EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

YAPAY ZEKA TEKNİKLERİ KULLANAN ÜÇ BOYUTLU GRAFİK YAZILIMLARI İÇİN "EXTENSIBLE 3D" (X3D) İLE BİR ALTYAPI OLUŞTURULMASI VE GERÇEKLEŞTİRİMİ

Tahir Emre KALAYCI

Bilgisayar Mühendisliği Anabilim Dalı

Bilim Dalı Kodu : 619.01.00

Sunuş Tarihi : 27.07.2006

Tez Danışmanı : Yrd. Doç. Dr. Aybars UĞUR

BORNOVA – İZMİR

Tahir Emre KALAYCI tarafından YÜKSEK LİSANS TEZİ olarak sunulan "Yapay Zeka Teknikleri Kullanan Üç Boyutlu Grafik Yazılımları İçin Extensible 3D (X3D) İle Bir Altyapı Oluşturulması ve Gerçekleştirimi" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 27.07.2006 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:	**İmza**
Jüri Başkanı : Yrd. Doç. Dr. Aybars UĞUR	.....
Raportör Üye: Prof. Dr. Serdar KORUKOĞLU	.....
Üye : Yrd. Doç. Dr. Muhammet G. CİNSDİKİCİ	.....

IV

ÖZET

YAPAY ZEKA TEKNİKLERİ KULLANAN ÜÇ BOYUTLU GRAFİK YAZILIMLARI İÇİN "EXTENSIBLE 3D" (X3D) İLE BİR ALTYAPI OLUŞTURULMASI VE GERÇEKLEŞTİRİMİ

KALAYCI, Tahir Emre

Yüksek Lisans Tezi, Fen Bilimleri Enstitüsü

Tez Yönetici: Yrd. Doç. Dr. Aybars UĞUR

Temmuz 2006, 114 sayfa

Bu tez çalışmasında, yapay zekanın güçlü tekniklerinden olan genetik algoritmalar yardımıyla, birçok alanda yaygın kullanılan Gezgin Satıcı Probleminin çözümü üzerinde durulmuştur. Üç boyutlu uzayda belirtilen tüm noktaların, en kısa yol uzunluğunu hedefleyerek dolaşılmasını sağlayan web tabanlı bir yazılımın gerçekleştirimi yapılmıştır. Problemin farklı boyutları için, çözüm başarısı değişik parametre değerleri kullanılarak ölçülmüştür. Eniyileme (optimizasyon) yöntemlerinin sonuçlarının daha iyi görülebilmesini de sağlayacak şekilde Extensible3D (X3D) yardımı ile görselleştirme aracı da geliştirilmiştir. Web3D teknolojilerinden olgunlaşma sürecindeki X3D'nin (XML tabanlı VRML olarak da düşünülen) yapay zeka alanındaki kullanımı denenmiştir.

Geliştirilen bu araç yardımıyla üç boyutlu ortamlarda değişik genetik algoritma parametreleri kullanılarak yol planlaması yapılabilmekte ve güzergah, üç boyutlu helikopter simülasyonu şeklinde görüntülenebilmektedir.

Anahtar Sözcükler: Web3D, X3D, Yapay Zeka, Genetik Algoritma, Gezgin Satıcı Problemi

ABSTRACT

CONSTRUCTION AND IMPLEMENTATION OF X3D FRAMEWORK FOR THREE DIMENSIONAL GRAPHICS SOFTWARE USING ARTIFICIAL INTELLIGENCE TECHNIQUES

KALAYCI, Tahir Emre

MSc., Fen Bilimleri Enstitüsü

Supervisor: Asst. Prof. Dr. Aybars UĞUR

July 2006, 114 pages

In this thesis, Researches are made in solving traveling salesman problem using genetic algorithms which play important role as artificial intelligence technique. Web based application has been implemented that tries to calculate shortest tour length of defined points in three dimensional space. Actual performance is measured by using different parameter values for different problem sizes. To examine optimization methods better, visualization tool has been developed using Extensible 3D (X3D). Usability of Web3D technology X3D which is in the process of maturation (also can be thought as XML based VRML) has been tested in the artificial intelligence area.

Implemented tool can plan paths using different genetic algorithm parameters in three dimensional environment. And calculated path can be viewed as three dimensional helicopter simulation.

Keywords: Web3D, X3D, Artificial Intelligence, Genetic algorithms, Travelling Salesman Problem

TEŞEKKÜR

Tez çalışmam süresince sağladığı destek ve sabrı dolayısıyla değerli danışmanım Sayın Yrd. Doç. Dr. Aybars Uğur'a teşekkürü bir borç bilirim.

Çalışmam sırasında gösterdikleri fedakarlık ve sağladıkları destekten dolayı aileme ve arkadaşlarıma teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

KISALTMALAR

X3D	: Extensible 3D
ISO	: International Organization for Standardization
SAI	: Scene Access Interface
NURBS	: Nonuniform rational B-spline
VRML	: Virtual Reality Modeling Language
JOGL	: Java OpenGL
OpenGL	: Open Graphics Library
WWW	: World Wide Web
MPEG	: Moving Picture Experts Group
UGA	: Uygulama Geliştirme Arayüzü
API	: Application Programming Interface
CAD	: Computer Aided Design
XML	: Extensible Markup Language
GZIP	: GNU Zip
LOD	: Level Of Detail
URL	: Uniform Resource Locator
DTD	: Document Type Definition
JRE	: Java Runtime Environment
ALICE	: Artificial Linguistic Internet Computer Entity
NP	: Nondeterministic Polynomial
HEDOS	: Helikopter Dolaşım Sistemi

1 GİRİŞ

İnternet hızlı bir şekilde büyümektedir. Bu büyümeyle birlikte insanların İnternet üzerindeki etkileşim ve görsellik talepleri artmaktadır. Gelişim incelenecek olursa her şey metinle başlamıştır. Daha sonra metinden daha fazla veri verebilen resimler, bunun üzerine canlandırmalar gelmiştir. Artık üç boyuta geçmek için gereksinimler artmıştır. Üç boyut her zaman iki boyutlu bir resimden daha fazlasını anlatmaktadır. Bir cep telefonunun resmini görmek o telefonu üç boyutlu incelemek kadar etkili değildir.

İnternet üzerinde üç boyut için çalışmalar son yıllarda hızlanmıştır. Standartlaşmanın yavaş yavaş tamamlandığı bu alandaki itici gücü Web3D Birliği sağlamaktadır. X3D ("Extensible 3D") teknolojisini ISO ("International Organization for Standardization") standardı yapan ve belirtimlerini oluşturan birlik, bu teknolojiyi tüm hızıyla geliştirmeye devam etmektedir. Bunun yanı sıra X3D teknolojisinin kullanımını arttırmak ve oluşturulan belirtimleri doğrulamak için X3D-Edit, Xj3D gibi açık kaynak projeleri sürdürmekte, diğer tüm ticari ürün projelerine destek sağlamaktadır.

Yapay zeka günümüzde çoğu yazılımda önemli bir işlevi yerine getirmektedir. En küçük yazılımda bile çok basitte olsa bir yapay zeka tekniği kullanılmaktadır. Bu tekniklerin bazıları yıllardır var olan teknikler olmasına rağmen bazıları da yeni sayılabilir. Bu tezin kapsamında önemli bir problem olan ve bir çok uygulama alanı bulunan gezgin satıcı problemi genetik algoritmalar yardımıyla çözülmektedir. Farklı parametre ve teknikler yardımıyla çözümlerin nasıl farklılaştığı incelenmektedir. Üretilen çözüm daha sonra X3D ve Xj3D yardımıyla görselleştirilmektedir. Problem olarak bir helikopterin belli hedefleri dolaşması ele alınmaktadır. Problemin ortaya çıkması kurtarma görevi yapan helikopterlerin en verimli şekilde kurtarılacak varlıklara nasıl erişebileceği konusuna dayanmaktadır.

Tezin 2. bölümünde üç boyutlu grafik yazılımlarından bahsedilmiş, yaygın kullanılanlardan bazıları tanıtılmıştır.

3. Bölümde Web3D birliği tarafından standart olarak sunulan X3D tanıtılmış, kavramları, kullanımı hakkında belirtimlerden yararlanılarak

bilgilendirme ve X3D teknolojisini dışsal uygulamalara bağlamaya yarayan SAI ("Scene Access Interface") teknolojisi tanıtıldıktan sonra, Java programlama diline bağlamak için geliştirilmiş olan Xj3D Araç kutusu anlatılmıştır. Bu araç kutusunun uygulamalarda kullanımı örneklenmiştir.

4. Bölümde Yapay Zeka hakkında biraz bilgi verilip Genetik Algoritma kavramları anlatılmıştır. Bu anlatım örneklerle ve resimlerle güçlendirilmiştir.

5. Bölümde İnternet üzerinde üç boyut, genetik algoritmalar ve Gezgin Satıcı Problemi ile ilgili olarak önceden yapılmış olan çalışmaların özeti araştırmak isteyenler için sunulmuştur.

6. Bölümde Helikopter Dolaşım Sistemi olarak adlandırılan ve Gezgin Satıcı Problemini üç boyutlu engelsiz bir ortamda kullanıcının seçimlerine dayanarak genetik algoritmayla çözen yazılım sisteminin tasarımsal ve gerçekleştirimsel tanıtımı yapılmıştır. Sistemin sınıf diyagramları ve algoritmaları anlatılmıştır.

7. Bölümde yazılımı kullanmak isteyenlerin nasıl işleteceği hakkında ve yazılımın gereksinimleri hakkında bilgilendirme yapıldıktan sonra yazılımın farklı işletim parametrelerine göre değerlendirilmesi yapılmıştır.

8. bölümde, yapay zeka ve internet tabanlı bilgisayar grafikleri bileşenlerinin ortak yer aldığı bu projenin yazılım geliştirme sürecinden elde edilen sonuçlar açıklanmıştır.

2 ÜÇ BOYUTLU GRAFİK YAZILIMLARI

Gerçek dünyanın üç boyutlu olması, bilgisayarlarda üç boyut gereksinimini arttırmaktadır. Üç boyutlu görüntüler daha fazla ilgi çekmekte ve görselleştirmeyi gerçeğe en yakın şekilde sağlamaktadır. Günümüzdeki birçok modelleme programı, 3B (Üç Boyutlu) modellere etkileşimli olarak herhangi bir eksen etrafında döndürülerek kolaylıkla bakabilme ve ayrıca modelin hareketli görüntülerini elde edebilme imkanı sunmaktadır. İki boyutlu grafikler bu açıdan bakıldığında çok yetersiz kalmaktadır.

