...
ParseDemDouble
ParseDemInt
DatabaseHandler
Class
Fields
connectionString
scenarioDateTime
sqlCommand
sqlConnection
Methods
Connect
DatabaseHandler
DeleteFromDatab...
Disconnect
InsertEntityIntoD...
InsertScenarioInt...

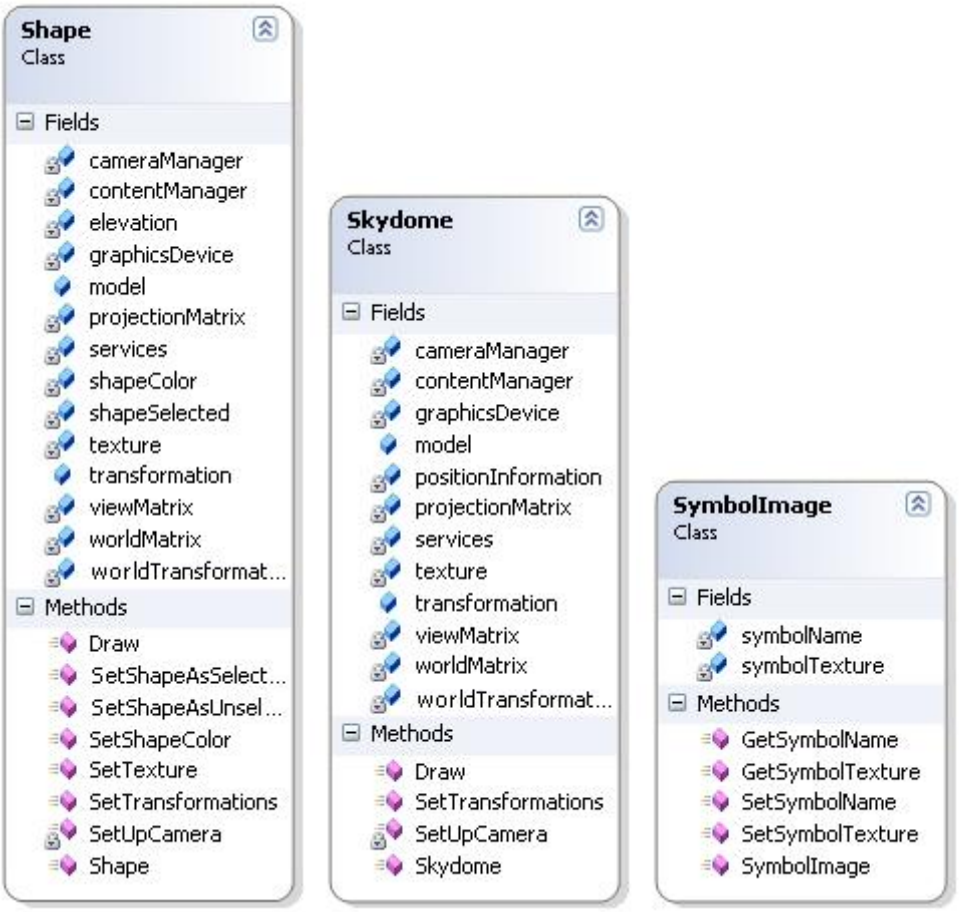

Entity
Class
Fields
entityCount
entityID
entityNo
entitySelected
heading
newPosition
positionInformati...
positionInformati...
positionUpdateNe...
services
shape
shape3d
symbol3dModelN...
symbolImageMan...
symbolInformation
Methods
~Entity
Entity
GetEntityCount
GetEntityID
GetHeading
GetPosition
IsEntitySelected
SetAffiliation
SetEntityAsSelect...
SetEntityAsUnsel...
SetEntityID (+ 1...
SetHeading
SetPosition
SetRotation
SetSymbolName
UpdatePosition
EntityData
Class
Fields
affiliationField
commentsField
idField
xField
yField
zField
Properties
Affiliation
Comments
ID
X
Y
Z
EntityInformatio...
Class
Fields
additionalInforma...
affiliation
combatEffectiven...
comments
entityID
entityResource
entityType
evaluationRating
position
quantity
reinforcement
reportDateTime
returnValueMeani...
Methods
EntityInformation...