2.1 3B Paket Programlar

Modelleme, benzeştirim, canlandırma ve tasarım işlemlerini hızlandırmak için paket programlar vardır. Bu paket programlar mühendislikten, eğlence sektörüne kadar bir çok alanda kullanılmaktadır. Bu programların kullanılması üretim sürecini kolaylaştırmakta ve hızlandırmaktadır.

En çok kullanılan paket programlar 3DS Max, Autocad, Blender olarak sayılabilir.

3ds Max: Autodesk tarafından geliştirilen bir 3B modelleme programıdır. MSDOS ortamında çalışan 3D Studio programının devamı olan 3ds Max'in son sürümü, 2006 yılının Mart ayında çıkan 3ds Max 8'dir.

Gelişmiş eklenti desteği ve kolay kullanımı ile 3ds Max, 3D modelleme programları arasında en yaygın kullanıma sahip uygulamalardan biridir. Gelişmiş karakter modelleme özellikleri ile oyun geliştiricilerinin gözdesi haline gelmiştir. Film özel efektleri, mimari sunumlar ve endüstriyel tasarım sunumları gibi alanlarda da yaygın olarak kullanılmaktadır.

3ds Max, parçacık sistemleri, karakter modelleme araçları, hareket yakalama araçları ve gelişmiş denetçiler gibi özellikleriyle tek bir pakette çok sayıda özelliği sunmaktadır. Ayrıca *MAXScript* adında tümleşik bir betik dili vardır.

3ds Max çok sayıda temel objeyi hazır olarak sunar. Mimari tasarımlar için de duvar, kapı, pencere ve merdiven gibi bileşenleri ölçülerini kolayca değiştirerek projeye eklemek mümkündür. 3ds Max ayrıca çokgensel modelleme, NURBS ("Nonuniform rational B-spline") modelleme, yüzey modelleme gibi teknikleri destekler.

3ds Max'in animasyon kontrolleri ile objelerin tüm özellikleri, materyaller, kameralar, ışıklar ve çevre özellikleri zaman içinde değiştirilebilir ve *"Curves Editor"* ile tüm bu özellikler üzerinde tam bir kontrol sağlanabilir. Değişken grafiklerinin Bezier eğrileriyle kontrol edilebildiği bu editör ile karmaşık animasyonların üstesinden gelmek mümkündür.

3ds Max, animasyon için klasik anahtar kare yöntemini kullanır. Zaman doğrusu içinde farklı noktalarda verilen değerler arası geçişi otomatik olarak yapar ve *"Curves Editor"* ile bu geçişlere ince ayarlar yapılmasına olanak verir.

Ters Kinematik çözümleyicisi ile birbirleriyle bağlantılı hareket eden objeler arası ilişkiler kolayca çözümlenir ve kare anahtarlama yöntemi ile kompleks mekanizmaların animasyonu yapılabilir.Ayrıca, pozisyon, bakış, yüzey, bağlanma, tutunma ve yönelme kısıtlayıcılarıyla gelişmiş animasyonlar yapılabilir.

Animasyon için kullanılabilecek diğer özellikleriyse uzay saptırıcıları ve niteleyicilerdir. Uzay saptırıcıları kendilerine bağlanan objelere, bükme, patlatma, rüzgar ve yerçekimi gibi etkileri uygularlar. Niteleyiciler ise modellemede kullanıldıkları gibi animasyon için de objeleri zaman içinde değiştirmede kullanılabilirler.

3ds Max'in en güçlü özelliklerinden biri de *Havoc* tarafından yazılan ve pek çok oyunda kullanılan ünlü fizik motoru *reactor*'dür. *reactor* ile yerçekimi etkisiyle düşme, esneme, sıçrama gibi fizik benzetimleri yapılabilir.

3ds Max'in en zayıf yönü tümleşik imge oluşturma motorunun Maya, Cinema4D, Lightwave ve SoftImage gibi programlara göre daha kötü sonuçlar vermesiydi. Autodesk bu sorunun üstesinden gelmek için daha önce ayrı bir ürün olarak satılan *"Mental Ray"*'i 3ds Max'e dahil etti. 6. sürümden itibaren tüm 3ds Max sürümleriyle birlikte piyasadaki

en güçlü imge oluşturma motorlarından biri olan “Mental Ray”'i kullanmak mümkün. Ayrıca, 3ds Max ile 3. parti imge oluşturma motorları da kullanılabiliyor, bunlardan bazıları; VRay Renderer, Maxwell Renderer, Brasil r/s ve finalRender'dır (Vikipedi, 2006).

Blender: 3B Modelleme, canlandırma, imge oluşturma, üretim sonrası, etkileşimli yaratım ve oynatım için açık kaynak bir yazılımdır. En son sürümleri ticari yazılımlarla yarışabilecek düzeye gelmiştir. Oldukça küçük boyutta bir kurulum dosyasına sahip olup bir çok platformda sorunsuz çalışabilmektedir. Yetenekleri aşağıda listelenmiştir:

- Bir çok farklı geometri ilkeli desteğine,
- YafRay açık kaynak ışın izleme ile bütünleştirilmiş içsel imge oluşturma yeteneğine,
- Bir çok yeteneği olan canlandırma araçlarına,
- Farklı işlevler için Python betik desteğine,
- Doğrusal olmayan görüntü değiştirme ve birleştirme yeteneğine,
- Gerçek zamanlı ve oyun uygulamalarında kullanılmak üzere etkileşim desteği sunan Game Blender alt projesine,

sahiptir (Wikipedia, 2006).

AutoCAD: Tüm dünyada başta mühendisler ve mimarlar tarafından kullanılan, dünyaca tanınan yazılım firması Autodesk tarafından hazırlanan, bilgisayar destekli çizim-tasarım yazılımıdır. Gerek kullanım ve öğrenim kolaylığı gerekse kullanıcı kitlesinin çok geniş olması tüm dünyada tartışmasız kullanılan en yaygın çizim yazılımı olmasını sağlamıştır. Son olarak 2006 sürümü çıkmıştır (Vikipedi, 2006a).

Bu paket programlarda Web3D'de kullanılmak üzere özel biçimlerde dışa alım gerçekleştirilebildiği gibi, özellikle VRML ve X3D dosya dışa alımı için *Flux Development* paket programı mevcuttur.

2.2 Uygulama Geliştirme Arabirimleri

Günümüzde üç boyutlu görüntüler oluşturmak için uygulama geliştirme arabirimlerinden (UGA) yararlanılmaktadır. Kütüphaneleri tarafından sağlanan şekil, özellik ve hizmetlerin kullanımı, üç boyutlu grafik yazılımları geliştirme sürecini kolaylaştırıp hızlandırmaktadır.

Birçoğu halen etkin kullanımda olan grafik UGA'leri, sunuldukları günden itibaren yenilenmektedirler. Bu arabirimler çok farklı amaçlar için kullanılabilmektedir. Günümüzde en çok kullanılan grafik UGA'i olarak OpenGL, DirectX, Java 3D, JOGL sayılabilir. Ayrıca Internet üzerinde üç boyutlu geliştirme için X3D UGA'i kullanılmaktadır.

OpenGL: Etkileşimli ve taşınabilir 2B ve 3B grafik uygulamaları geliştirmek için birincil ortamlardandır. 1992 yılında tanıtılmasından bu yana OpenGL endüstri içerisinde en çok kullanılan ve desteklenen UGA olmuştur. OpenGL geliştiriciler için yüksek görsel kalite ve başarımı sağlamaya çalışmaktadır. Günümüzde başta oyunlar olmak üzere endüstrinin bir çok alanında başarıyla kullanılmaktadır.

Geliştiriciler için aşağıdaki avantajları sağlamaktadır:

- Endüstri için bir standarttır.
- Yedi yıldır oluşmuş bir belirtimi ve bu belirtime uygun olarak kararlı bir yapısı vardır.
- Güvenilir ve taşınabilir bir yapıdadır.
- Sürekli evrimleşmektedir, donanım gelişmelerine ayak uydurmaktadır.
- Her türlü donanıma ve ortama ölçeklenebilirdir.
- Kullanımı kolaydır.
- İyi belgelenmiştir. (OpenGL, 2006)

Direct3D: Microsoft tarafından geliştirilmiştir. Sanal olarak tüm 3B hızlandırıcılar tarafından desteklenmektedir. Ancak sadece Microsoft Windows işletim sistemleri ve Xbox platformlarına özgü bir UGA'dir.

Java 3D: Sahne çizgesi tabanlı bir UGA'dir. Java platformu için geliştirilmiştir. OpenGL veya Direct3D üzerinde çalışan daha yüksek seviyeli bir arabirimdir. Temel hedefi grafik geliştirmede nesne yönelimli geliştirmenin kullanılabilmesini sağlamaktır. İlginin az olması nedeniyle geliştirmesi durmuş, gönüllüler tarafından tekrar sürdürülmüştür.

Yetenekleri şunlardır:

- Çok iş parçacıklı sahne çizgesi yapısı
- Platform bağımsızlığı
- Görselleştirme ve oyun geliştirmede kullanılabilen genel gerçek zamanlılık
- Farklı imge oluşturmalara destek
- 3B uzaysal ses
- Değişik model içe aktarma desteği (Wikipedia, 2006a)

JOGL: OpenGL'in Java programlama dili tarafından kullanılabilmesi için geliştirilmiştir. Sun Microsystems Oyun Teknoloji tarafından geliştirilmektedir. Java 3D'nin göremediği ilgiyi görmüştür. Onun yerine daha çok kullanılmaya başlamıştır. OpenGL'in Java nesne yönelimli yaklaşımla çok iyi bir şekilde kullanılmasını sağlamaktadır.

2.3 Web3D Teknolojileri

Aşağıda dört tanesi verilen ve X3D'ye alternatif olarak kullanılabilecek bazı yazılımlar vardır. Bu yazılımlar farklı amaçlarla farklı kurumlar tarafından geliştirilmiştir.

- Java 3D : Java'nın doğrudan programlama gücünden yararlanarak İnternet üzerinde üç boyutlu geliştirme yapmak için kullanılmaktadır.
- Cult3D : Bu tasarım ve modelleme aracıyla geliştirilen modeller daha sonra tarayıcıya yüklenen Cult3D eklentisi yardımıyla görüntülenmektedir. (Cult3D , 2006)
- U3D ("Universal 3D"): 3B CAD görselleştirmesi için tanımlanmış olan genişletilebilir bir dosya biçemidir. Henüz

imge oluşturma, aktarım veya iletişim kanalı ve çalışma zamanı konularında belirtimleri yoktur. (ECMA , 2006)

- Internet Space Builder: Web için 3B modeller oluşturmak için kullanılabilen gelişmiş yetenekleri olan bir geliştirme aracıdır. (Parallel , 2006)

3 “EXTENSIBLE 3D” (X3D)

X3D, tüm uygulama ve ağ uygulamalarında gerçek zamanlı 3B verinin iletişimini sağlayan XML tabanlı 3B dosya biçemi açık standardıdır. Mühendislik ve bilimsel görselleştirme, CAD (“Computer Aided Design”) ve Mimari, Tıbbi görselleştirme, eğitim ve benzeştirim, çoklu ortam, eğlence, eğitimsel ve daha fazla alanda kullanılmak üzere zengin yetenekler içerir.

Temel özellikleri şunlardır:

- XML (“Extensible Markup Language”) bütünleşiktir: Web servisleri, dağıtık ağlar, uygulama-içi dosya ve veri transferi, cross-platform
- Bileşenlere dayalıdır: Hafif 3B çalışma zamanı motorunu destekler.
- Genişletilebilirdir: Farklı market ve servisler için işlevlerin bileşenlerle genişletilmesini sağlar.
- Profiller vardır: Uygulamaya özgü genişletmeler ve bileşenler kullanılır.
- Evrimseldir: VRML97 içeriğini X3D içeriği şeklinde güncellemek ve saklamak kolaydır.
- Geniş uygulama desteği: Mobil telefonlardan süper bilgisayarlara
- Gerçek Zamanlı: Grafikler yüksek kalitede, gerçek zamanlı, etkileşimli, ve 3B verinin yanında ses ve müzik içerir.
- İyi tanımlanmış: Uygun, tutarlı ve hatasız gerçekleştirim yapmayı kolaylaştırır.

Desteklediği özellikler ise şu şekilde listelenebilir:

- **3B Grafikler:** Çokgensel geometri, parametrik geometri, sıradüzensel dönüşümler, ışıklandırma, materyaller, çoklu-geçiş/çoklu-evre desen kaplama, piksel ve köşe gölgelendiriciler, donanım hızlandırma

- **2B Grafikler:** Uzaysal metin, 2B (İki Boyutlu) Vektör grafikler, 2B/3B derleme
- **CAD Verisi:** CAD verilerinin sunum için açık biçime çevrilmesi ve etkileşimli ortam
- **Canlandırma:** Zamanlı canlandırmalar için zamanlayıcı ve ara değerleyiciler, insansı canlandırma ve biçim değiştirme
- **Uzaysal ses ve görüntü:** görsel-işitsel ("audiovisual") kaynakların sahnedeki geometrilere bağlanması
- **Kullanıcı Etkileşimi:** Fare tabanlı seçmeler ve taşımalar, klavye girdisi
- **Dolaşım:** Kameralar, 3B sahnede kullanıcı hareketi, çarpışma, yakınlı ve görünürlük bulma
- **Kullanıcı tanımlı nesneler:** Kullanıcı tanımlı nesne tipleri yardımıyla, var olan işlevselliğin genişletilebilmesi
- **Betimleme:** Programlama ve betimleme yardımıyla sahnenin dinamik olarak değiştirilmesi veya yaratılması
- **Ağ:** Ağdaki farklı bileşenlerden derlenerek tek bir 3B sahnenin oluşturulması, WWW ("World Wide Web") üzerindeki diğer nesnelerin bağ yardımıyla kullanılması
- **Fiziksel Benzeştirim:** İnsansı canlandırma, coğrafi-uzaysal veri kümeleri, dağıtık etkileşimli benzeştirim (DIS "Distributed Interactive Simulation") protokolleriyle bütünleştirme

VRML97'in yekpare doğasından dolayı, uygulamalar tüm özellik kümesini içermesini gerektirir. X3D geliştiricilerin, belirtimlerin alt kümelerini (Profiller) desteklemesine olanak sağlar. Bu profiller bileşen ve işlevsellik bloklarından derlenmiştir.

Bileşen tabanlı mimari her biri bireysel olarak desteklenen farklı "profillerin" yaratılmasını destekler. Bileşenler bireysel olarak genişletilebilir veya yeni seviyeler eklenerek değiştirilebilir. Veya yeni bileşenler yeni özellikler tanımlamak için eklenebilir. Bu düzenekle,

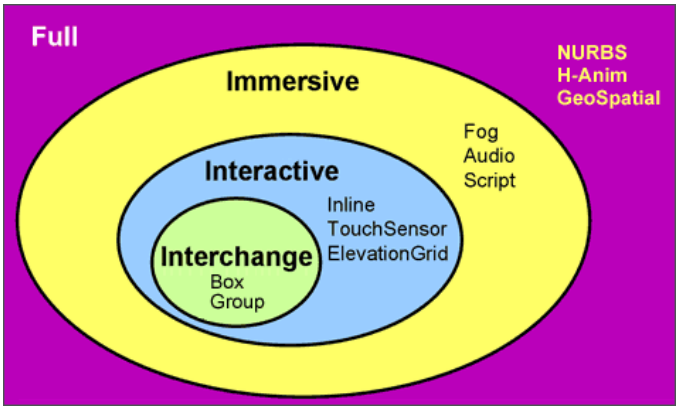
belirtim ilerlemeleri bir alandaki geliştirme tüm belirtimi yavaşlatmayacağı için daha hızlı olur. Daha önemlisi, belirli bir içerik için gereken uygunluk gereksinimleri, karışıklık yaşanmadan, içerik için gereken profillerin, bileşenlerin ve seviyelerin belirtilmesiyle tanımlanır.

X3D'nin desteklediği temel dayanak profiller (Bkz. Şekil 3.1) şu şekildedir:

- **Arasında Değişim ("Interchange"):** Uygulamalar arasında iletişim için gereken temel profildir. Geometri, desen kaplama, temel ışıklandırma ve canlandırmayı destekler.
- **Etkileşimli ("Interactive"):** Kullanıcı dolaşımı ve etkileşimi (PlaneSensor, TouchSensor,..) için algılayıcı düğümlerini ekleyerek 3B ortam ile basit etkileşimi, geliştirilmiş zamanlamayı, ek ışıklandırmayı (SpotLight, PointLight) olanaklı kılar.
- **"Immersive":** Tüm 3B grafik ve etkileşimleri, işitsel destek, çarpışma, sis ve betimlemeyi içerir.
- **Tam ("Full"):** Tanımlı tüm düğümleri içerir. NURBS, H-Anim ve GeoSpatial bileşenlerde dahildir.

Ayrıca ek olarak şu profiller mevcuttur:

- **MPEG-4 Etkileşimli ("MPEG-4 Interactive"):** Etkileşimli profilin, yayın, el cihazları ve mobil telefonlar için olan ufak bir örneğidir.
- **CDF (CAD Distillation Format):** Geliştirme aşamasında olan CAD verisinin sunum ve etkileşimli ortam için açık biçime çevrilmesini olanaklı kılma çalışmasıdır.



Şekil 3.1: X3D Profilleri (Web3d, 2006)

X3D, VRML'den daha olgun bir standart olduğundan dolayı geliştiriciler beklentilerine ulaşırlar. Buna rağmen bazı insanlar, VRML ("Virtual Reality Markup Language") ile geliştirmeye devam etmek yerine XML gerektiren X3D'nin daha iyi bir geliştirme seçeneği olduğunu sormaktadırlar. Bunun için X3D geliştiricileri aşağıdaki on maddeyi sunmaktadır: (Web3d, 2004)

- **VRML Uyumlu:** Hala **"**Klasik VRML" kodlama içermesi nedeniyle betimleme içermeyen çoğu VRML sahnesi küçük değişikliklerle rahatlıkla gösterilebilir. Hiç bir teknoloji kaybolmamıştır, X3D şeklinde evrimleşmiştir. X3D için VRML ile uyumluluğunu koruması için çok fazla güç harcanmıştır. Buna rağmen oynatıcılar arasındaki içsel olmayan ortam uyumsuzluk problemleri de giderilmiştir**.**
- **XML kodlama sayesinde diğer uygulamalarla rahatlıkla bütünleşme:** XML her türlü veri tabanlarında bilgi saklamak için önkoşul olmaktadır. XML kodlamaya sahip olmak, yönetimi, kontrolü, doğrulamayı ve veri değişimini kolaylaştırmaktadır. X3D'nin XML kodlaması bu dünyada uygun şekilde yer almaktadır.
- **X3D sahne ve ortamların farklı oynatıcılarda önceden kestirilebilir çalışması:** VRML' deki temel sorunların başında tüm uyumlu oynatıcılarda aynı şekilde davranan VRML ortamlarının geliştirilmesinin zorluğuydu. Bunun nedeni VRML standardındaki

VRML davranışının yeterli belirtiminin eksikliğiydi. X3D davranışının tüm uyumlu oynatıcılarda aynı olması için sıkı bir güç harcanıp yeterli belirtimler sağlanmıştır.