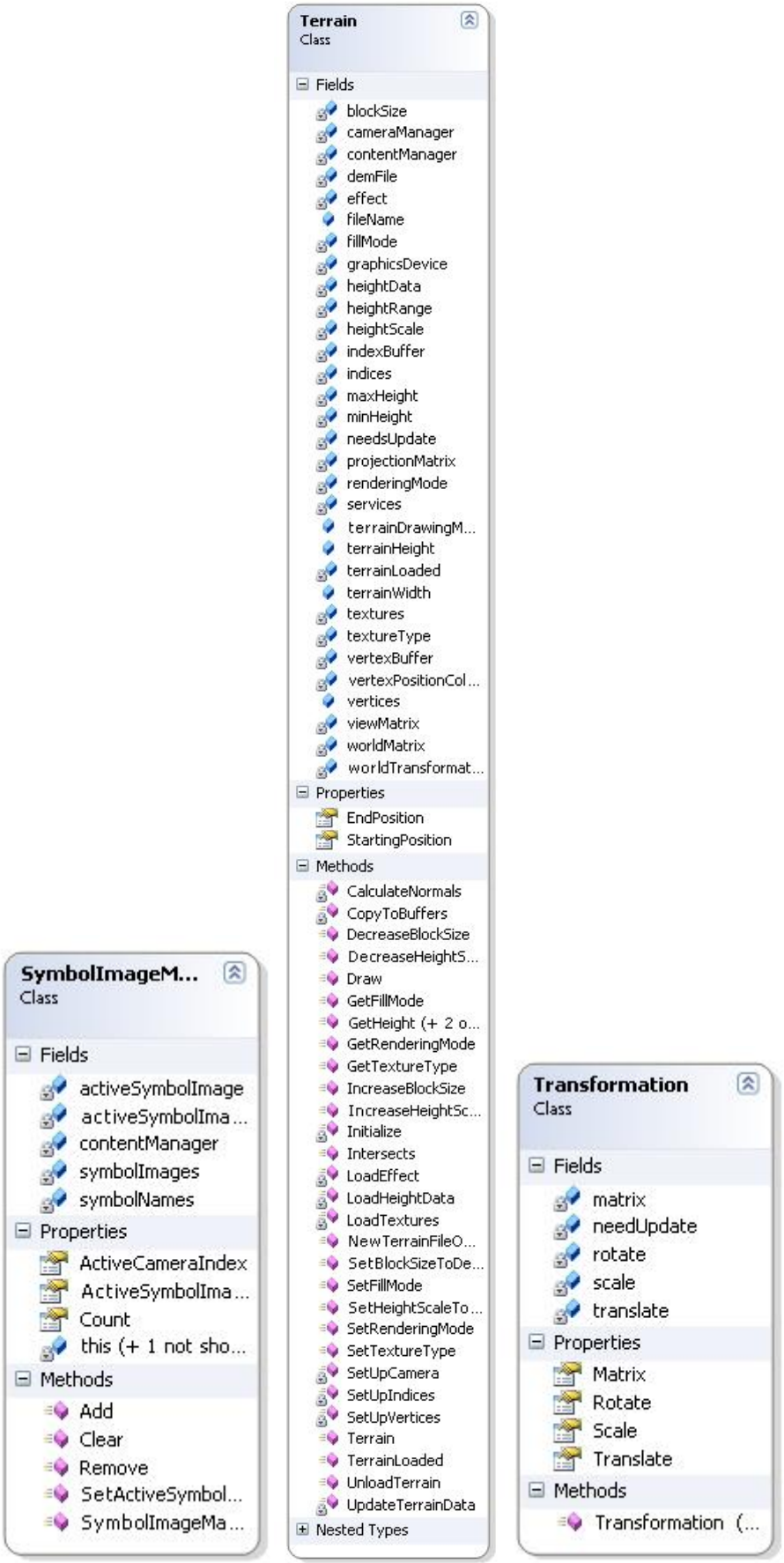

EntityManager
Class
Fields
activeEntity
activeEntityIndex
entities
Properties
Count
Methods
Add
Clear
DrawEntities
EntityManager
Remove
SetactiveEntity
SetActiveEntity
UpdateEntities
EntityTableAdap...
Class
Component
Fields
_adapter
_clearBeforeFill
_commandCollect...
_connection
Properties
Adapter
ClearBeforeFill
CommandCollecti...
Connection
Methods
Delete
EntityTableAdapter
Fill
FillBy
GetData
GetDataBy
InitAdapter
InitCommandColl...
InitConnection
Insert
Update (+ 4 over...