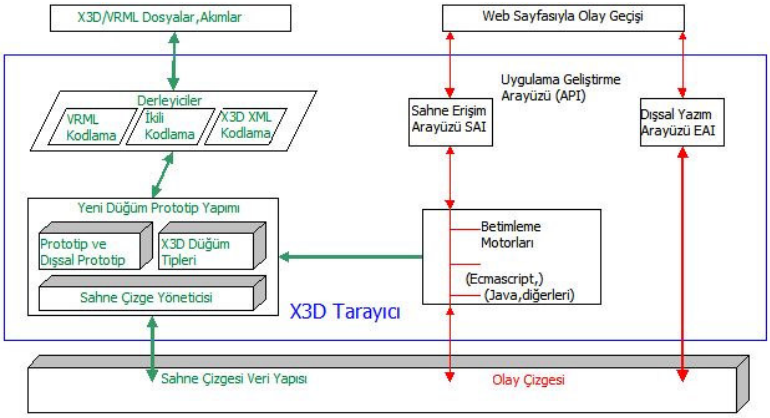
- **X3D bileşenlere dayanır:** Bu sayede X3D farklı marketler için farklı profiller oluşturulmasını ve endüstri tarafından geliştirilen yeni teknolojilerinin daha temiz bir şekilde tanıtılmasını sağlar .
- **Herhangi bir oynatıcı için yazış tutarlı ve kolaydır:** X3D "Scene Authoring Interface" içsel ve dışsal tüm betik dilleri için tutarlı işlevsellik sağlar. Bu VRML için, Java ve ECMAScript farklı programlama modellerine sahip olması dolayısıyla doğru değildir. X3D SAI sayesinde soyut servisler tanımlayıp bunlara herhangi betik/programlama dillerinin eşlenmesiyle tutarlı çalışan ortamlar gerçekleştirilir. Java ve ECMAScript dil bağlantıları sağlanmıştır. Bu X3D yazışı çok kolaylaştırır.
- **X3D daha fazla özellik zenginidir:** VRML'de eksikliği hissedilen geniş sayıda özellik X3D mimarisine eklenmiş ve standart olmuştur. Böylece, sağlayıcıya özgü geçici çözümler önlenmiştir. X3D'yi VRML3 olarak düşünebiliriz.
- **X3D sürekli geliştirilip güncellenmektedir:** X3D gün geçtikçe daha fazla özellikle güçlendirilmektedir. Bu şekilde güncellenmesi yapısı dolayısıyla daha kolaydır. Sürekli olarak işlevsellik olarak büyümektedir.
- **X3D uygulamaları güvenilir ve kestirilebilir olarak onaylanabilir:** Web3D birliği tarafından X3D uyumlu yazılımların servis notlarını belirlemek için X3D Uyumluluk programı geliştirilmektedir. Bu şekilde yazış ve oynatma (tarayıcılar/oynatıcılar) uygulamaları güvenilir ve kestirilebilir olacaktır.
- **X3D açık kaynak uyumluluk uygulaması geliştirici kaynağı olarak mevcuttur:** Neredeyse X3D'nin tümü açık kaynak gerçekleştirim olarak (Xj3D, Flux, FreeWRL) mevcuttur ve mülki uyumlu tarayıcılar (BS Contact gibi) geliştirilmektedir. VRML sahnelerinin tersine X3D sahneleri uyumlu tüm onaylı oynatıcılarda tutarlı bir şekilde oynayacaktır.

X3D İkili Kodlama şifreleme (güvenlik) ve sıkıştırma (hız) teklif ediyor. Geliştirilmekte olan Sıkıştırılmış İkili Kodlama, X3D ortamlarının model güvenliği için şifreleme ve çok yüksek sıkıştırmaya(VRML'in GZIP'inden daha yüksek oranda) olanak sağlar. Sahne ayrıştırma ve yükleme %300-%500'e varan hız kazanıyor. Tüm tarayıcılar tüm kodlamaları rahatça destekleyebilir. Kodlamalar arasındaki tek fark farklı ayrıştırıcılar kullanmasından ibarettir. Böylece, kodlamalar sağlanan dünyanın içerisine karıştırılabilir ve tarayıcı tüm kullanılan kodlamaları destekleyebilir. Şu anki X3D geliştiricileri tüm kodlamaları desteklemeyi planlamaktadır.

Kavramsal olarak, her X3D uygulaması, ağ üzerinden yüklenebilen ve farklı düzenekler yardımıyla dinamik olarak değiştirilebilen grafik ve işitsel nesneler içeren 3B zaman-tabanlı uzaydır (Bkz. Şekil 3.2). X3D'nin anlam bilimi; zaman-tabanlı soyut işlevsel davranışları, etkileşimli 3B, çoklu ortam bilgisini tanımlar. X3D fiziksel cihazları veya diğer gerçekleştirim-bağımlı kavramları (örneğin ekran çözünürlüğü ve giriş cihazları) tanımlamaz. X3D geniş çeşitte cihazlar ve uygulamalar için tasarlanmıştır, ayrıca işlevselliğin gerçekleştiriminde ve yorumlanmasında geniş ölçeklidir. Örneğin, X3D fare veya 2B görüntüleme cihazının olduğunu var saymaz.

Her X3D uygulaması:

- Dolaylı olarak tanımlanmış tüm nesneler ve uygulama tarafından içerilen tüm nesneler için dünya koordinat uzayını kurar.
- Açıkça 2B, 3B ve çoklu ortam nesne kümelerini bileştirir.
- Diğer dosya ve uygulamalara bağlar tanımlayabilir.
- Nesne davranışlarını tanımlayabilir.
- Programlama ve betik dilleri yardımıyla dışsal modüllere veya uygulamalara bağlanabilir.



Şekil 3.2: X3D Mimarisi

3.1 Oluşturma ve Oynatma

3.1.1 X3D Tarayıcılar

X3D dosyalarının yorumlanması, çalıştırılması ve sunulması *tarayıcı* olarak bilinen düzenek kullanılarak yapılır. Tarayıcı sahne çizgesindeki şekil ve sesleri gösterir. Bu sunum *sanal dünya* olarak bilinir ve tarayıcıda *kullanıcı* denilen insan veya mekanik bir varlık tarafından dolaşılır. Dünya belli bir noktadan deneyim edilerek gösterilir; dünyadaki bu nokta ve yönelim *izleyici* olarak bilinir. Tarayıcı, kullanıcının izleyiciyi sanal dünyada hareket ettirebileceği farklı dolaşım modelleri sağlayabilir.

Dolaşıma ek olarak, tarayıcı kullanıcının dünyayla sahne çizge sıradüzenindeki algılayıcı düğümleri yardımıyla etkileşebileceği sınırlı düzenekler sağlar. Algılayıcılar kullanıcının etkileşimine sahnedeki geometrik nesnelerle, kullanıcının dünyadaki hareketiyle veya zaman geçişiyle karşılık verirler. Ek olarak X3D Sahne Erişim Arayüzü (SAI "Scene Access Interface") kullanıcı girdisini almak ve o anki bakış noktasını almak ve ayarlamak için düzenekler sağlar. Dolaşım yeteneklerini sağlamak için, izleyici kullanıcıya dolaşabilme yeteneği

sağlamak için SAI kullanabilir. Ayrıca, yazarlar betik ve programlama dillerini SAI'yi kendi dolaşım algoritmalarına bağlamak için kullanabilirler. Diğer profiller dolaşım yeteneklerini göstericinin gereksinimi olarak tanımlayabilirler, bu tip göstericilerin gerçekleştirimi SAI kullanarak yapılır.

X3D dünyasındaki geometrik nesnelerin görsel sunumu, ışığın fiziksel özelliklerini benzetmek için tasarlanmış kavramsal modeli izler. X3D ışıklandırma modeli, gösterilen renklerin üretilmesi için dünyadaki görünüm özellikleri ve ışıkların nasıl birleştirildiğini tanımlar.

3.1.2 X3D üreticiler

Üretici, bir insan veya bilgisayarlaştırılmış X3D dosya yaratıcısıdır. X3D dosyasının doğruluğu ve içerisinde başvurulan destek kaynaklarının (Örneğin resimler, ses klipleri, diğer X3D dosyaları) varlığının garantisi üreticinin sorumluluğundadır. Ayrıca, X3D dosyasında sunulan işlevselliğin profil,bileşen ve seviye bilgilerinin dosyanın başlık cümlelerinde doğru bir şekilde belirtilmiş olması üreticinin sorumluluğundadır.

3.1.3 X3D yükleyiciler

Yükleyici, X3D içeriğinin yüklenmesinden sorumludur, ancak herhangi bir çalışma zamanı işlemi uygulamaz. Geometri sunulur, zaman çalıştırılmaz, buna karşın yükleyici dokuların ve diğer uzak tanımlı içeriğin yüklenmesi konusunda özgürdür. Sıfır zamanlı yükleyiciler genellikle var olan X3D içeriğini çalışma zamanı yönünü değerlendirmeden yapmak ve değiştirmek amaçlı modelleme araçlarında bulunur.

Yükleyicinin ikinci bir biçimi, dosyaları yükleyebilir ve içeriğin çalışma zamanı işletilmesine olanak sağlar, ama bu şekliyle daha geniş kullanıcı arayüzü ve 3B grafik imge oluşturma makinesinin bir parçasıdır. Örneğin bu tip yükleyiciler oyun ortamındaki ağaçlar gibi bireysel modellerin yüklenmesinde kullanılabilir. Ancak X3D içeriğinin

çalışma zamanı değerlendirmesi dışsal uygulamaya bağımlı olup, tarayıcıda olduğu gibi kendi içerisinde yoktur.

3.2 Sahne Çizgesi

X3D çalışma-zamanı ortamının temel birimi *Sahne Çizgesi*dir. Bu yapı sistemdeki tüm nesneleri ve onların ilişkilerini içerir. İlişkiler sahne çizgesinin bir çok ekseni boyunca içerilir. *Dönüşüm Sıradüzeni* imgeleri oluşturulan nesnelerin uzaysal ilişkisini tanımlar. *Davranış Çizgesi* sahalar arasındaki bağlantılar ve sistem içerisindeki olayların akışını tanımlar.

Bir X3D dosyası sıfır veya daha fazla kök düğüm içerir. X3D dosyasının kök düğümleri, başka düğüm veya PROTO deyimlerinde içerilmeyen düğüm deyimleri veya USE deyimleri yardımıyla tanımlanmış düğümlerdir. Kök düğümlerinin hepsi çocuk düğümleri olarak tanımlanacaktır.

X3D sahne çizgesi yönlü çevrimsiz çizgedir. Düğümler sıradüzene katılan bir veya daha fazla çocuk düğüm içeren belirli sahalar içerebilir. Bunlar düğümleri içerir (Veya düğümlerin örnekleri). Düğümlerin bu sıradüzenine *sahne çizgesi* denilmektedir. Çizgedeki A'dan B'ye olan her yay, düğüm A'nın içerisinde direkt olarak B düğümünü değer olarak tutan bir saha olduğu anlamına gelir. Bir düğümün *torunları* onun sahalarındaki düğümler ve bu düğümlerin torunlarıdır. Bir düğümün *ataları* bu düğümün torun olduğu düğümlerdir.

Dönüşüm sıradüzeni, sanal dünyada bir veya daha fazla özel konumlar içeren tüm kök düğüm ve kök düğüm torunlarını içerir. X3D, ata koordinat sistemlerinden dönüşümler olarak tanımlanan *yerel koordinat sistemleri* fikrini içerir. Kök düğümlerin gösterildiği koordinat sistemi *dünya koordinat sistemi* olarak adlandırılır.X3D tarayıcının görevi X3D dosyasını kullanıcıya sunmaktır; bunu dönüşüm sıradüzenini kullanıcıya sunarak gerçekleştirir. Dönüşüm sıradüzeni, sanal dünyanın doğrudan algılanabilir parçalarını tanımlar.

Algılayıcılar ve ortam düğümleri gibi bazı düğümler, sahne çizgesinde bulundukları halde dönüşüm sıradüzeni tarafından

etkilenmezler. Bu düğümlere CoordinateInterpolator, Script, TimeSensor ve WorldInfo örnek olarak verilebilir.

Switch veya LOD gibi bazı düğümler, imge oluşturma sırasında dolaşılan çocuk listeleri içerir. Ancak, sahne konumunun hesaplanması amacıyla, imge oluşturma sırasında dolaşılsın, dolaşılmasın tüm çocuklar dönüşüm sıradüzeninin parçası olarak dikkate alınır. Örneğin, whichChoice seçeneği -1 olan (hiç bir çocuğu imge oluşturma sırasında dolaşılmayacak anlamında) bir Switch düğümünün çocuğu olan bir Viewpoint düğümü sahnedeki konumunu belirlemek için Switch düğümünün yerel koordinat uzayını kullanmayı sürdürecektir.

Dönüşüm sıradüzeni yönlü çevrimsiz çizge olmalıdır; dönüşüm sıradüzenindeki kendi kendisinin atası olan bir düğüm hatalı olarak dikkate alınmalı ve ihmal edilmelidir. Aşağıdaki örnek kendi kendisini ata olarak kullanan bir sahne çizgesi örneğidir:

```
<Transform DEF="T">
<Transform USE="T" />
</Transform>
```

X3D'nin olay modeli, sahalar arasında rotalar (Route) tanımlanmasına olanak sağlar ve bu bağlantılar arasında olay yayılması için bir model tanımlar. Davranış çizgesi bu saha bağlantılarının derlemesidir. Bağlantıların yeniden yönlendirilmesi, eklenmesi ve kırılması ile dinamik olarak değiştirilebilir. Olaylar sistem içerisine enjekte edilir ve davranış çizgesi boyunca iyi tanımlanmış bir sırada yayılır.

Sahalar eğer bileşen bu kurala aykırı bir genişletme içermiyorsa sadece aynı veri tipine sahip diğer sahalara yönlendirilebilir.

3.3 Çalışma Zamanı Ortamı

X3D çalışma zamanı ortamı sahne çizgesinin güncel durumunu korur, sahneyi istendiği şekil oluşturur, farklı kaynaklardan (Algılayıcılar) girdiler alır ve davranış sisteminden gelen bilgilere göre sahne çizgesini değiştirir. Ayrıca içinde var olan ve kullanıcı tanımlı nesne ve programatik betiklerde dahil olmak üzere nesnelerin yaşam

döngüsünü yönetir. Olayların işlenmesini, birincil anlamıyla X3D'de davranışların üretilmesini düzenler. Ek olarak X3D tarayıcı ve ev sahibi uygulama arasındaki dosya dağıtımı, üst bağlama, sayfa bütünleşmesi ve dışsal programatik erişim için birlikte çalışabilirliği yönetir.

Çalışma zamanı ortamı nesneleri yönetir. X3D çalışma zamanı içerisinde faydalı işlevler içeren içinde *yerleşik nesne* tiplerini destekler. Veri yapılarını temsil eden *SFVec3f* 3B vektör değeri gibi, geometri (örneğin *Cylinder*) gibi ve düğümler arası bağlantıları gösteren *Route* gibi yerleşik nesneler vardır. Her düğüm sıfır yada daha fazla veri değerleri için depolama tanımları içeren *sahalar* ve/ya nesneler arası mesaj iletimi için sıfır ya da daha fazla *olaylar* içerir. Düğümler dosyada tanımlanarak veya çalışma zamanında yordamsal kodları kullanarak ilklenirler. Yazarlar prototip işleyişini kullanarak yeni düğüm tipleri oluşturabilir. Bu düğümler çalışma zamanının bir parçası olup yerleşik düğümlerle aynı şekilde davranırlar. Yeni düğümler prototip tanımını içeren dosyanın içerilmesi, farklı bir konumdaki prototip tanımlamasına dışsal referans içerilmesi veya çalışma zamanı tarafından sağlanan yerli prototip tanımlamasıyla yaratılabilir. PROTO'lar sadece yeni düğümler yaratmak için kullanılabilir, saha veya rotalar yaratılamaz.

Olaylar X3D çalışma zamanında davranışların üretilmesi için birincil yöntemdir. Olaylar X3D içerisinde zaman tabanlı canlandırmaların işletilmesi, nesne seçiminin kotarılması, kullanıcı hareket ve çarpışmasının ortaya çıkarılması ve sahne çizge sıradüzeninin değiştirilmesi için kullanılmaktadır. Çalışma zamanı ortamı olayların sistem boyunca yayılımını ve iyi tanımlanmış kurallara dayanarak değerlendirilmesini yönetir.

X3D içeriğinin yazarı sahnelerin yaratılmasını ve yönetilmesini, imge oluşturma ve davranış, ve ortam varlıklarının yüklenmesini denetleyebilir. Genişletmelerin –X3D'de veya dışsal bir dilde yazılmış olabilir- yüklenmesi ve birleştirilmesi de denetlenebilir. İçerik tanımlı genişletmelerin yapım yeteneği prototipleme işleyişini destekleyen profillerle sağlanmaktadır.

3.4 Nesne Modeli

X3D sistemi nesne olarak adlandırılan soyut bağımsız varlıklardan oluşmuştur. Veri saklama ve veri üzerindeki işlemler gibi hafif sıklet kavramları temsil eden nesnelere *saha* adı verilmekte olup *X3DField* nesnesinden türetilmektedir. Daha eksiksiz, uzaysal veya zamansal işleme kavramlarını temsil eden nesnelere *düğüm* adı verilmekte olup *X3DNode* nesnesinde türetilmektedir. Düğümler bir veya daha fazla veri değerleri içeren veya olay alıp gönderen saha içerir.

Bazı düğümler ortak özellik veya işlevsellikleri temsil eden *arayüzlerden* kalıtılarak ek işlevsellikler gerçekleştirirler. Örnek olarak görsel nesneler için sınır kutuları ve gruplama düğümleri veya üst bilgiyi temsil eden özel nesne belirtimleri verilebilir. Ek olarak, X3D ROUTE,PROTO tanımları, Bileşen/Profil bilgisi ve dünya üst bilgisi gibi saha veya düğümlerde saklanmayan sahne çizgesi bilgisine erişim için nesne tipleri tanımlar.

Bir saha bir tipte tek bir değer veya bu tipten bir dizi içerebilir. Tek değer içeren ilgili tipteki bir saha *SF* (Örneğin *SFVec3f*) *ön ekiyle*, dizi içeren tip ise *MF* (Örneğin *MFVec3f*) ön ekiyle gösterilmektedir. Bir saha bir veya daha çok düğüme *SFNode* tipi kullanarak referans içerebilir.

Her nesne aşağıdaki ortak özellikleri içermektedir:

- **Tip İsmi.** SFVec3f, MFColor, SFFloat, Group, Background, veya SpotLight örneklerdir.
- **Bir Gerçekleştirim.** Her nesnenin gerçekleştirimi özellik değerlerindeki değişimlere nasıl tepki verir, hangi diğer özellik değerleri bu değişimlere göre değişir, çalışma zamanı ortamı bu değişimlerle nasıl etkilenir tanımlar.

X3DNode'tan türetilen bir nesne aşağıdaki ek özelliklere sahiptir:

1. **Sıfır veya daha fazla saha değeri.** Saha değerleri düğüm veya sahalarla birlikte X3D dosyasında saklanır, ve sanal dünyanın durumunu kodlar.

2. **Sıfır veya daha fazla alıp gönderebileceği olaylar.** Her düğüm sahalarına düğümün durumunu değiştirecek olaylar alabilir. Her düğüm sahalarında olay üretip düğümün durumunda bu olayları bildirebilir. Bir düğümde üretilen olaylar diğer düğümlerin sahalarına bağlanıp yayılabilir. Bu dosyada ROUTE cümlesiyle veya SAI hizmet referansıyla yapılır.
3. **Bir İsim.** Düğümler dosyada DEF cümlesi kullanılarak veya çalışma zamanında SAI hizmet çağrımı kullanılarak adlandırılabilir. Bu isim belli bir düğüm örneğine referans edilecek cümlelerde kullanılır. Ayrıca sahne sıradüzeninde belli isimdeki bir düğümü bulmak için de kullanılabilir.

Düğüm gerçekleştirimleri iki kaynaktan gelebilir, yerleşik düğümler ve prototipler. Yerleşik düğümler, yazarın kullanması için uygulanabilir profil ve/ya bileşen tanımlarıyla belirtilmiş düğümlerdir. Farklı bileşenler, farklı yerleşik düğüm kümeleri tanımlar.

Ek olarak, X3D prototip kullanılarak içerik genişletmeyi destekler. Prototipler yazar tarafından PROTO veya EXTERNPROTO cümleleri kullanılarak yaratılmış nesnelerdir. Bu nesneler sahne çizgesindeki düğümleri tanımlamak için kullanılan tanıtıcı yazımla yazılır. Sadece yüklendikleri oturum yaşamı boyunca kullanılabilir yeni düğümleri sisteme eklerler. Bazı profiller bu genişletmeler için destek içermeyebilir.