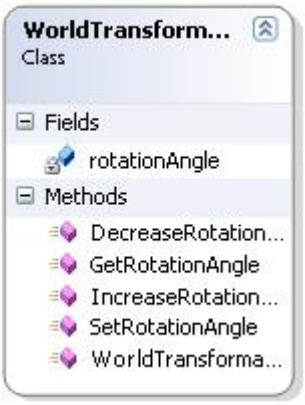

GraphicsDeviceC...
Abstract Class
Control
Fields
graphicsDeviceSe...
services
Properties
GraphicsDevice
Services
Methods
BeginDraw
Dispose
Draw
EndDraw
HandleDeviceReset
Initialize
OnCreateControl
OnPaint
OnPaintBackgrou...
PaintUsingSyste...
TerrainViewerCo...
Class
GraphicsDeviceControl
Fields
aerialFixedCamera
aerialMobileCamera
animationTimer
cameraManager
Content
entityDisplayMode
entityInformation
entityInfoWillBeD...
entityManager
entitySelectMode...
entitySpritePosition
obliqueFixedCam...
services
shape3d
skydome
spriteBatch
spriteFont
symbolImageMan...
terrain
worldTransformat...
xmlTextWriter
Methods
AddEntity
CalculateCursorRay
ClearEntities
Draw
DrawEntityInform...
FileOpened
GetEntityDisplay...
GetEntityIDs
GetServices
GrabPicture
Initialize
LoadCameras
LoadContent
LoadFromXML
MouseClicked
MoveCameraPosit...
MoveCameraPosit...
MoveCameraPosit...
MoveCameraPosit...
MoveCameraPosit...
MoveCameraTarg...
MoveCameraTarg...
MoveCameraTarg...
MoveCameraTarg...
MoveCameraTarg...
ReadEntity
ReleaseActiveEntity
RemoveEntity
Reset
SaveStateToXML
SaveToDatabase
SetActiveEntity
SetCamera
SetEntityDisplay...
SetEntitySelectM...
SetEntitySelectM...
ThumbnailCallBack
UpdateEntity
UpdateEntityFro...
WriteEntity
ZoomIn
ZoomOut
GraphicsDeviceS...
Class
Fields
graphicsDevice
parameters
referenceCount
singletonInstance
Properties
GraphicsDevice
Methods
AddRef
GraphicsDeviceSe...
Release
ResetDevice
Events
DeviceCreated
DeviceDisposing
DeviceReset
DeviceResetting


LanchesterForm
Class
Form
Fields
blueOperationalL...
blueBattleLosesN...
blueInitialNumbe...
blueReinforceme...
components
groupBox1
groupBox2
groupBox3
label1
label10
label11
label2
label3
label4
label5
label6
label7
label8
label9
numericUpDown1
redBattleLosesNu...
redInitialNumber...
redOperationalLo...
redReinforcemen...
simulateButton1
textBox1
textBox2
Methods
Dispose
InitializeCompon...
LanchesterForm
simulateButton1_...
GroundCoordina...
Class
Fields
LATITUDE
LONGITUDE
METER_PER_NA...
METERS_PER_NA...
NAUTICAL_SECO...
ne
nw
se
SECONDS_PER_N...
sw
Methods
GroundCoordinates
lengthMeters
lengthSeconds
widthMeters
widthSeconds
Report
Class
Fields
datetimeField
reportListField
Properties
Datetime
ReportList


Service
Class
SoapHttpClientProtocol
Fields
ClearReportOper...
GetIDListOperati...
GetImageOperati...
GetReportOperati...
SetIDListOperatio...
SetImageOperati...
SetReportOperati...
useDefaultCreden...
Properties
Url
UseDefaultCrede...
Methods
CancelAsync
ClearReport
ClearReportAsync...
GetIDList
GetIDListAsync (...
GetImage
GetImageAsync (...
GetReport
GetReportAsync (...
IsLocalFileSystem...
OnClearReportOp...
OnGetIDListOper...
OnGetImageOper...
OnGetReportOpe...
OnSetIDListOper...
OnSetImageOper...
OnSetReportOper...
Service
SetIDList
SetIDListAsync (...
SetImage
SetImageAsync (...
SetReport
SetReportAsync (...
Events
ClearReportComp...
GetIDListComplet...
GetImageComple...
GetReportComple...
SetIDListComplet...
SetImageComple...
SetReportComple...
ScenarioEntityT...
Class
Component
Fields
_adapter
_clearBeforeFill
_commandCollect...
_connection
Properties
Adapter
ClearBeforeFill
CommandCollecti...
Connection
Methods
Delete
Fill
GetData
InitAdapter
InitCommandColl...
InitConnection
Insert
ScenarioEntityTa...
Update (+ 4 over...
Shape3d
Class
Fields
cameraManager
contentManager
defaultModelTran...
graphicsDevice
model3d
projectionMatrix
services
transformation
viewMatrix
worldMatrix
worldTransformat...
Methods
Draw
Set3DModel
SetTransformations
SetUpCamera
Shape3d