Hem prototipler hem de yerleşik düğümler, aynı düzenekler kullanılarak örneklenmektedir. Bir nesne bildirilerek veya çalışma zamanında SAI hizmetleriyle ilklenebilir. Tüm prototipler *X3DPrototypeInstance* ana düğüm tipinden kaşıtılır.

3.4.1 Saha anlam bilimi

Sahalar düğümlerin kalıcı durumlarını ve olay biçiminde gönderip alabilecekleri değerleri tanımlar. X3D dört tip düğüm sahalarına *erişim* tipi destekler:

- *initializeOnly* erişim, içerikte sahanın ilk değeri olmasına olanak sağlar fakat ileride bu değer değiştirilemez.

- *inputOnly* erişim, düğüm sahanın değerinin değişmesine yönelik olayları alabilir, fakat sahanın değerinin okunması olanağı yoktur.
- *outputOnly* erişim, düğüm sahanın değeri değiştiği zaman olay gönderebilir, fakat sahanın değerinin değiştirilmesine olanak vermez.
- *inputOutput* erişim, sahaya tam erişim vardır: içerik tarafından saha için bir ilk değer sağlanabilir, düğüm sahanın değiştirilmesine yönelik olayları alabilir ve düğüm sahanın değeri değişince olay gönderebilir.

inputOutput saha inputOnly saha gibi olay alabilir, outputOnly saha gibi olaylar üretebilir ve initializeOnly saha gibi X3D dosyada değeri saklanabilir. *zzz* olarak adlandırılmış olan bir inputOutput saha *'set_zzz'* şeklinde başvurularak inputOnly saha olarak ele alınır, ve '*zzz_changed*' şeklinde başvurularak outputOnly saha olarak ele alınır. Genellikle inputOutput ve inputOnly erişime sahip sahalar *girdi* sahaları, inputOutput ve outputOnly erişime sahip sahalar *çıktı* sahaları olarak ve bu sahaların aldığı ve gönderdiği olaylar *girdi olayları* ve *çıktı olayları* olarak adlandırılır.

inputOutput bir sahanın ilk değeri X3D dosyasındaki değer veya eğer dosyada belirtilmemişse içerildiği düğüm tarafından tanımlanan var sayılan değerdir. inputOutput saha bir olay aldığı zaman, aynı değerde ve zamanlı bir olay üretmelidir. Aşağıdaki kaynaklar, sıralandıkları şekilde inputOutput sahanın ilk değerinin belirlenmesinde kullanılacaktır:

- İlklemede kullanıcı tanımlı değer
- Düğüm veya prototip tanımında belirtilmiş olan var sayılan değer

initializeOnly sahalar, inputOutput sahalar, outputOnly sahalar ve yerleşik düğümler için inputOnly sahaların isimlendirilmesi şu şekildedir:

- Birden fazla kelime içeren tüm isimler küçük harf ile başlamalı ve diğer tüm kelimelerin ilk harfi büyük harf olmalıdır (Ör. *addChildren*) , istisnai olarak aşağıdaki *set_* ve *_changed* vardır.

- Tüm inputOnly sahalar "*set_*" ön ekine sahiptir, *addChildren* ve *removeChildren* istisnadır.
- SFTime tipinde bazı inputOnly sahalar ve outputOnly sahalar "*set_*" ön eki veya "*_changed*" sonekini kullanmazlar.
- Diğer tüm outputOnly sahalar "*_changed*" sonekine sahiptir, SFBool tipindeki outputOnly sahalar istisnadır. Boolean outputOnly sahalar daha iyi okunabilirlik için "*is*" (Ör. *isFoo*) ön ekiyle başlarlar.

3.4.2 Ara birim Sıradüzeni

Çoğu nesne tipi diğer nesnelerin işlevselliği ve ara birimlerden türer. Bu tipler süper tip olarak adlandırılır, ve nesneye bir süper tipten *türetilmiş* denilir. Aynı şekilde süper tipler yeteneklerini diğer nesne tiplerinden türetebilir, bu tüm nesnelerin türediği temel bir kaç sınıflı bir zincir oluşturur. Bu türeme çizelgesine ara birim sıradüzeni adı verilir.

Çoğu nesne tipi, ara birimlerini ve işlevselliklerini sistemdeki diğer nesne tiplerinden sağlarlar. Bu tiplere *süper tip* adı verilir, bu tipten türetilen nesneye de *türetilmiş* olduğu belirtilir. Bu şekilde süper tiplerde başka tiplerden türeyip zincir şeklinde bir yapı oluşabilir. Tüm nesne tipleri arasındaki ilişkiyi tanımlayan çizgeye ara birim *sıradüzeni* adı verilmektedir.

Nesne sıradüzeni hem *soyut* hem de *somut* düğüm tiplerini tanımlar. Soyut ara birimler işlevselliği tanımlar ve diğer ara birim ve/ya düğümler tarafından kalıtılır, ancak hiç bir zaman sahne çizgesinde nesne olarak ilklenmez. Somut düğüm tipleri bir veya daha fazla soyut ara birimden türetilir ve sahne çizgesinde ilklenebilir. Böylece, canlı sahne sadece somut düğüm tiplerinin örneklerinden oluşur.

Çoğu nesnenin türetildiği iki ana tipi *X3DNode* ve *X3DField* tipleridir. X3DNode tipi düğümleri, X3DField tipide bu düğümlerin barındırdığı diğer düğümlere referans ve verileri tutan sahaları tutan tiptir. *X3DNode* tipi *X3DField* düğümünü içerir. *X3DField* tipide *X3DNode* tipine referanslar içerir. Bu şekilde X3D'nin sahne çizge sıradüzenleri oluşturması sağlanır.

```
<Transform translation="1 2 3">
<Shape>
     <Box />
</Shape>
<Group>
     .......
</Group>
</Transform>
```

Yukarıdaki örnekte Transform düğümü, 3 sayıdan oluşan bir vektör saklayan basit bir `translation` sahası içerir. Ayrıca diğer düğümleri tutabilen `children` sahasını içermektedir. Yukarıdaki örnekte bu saha bir Shape ve Group düğümü içermektedir. Bu iki düğüm tipide (Shape ve Group) diğer düğümleri tutabilen sahalar içermektedir.

Türetme yardımıyla tüm nesneler sıkı tiplenmiştir. Yukarıdaki örnekte children sahası *X3DChildNode* tipinden türetilmiş nesneleri barındıran liste içerme kısıtına sahiptir. Shape ve Group nesneleri (dolaylı olarak) bu nesne tipinden türetilmiştir ve böylece children sahası içerisine yerleşebiliyor. Diğer taraftan Shape'in geometry sahası *X3DGeometryNode* tipinden türetilmiş tek bir düğüm içerebilir. Box bu tipten türetilmiş olduğu için geometry sahası içerisine yazılabilmektedir. Ancak Box *X3DChildNode* tipinden türetilmediği için children sahasına, aynı şekilde Group *X3DGeometryNode* tipinden türetilmediği için geometry sahasına yerleştirilemez.

Yukarıdaki örnek, türetmenin başka bir kalitesini de gösterir. Transform *X3DGroupingNode* tipinden türetilmiştir, bu tipin children sahasını kalıtır. Bu Transform'un belirtimini kolaylaştırır, çünkü children sahasının işlevselliğini açıklamaya gerek yoktur. Çünkü *X3DGroupingNode* tipinden türetilmiştir, yazar children sahası içerdiğini ve *X3DGroupingNode* tipinden türetilmiş Group'taki children sahası gibi davrandığını bilmektedir.

3.4.3 Nesneleri Değiştirmek

Rotalar: Bir nesnenin sahalarını değiştirmenin bir çok yolu vardır. Bir X3D dosya biçemini kullanarak, bir yazar düğüm kümesini, sahaların ilk değerlerini ve *Route* adı verilen bu sahalar arasındaki ilişkileri tanımlayabilir. X3D, çalışma zamanında sahaların değerlerinin değiştirilmesi için olay yayımı veya *veri akış* modelini kullanır. Soyut belirtimin yardımıyla düğümün sahalarına gönderilen olaylara tepki davranışı ve hangi durumlarda sahalarından dışarıya olay gönderildiği tanımlanmıştır. Sadece bu olay yayım modelini kullanan çalışma zamanı davranışı içeren sahne yaratılması mümkündür. Örneğin:

```
<Transform DEF='T'>
<Shape><Box/></Shape>
</Transform>
<TimeSensor DEF='TS'
loop='true' cycleInterval='5'/>
<PositionInterpolator DEF='I'
key='0 0.5 1'
keyValue='0 -1 0, 0 1 0, 0 -1 0'/>
<Route fromNode='TS'
fromField='fraction_changed'
toNode='I' toField='set_fraction'/>
<Route fromNode='I'
fromField='value_changed'
toNode='T' toField='set_translation'/>
```

Bu örnek sürekli olarak 5 saniye boyunca devam eden bir kutunun aşağı ve yukarı hareketini içerir. TimeSensor nesnesi *fraction* sahası üzerinden sürekli dışarıya olay fırlatacak şekilde tanımlanmıştır. Bu olay, *cycleInterval* sahasında tanımlanmış 5 saniye aralığında 0 ile 1 arasında noktalı bir değer gönderir. *loop* sahası bu gönderimin sürekli olacağını belirtir. Bu kesir PositionInterpolator nesnesinin *fraction* sahasına

gönderilir. Bu nesne fraction sahasına bir olay aldığı zaman *value* sahası üzerinden olay gönderecek şekilde tanımlanmıştır. Value *key* ve *keyValue* sahaları tarafından belirlenmektedir. Bu durumda -1 ve +1 arasında y değerine sahip bir vektör göndermektedir. Bu değer Transform düğümünün *translation* sahasına gönderilmektedir. Bu düğüm *translation* sahasındaki değere göre çocukların yerini ayarlama tanımına sahiptir.

Nesneleri program erişimi yoluyla değiştirmek: Rotalama düzeneği basittir, ancak düğümlerin saha değerlerini değiştirmeyle sınırlanmıştır, ve sadece verilen düğüm kümesini değiştirecek şekilde tasarlanmıştır. Daha iyi esneklik için, bazı profiller sistemdeki nesnelere program erişimi olanağı sağlamaktadır. Bu şekilde erişim, sahaların değerlerinin okunması, yazılması ve işlevlerin çağrılmasını sağlar. Düzenekler ayrıca PROTO nesnelerinin bulunmasına, böylece bu tip nesnelerin örneklenmesine olanak sağlar.

Bu şekil program erişim X3D içerisinde iki şekildedir: Dışsal Erişim (gömüldüğü program içerisinden erişim), İçsel Erişim (desteklenen betik dilleri kullanılarak betikler aracılığıyla erişim)

Nesnelere program erişimi bu nesnelere *arabirimler* yardımıyla sağlanır. Bir nesnenin arabirimi (veri ve işlev özellikleri) tanımlanmış ve *nesne tipi* olarak adlandırılmaktadır. Bir düğümü temsil eden nesne tipi ayrıca *düğüm tipi* olarak da bilinir. Nesne tipleri hem soyut hem de somut olabilir. Soyu nesne tipleri örneklenemez. Bunun yerine, diğer nesne tiplerinden kalıtmak veya bir sahanın türetilebilen düğüm tipinden herhangi birini içerebildiğini belirtmek için kullanılır. Somut nesne tipleri, soyut nesne tiplerinden türetilmiş ve örneklenebilir tiplerdir.

3.4.4 Nesne Yaşam Döngüsü

Düğümler bir yaşam döngüsüne sahiptir: yaratılır, kullanılır ve yok edilirler. Aşağıdakilerden bir veya daha fazlası doğru ise nesne canlı olarak kabul edilir:

a) Düğüm sahnedeki kök düğümse
b) Düğüm canlı bir düğümün sahasında referans edilmişse
c) Canlı bir betikten düğüme referans varsa

d) Düğüme dışsal program referansı varsa

b ve c kuralları öz yinelemeli olarak tüm canlı sahne çizgesini kapsamaktadır.

3.5 DEF/USE Anlam bilimi

DEF anahtar kelimesi kullanılarak isimlendirilmiş bir düğüme bu isim kullanılarak daha sonra USE ve ROUTE cümleleri yardımıyla başvurulabilir. USE cümlesi düğümün bir kopyasını oluşturmaz. Bunun yerine, aynı düğüm sahne çizgesine ikinci kere yerleştirilir, böylece düğüm birden fazla ataya sahip olur.

Düğüm isimleri tek bir X3D dosyasına, prototip tanımına veya, tarayıcı genişletmesi createX3DfromString veya betikteki SFNodes yaratma düzenekleriyle oluşturulmuş karakter dizileri kapsamına sınırlandırılmıştır. “NewNode” adı verilmiş bir düğüm (DEF NewNode), NewNode faaliyet alanındaki SFNode veya MFNode sahalarında herhangi bir “USE NewNode” cümlesi NewNode düğümünü referans eder.

Düğüm isimleri DEF anahtar kelimesinin bulunduğu içerikte tekil olmalıdır.

3.6 Prototip anlam bilimi

PROTO cümlesi daha önce tanımlanmış (var olan veya prototiplenmiş) düğüm tiplerini kullanarak yeni düğüm tanımlar. Bir kere tanımlandıktan sonraki prototiplenmiş düğüm tipler var olan düğüm tipleri gibi sahne çizgesinde örneklenebilir.

Düğüm isim tipler her X3D dosyası için tekil olmalıdır. Var olan düğümlerle veya daha önce aynı kapsamda tanımlanmış bir prototiple aynı isim verilen prototip oluşturmanın sonucu bilinmemektedir.

Prototip ara birimi yeni düğümün sahalarını ve bu sahaların erişim türünü belirler. ara birim tanımlaması, prototipin sahalarının tipleri,

isimleri ve var sayılan değerlerini (sadece initializeOnly ve inputOutput sahalar için) içerir.

Arabirim tanımlaması inputOutput saha tanımlamalarını içerebilir, bu sahalar aynı zamanda initializeOnly, inputOnly ve outputOnly olan sahayı tanımlamanın elverişli bir yoludur. Zzz olarak adlandırılmış bir inputOutput saha, zzz adlı initializeOnly, set_zzz isimli inputOnly ve zzz_changed isimli outputOnly bir sahayı ayrı tanımlamayla aynı anlamdadır.

Her prototip örneği prototipin tam kopyası olarak kabul edilebilir, bu örnek kendi saha değerleri ve prototip tanımının kopyasını içerir. Prototiplenmiş düğüm standart düğüm söz dizimi kullanılarak örneklenir. Örneğin aşağıdaki prototip (boş arabirim tanımlamasına sahip):

```
<ProtoDeclare name='Cube'>
<ProtoBody>
     <Shape><Box/></Shape>
</ProtoBody>
</ProtoDeclare>
```

Aşağıdaki şekilde ilklenebilir:

```
<Transform>
<ProtoInstance name='Cube'
     containerField='children'/>
</Transform>
```

Eğer prototip içerisindeki outputOnly bir saha, inputOutput bir saha ile ilişkilendirilmişse ilişkili outputOnly saha inputOutput sahanın ilk değeri olmalıdır. Eğer outputOnly saha birden fazla inputOutput saha ile ilişkilendirilse beklenmeyen sonuçlar oluşabilir.

Prototip tanımı bir veya daha fazla düğüm, iç içe geçmiş PROTO cümleleri ve ROUTE cümlelerinden oluşur. İlk düğüm tipi prototipin X3D dosyası içerisinde nasıl ilkleneceğini belirler. İlkleme prototip tanımlamasının parametreleri doldurmak ve ilk düğüm (ve düğümün sahne çizgesinin) kopyasını ilklemenin oluştuğu yere yerleştirmek

şeklinde gerçekleşir. Örneğin eğer prototipteki ilk düğüm bir Material düğümü ise, bu prototipin örnekleri Material düğümünün kullanılabildiği her yerde kullanılabilir. Diğer düğümler ve eşlik eden sahne çizgeleri dönüşüm sıradüzeninin parçası olmayıp, prototip tanımı içerisindeki ROUTE veya Script düğümleri tarafından referans edilebilirler.

Prototip tanımı içerisindeki düğümlerin sahaları prototip ara birim tanımlamasındaki sahalarla düğüm gövdesinde IS cümlesi kullanılarak ilişkilendirilebilir. X3D dosyasından prototip örnekleri okunduğu zaman, prototip ara birim saha değerleri verilebilir. Verildiği zaman prototip tanımındaki tüm düğümler için bu IS cümlesindeki değerler kullanılır. Benzer olarak, prototip örneğindeki bir girdi sahası bir olay gönderdiği zaman, bu saha için IS cümlesine sahip olan tüm düğümlere bu olay iletilir. Prototip örneğindeki bir düğüm IS cümlesine sahip bir çıktı olayı ürettiği zaman, bu çıktı sahasına bağlantılı (ROUTE yoluyla) tüm girdi sahalarına olay gönderilir.

IS cümleleri prototip tanımlamasının içerisinde sahalar nerede gözüküyorsa gözükebilir. IS cümleleri prototip tanımlamasında tanımlanmış sahalara referans verebilir. Var olmayan bir tanımlamayı referans eden IS cümlelerinin sonuçları bilinmemektedir. IS cümlesiyle ilişkilendirilen sahanın tipi prototip ara birim tanımlamasındaki tip ile uyumsuz ise oluşacaklar bilinmemektedir. Örneğin, SFColor tipini SFVec3f ile ilişkilendirmek geçersizdir. SFColor ve MFColor'u da ilişkilendirmek geçersizdir. Bunların terside doğrudur.

Eğer IS cümlesi aşağıdaki şekildeyse sonuçlar bilinmemektedir:

1. inputOnly saha initializeOnly veya outputOnly saha ile ilişkiliyse
2. outputOnly saha initializeOnly veya inputOnly saha ile ilişkiliyse
3. initializeOnly saha inputOnly veya outputOnly saha ile ilişkiliyse

Prototip arabirimindeki bir inputOutput saha sadece prototip tanımındaki inputOutput saha ile ilişkilendirilebilir, fakat prototip tanımındaki inputOutput saha prototip arabirimindeki inputOutput, inputOnly veya outputOnly sahayla ilişkilendirilebilir (Bkz. Çizelge 3.1).

Çizelge 3.1: Düğüm örneklerine PROTOTYPE eşleme kuralları

	inputOutput	initializeOnly	inputOnly	outputOnly
inputOutput	Evet	Evet	Evet	Evet
intializeOnly	Hayır	Evet	Hayır	Hayır
inputOnly	Hayır	Hayır	Evet	Hayır
outputOnly	Hayır	Hayır	Hayır	Evet

Başka bir prototip tanımlaması içerisinde yer alan prototipler (iç içe geçmiş) kendisini kapsayan prototipe yereldir. İç içe geçmiş prototip gerçekleştirimleri en içteki prototip tanımlamasına referans ediyor olabilir.

Bir PROTO cümlesi, tüm sahneden ayrı ve herhangi iç içe geçmiş PROTO cümlesinden ayrı DEF/USE isim kapsamı oluşturur. Prototipin içerisinde DEF yapısıyla isimlendirilmiş düğümler prototipin kapsamı dışındaki USE yapıları tarafından referans edilemezler. Prototip kapsamı dışında DEF yapısıyla tanımlanmış düğümler de prototip kapsamı içerisinde USE yapılarıyla referans edilemezler.

Bir prototip prototip tanımlaması bittikten sonra dosyanın herhangi bir yerinde ilklenebilir. Bir prototip kendi gerçekleştirimi içerisinde ilklenemez (Öz yinelemeli prototipler geçersizdir).

3.7 Dışsal prototip anlam bilimi

EXTERNPROTO cümlesi yeni düğüm tipi belirler. İki istisna dışında PROTO cümlesine denktir. İlk olarak, düğüm tipinin gerçekleştirimi dışsal olarak ya uygun PROTO cümlesini içeren bir X3D dosyasında veya gerçekleştirim bağımlı başka bir düzenekle saklanır, ikinci olarak var sayılan saha değerleri verilmez, çünkü gerçekleştirim uygun değerleri tanımlar.

EXTERNPROTO ara birim anlam bilimi PROTO cümlesi ile aynıdır. Ancak var sayılan değerler yerel olarak belirlenmez. Ek olarak,

örneğe gönderilen olaylar düğümün gerçekleştirimi bulunana kadar ihmal edilir.