Shape
Class
Fields
cameraManager
contentManager
elevation
graphicsDevice
model
projectionMatrix
services
shapeColor
shapeSelected
texture
transformation
viewMatrix
worldMatrix
worldTransformat...
Methods
Draw
SetShapeAsSelect...
SetShapeAsUnsel...
SetShapeColor
SetTexture
SetTransformations
SetUpCamera
Shape
Skydome
Class
Fields
cameraManager
contentManager
graphicsDevice
model
positionInformation
projectionMatrix
services
texture
transformation
viewMatrix
worldMatrix
worldTransformat...
Methods
Draw
SetTransformations
SetUpCamera
Skydome
SymbolImage
Class
Fields
symbolName
symbolTexture
Methods
GetSymbolName
GetSymbolTexture
SetSymbolName
SetSymbolTexture
SymbolImage


Terrain
Class
Fields
blockSize
cameraManager
contentManager
demFile
effect
fileName
fillMode
graphicsDevice
heightData
heightRange
heightScale
indexBuffer
indices
maxHeight
minHeight
needsUpdate
projectionMatrix
renderingMode
services
terrainDrawingM...
terrainHeight
terrainLoaded
terrainWidth
textures
textureType
vertexBuffer
vertexPositionCol...
vertices
viewMatrix
worldMatrix
worldTransformat...
Properties
EndPosition
StartingPosition
Methods
CalculateNormals
CopyToBuffers
DecreaseBlockSize
DecreaseHeightS...
Draw
GetFillMode
GetHeight (+ 2 o...
GetRenderingMode
GetTextureType
IncreaseBlockSize
IncreaseHeightSc...
Initialize
Intersects
LoadEffect
LoadHeightData
LoadTextures
NewTerrainFileO...
SetBlockSizeToDe...
SetFillMode
SetHeightScaleTo...
SetRenderingMode
SetTextureType
SetUpCamera
SetUpIndices
SetUpVertices
Terrain
TerrainLoaded
UnloadTerrain
UpdateTerrainData
Nested Types
SymbolImageM...
Class
Fields
activeSymbolImage
activeSymbolIma...
contentManager
symbolImages
symbolNames
Properties
ActiveCameraIndex
ActiveSymbolIma...
Count
this (+ 1 not sho...
Methods
Add
Clear
Remove
SetActiveSymbol...
SymbolImageMa...
Transformation
Class
Fields
matrix
needUpdate
rotate
scale
translate
Properties
Matrix
Rotate
Scale
Translate
Methods
Transformation (...


WorldTransform...
Class
Fields
rotationAngle
Methods
DecreaseRotation...
GetRotationAngle
IncreaseRotation...
SetRotationAngle
WorldTransforma...

ÖZGEÇMİŞ

Adı Soyadı: Gazi Erkan BOSTANCI

Doğum Yeri: Ankara

Doğum Tarihi: 30.07.1985

Medeni Hali: Bekâr

Yabancı Dili: İngilizce, Almanca

Eğitim Durumu

Lise: Nermin-Mehmet Çekiç Anadolu Lisesi: 1996 – 2003

Lisans: Ankara Üniversitesi Bilgisayar Mühendisliği: 2003 – 2007

Yüksek Lisans: Ankara Üniversitesi Bilgisayar Mühendisliği Bölümü: 2007 – 2009

Çalıştığı Kurumlar

Ankara Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı Eylül 2007 –

Yayınları

Bilimsel Dergiler

1. Koyuncu B., Bostancı E., "Development of a Software Tool for Decision Making in Combat Models", International Journal of Mathematics and Computation, Vol. 2, No. M09, CESER, 2009
2. Koyuncu B., Bostancı E., "Development of an Image Processing Software", Computer Engineering Journal, Vol. 1, No. 2, 2007

Konferans, Sempozyum ve Bilimsel Toplantı Yayınları

3. Koyuncu B., Bostancı E., "A Scenario Based Virtual Military Sandbox Implementation Using Web Services", Proceedings of the International Conference on Advanced Computer Control, IEEE, Singapore, 2009
4. Koyuncu B., Bostancı E., "Using Lanchester combat models to aid battlefield visualization", ICCSIT 2009 –Kabul Edildi
5. Koyuncu B., Bostancı E., "Using Web Services to Support Battlefield Visualization and Tactical Decision Making", CSSIM 2009 –Kabul Edildi
6. Koyuncu B., Bostancı E., "Terrain Visualization Using TerraGen, 3D Studio Max and OpenGL", Proceedings of the International Joint Seminar in Engineering, Indonesia, 2008
7. Koyuncu B., Bostancı E., "Virtual Reconstruction of an Ancient Site: Ephesus", Proceedings of the XI Symposium on Mediterranean Archaeology, Archaeopress, UK, 2009