Tanımlama yüklenene kadar, tarayıcı inputOutput sahaların ilk değerlerine aşağıdaki kurallara dayanarak karar vermelidir (öncelik sırasına göre);

- Örnekteki kullanıcı tanımlı değer (eğer tanımlanmışsa)
- Bu saha tipi için var sayılan değer

outputOnly sahalar için, başlangıçtaki ilk değer, bu veri tipi için var sayılan değer olacaktır. EXTERNPROTO yüklenmesi aşamasında, outputOnly sahanın ilk değeri bulunduğunda bu değer sahaya uygulanmakta ve herhangi bir olay üretilmemektedir.

Ara birim tanımlamasındaki sahaların isim ve tipleri gerçekleştirimde tanımlanmış olanların bir alt kümesi olmalıdır. Gerçekleştirimde olmayan bir saha ve gerçekleştirimdekinden farklı tip tanımlaması bir hatadır.

Ara birim tanımlamasından sonra belirtilen karakter katarları prototipin gerçekleştiriminin konumunu verirler. Eğer birden fazla karakter katarı tanımlanmışsa, tarayıcı tercih sırasına göre arama gerçekleştirir.

Eğer EXTERNPROTO cümlesindeki bir URL bir X3D dosyasını referans ediyorsa, dosya içerisinde bulunan ilk PROTO cümlesi dışsal prototipin tanımlaması için kullanılır. EXTERNPROTO cümlesindeki prototipin ismi ile dosyada PROTO tanımlamasındaki ismin aynı olması gerekmez. Eğer URL X3D olmayan bir dosyayı referans ediyorsa sonuçlar tahmin edilemez.

Tekrar kullanılabilir PROTO tanımlama kütüphanelerinin kullanılabilmesi için, tarayıcılar “#isim” ile biten EXTERNPROTO URL tanımlamalarının referans edilen X3D dosyası içerisinde “isim” ile adlandırılmış PROTO cümlesini referans ettiğini bilmelidir. Örneğin, standart materyalleri tanımlayan “materyal.wrl” dosyası aşağıdaki şekilde tanımlanmış olsun:

```
#X3D V3.0 utf8
```

```
PROTO Altin [] { Material { ... } }
PROTO Gumus [] { Material { ... } }
...........
```

Bu kütüphanedeki bir materyal aşağıdaki şekilde kullanılabilir:

```
#X3D V3.0 utf8
EXTERNPROTO AltinKutuphaneden []
"http://.../materyal.wrl#Altin"
   ...
 Shape {
   appearance Appearance { material
AltinKutuphaneden {} }
   geometry   ...
 }.....
```

3.8 Dıştan alma (Import) / Dışa aktarma (Export) anlam bilimi

IMPORT özelliği, yazarların Inline düğümlerle veya programlama yoluyla yaratılan içeriğin dosyanın isim uzayına olay yönlendirme amacıyla birleştirilmesine olanak sağlar. Dışsal prototiplemeye -dışsal dosyalarda tanımlanmış prototiplerin her sahasına erişim sağlayan- karşıt olarak IMPORT, dışsal olarak tanımlanmış düğümün tüm sahalarına tek cümle ile erişim sağlar.

İçselleştirilmiş dosyalardan düğümlerin dıştan alınması iki cümle ile sağlanır: IMPORT ve EXPORT. IMPORT cümlesi hangi Inline düğümlerin ilgili dosyanın isim uzayına birleştirileceğini tanımlar. EXPORT cümlesi ise diğer dosyalara görünür kılınacak düğümlerin erişiminin kontrol edilmesinde kullanılır.

IMPORT cümlesi, yazarların Inline düğümlerle veya programlama yoluyla yaratılan içeriğin dosyanın isim uzayına olay yönlendirme amacıyla birleştirilmesine olanak sağlar. Bir düğüm içe aktarıldığı zaman, ROUTE yardımıyla sahalarına olaylar gönderilebilir, veya

sahalarından dışarıya olaylar yönlendirilebilir. IMPORT cümlesi aşağıdaki bileşenlere sahiptir:

- İçe aktarılacak düğümü içeren Inline düğümün ismi
- İçe aktarılacak düğümün ismi
- İsim çakışmalarını önlemek için içe aktarılan düğümün çalışma zamanı isim kapsamında tanımlı lakabı

IMPORT cümlesi aşağıdaki anlam bilimine sahiptir:

- İçe aktarıldıktan sonra yeni düğüm DEF ile tanımlanmış herhangi bir düğüm gibi yönlendirilebilir.
- IMPORT cümlesi ile içe aktarılan düğümler USE cümlesi yardımıyla örneklenemez.
- Sadece EXPORT cümlesi kullanılarak dışa aktarılmış Inline düğümler IMPORT cümlesi kullanılarak içe aktarılabilir.

Aşağıdaki örnek IMPORT cümlesinin kullanımını göstermektedir:

```
<Inline  DEF='I1'
  url=' "url.x3d"
"http://online.adres/url.x3d" '/>
<IMPORT  AS='I1Root'/>
<PositionInterpolator  DEF='PI'/>
<ROUTE  fromNode='PI'
fromField='value_changed' toNode='I1Root'
toField='set_translation'/>
```

Yukarıdaki örnekte rootTransform url.x3d dosyası içerisinde Transform düğümü olarak tanımlanmış ve EXPORT cümlesi kullanılarak dışa aktarılmıştır. AS kelimesi rootTransform için bir takma isim ayarlamakta, böylece rootTransform sahne içerisinde I1Root ismiyle referans edilmektedir.

EXPORT cümlesi X3D dosyası içerisinde diğer sahnelere aktarılabilecek düğümleri tanımlamakta kullanılmaktadır. Sadece EXPORT cümlesi ile dışa aktarılan isimler Inline cümlesi yardımıyla içe

aktarılmış dosyalardaki düğüm isimlerinin kullanılmasına olanak sağlar. EXPORT cümlesi aşağıdaki bileşenlere sahiptir:

- Dışa aktarılacak düğüm ismi
- Dışa aktarılan düğümün diğer dosyalarda içe aktarılırken kullanılacak olan takma ismi

EXPORT cümlesinin anlam bilimi:

- Dışa aktarılan düğümler içe aktarıldıkları zaman DEF ile tanımlanmış düğümler gibi yönlendirilebilir.
- Dışa aktarılan düğümler USE cümlesiyle örneklenemez.
- IMPORT cümlesi ile içe aktarılmış bir düğüm başka bir dosyada kullanılmak üzere EXPORT düğümü ile dışa aktarılamaz.

Aşağıdaki örnek EXPORT cümlesinin kullanımını göstermektedir:

```
<Transform  DEF='T1'/>
<EXPORT  localDEF='T1' AS='rootTransform'/>
```

Yukarıdaki örnekte, T1 diğer X3D sahneleri tarafından kullanılmak üzere rootTransform takma ismiyle dışa aktarılmaktadır. AS kelime bu takma ismi tanımlamak için kullanılmaktadır. Diğer sahneler sadece rootTransform kelimesini kullanarak içe aktarabilirler.

3.9 Olay Modeli

Olaylar X3D çalışma zamanı ortamında davranış üretmenin birincil yöntemidir. X3D içerisinde olaylar şu şekillerde kullanılmaktadır: zaman tabanlı canlandırmaların işletilmesi, nesne seçmenin işlenmesi, kullanıcı hareketi ve çarpışmanın saptanması, sahne çizge sıradüzeninin değiştirilmesi. Çalışma zamanı ortamı olayların sistemde yayılımını iyi tanımlanmış kurallara göre yönetir.

Düğümler davranışı tetikleyen girdi sahalarını (inputOutput veya inputOnly sahalar) tanımlar. Bir olay oluştuğu zaman, düğüm bildirimi alır ve içsel durumunu ve sahalarından bir veya daha fazlasının değerini değiştirir. Düğümler ayrıca düğüm içerisindeki durum değişim

işaretlerini veya diğer oluşumları gönderen çıktı sahalarını (inputOutput veya outputOnly sahalar) tanımlar. Girdi sahalarına gönderilen ve çıktı sahalarından gönderilen olaylar toplu olarak olaylar olarak tanımlanır.

Rotalar yazarın bir düğümün çıktı olaylarını bir başka düğümün girdi olaylarıyla birleştirmesini sağlar. Böylece karmaşık davranışların zorunlu programlama olmadan gerçekleştirilmesi sağlanır. Yönlendirilmiş bir olay tetiklendiği zaman, ilgili hedef girdi olayı bildirimi alır ve bu değişime göre tepkiyi işler. Bu işleme düğümün durumunu değiştirebilir, yeni olaylar üretebilir veya sahne çizgesinin yapısını değiştirebilir. Rotalar X3D dosyası içerisinde tanıtılabilir veya SAI çağrımlarıyla programlama yoluyla tanımlanabilir.

Rotalar düğüm değildir. ROUTE cümlesi düğümler veya sahalar arasında olay yolları oluşturmak için bir yapıdır. ROUTE cümleleri X3D dosyasının en üstünde veya bir düğümün sahaları nerede olabiliyorsa koyulabilir. Kaynak ve hedef düğümünden sonra koyulabilir, bir düğümün içine koyulması o düğümle bir ilişki olmasını sağlamaz.

Bileşen veya destek seviyesi daha farklı bir kural önermiyorsa hedef saha tipi kaynak sahanın tipiyle aynı olmalıdır.

Gereksiz yönlendirmeler ihmal edilir. Eğer X3D dosyası yönlendirme yolunu tekrarlıyorsa ikinci ve sonraki benzer rotalar ihmal edilir. Bu SAI yardımıyla dinamik olarak yaratılan rotalar için de geçerlidir.

Yazarlar tarafından prototipleme düzeneği kullanılarak yaratılan düğümler gelen olayların özel olarak işlenmesine olanak sağlar. Prototiplenmiş düğümlere gelen olaylar, ara birim sahası yardımıyla işlenmek için içsel düğümlere veya diğer ara birim sahalarına dışarıdaki düğümlere yayılması için yönlendirilebilir. Yazar ayrıca olayları işleme mantığını Script düğümü kullanarak gerçekleştiren düğümlere ara birim yardımıyla yönlendirebilir.

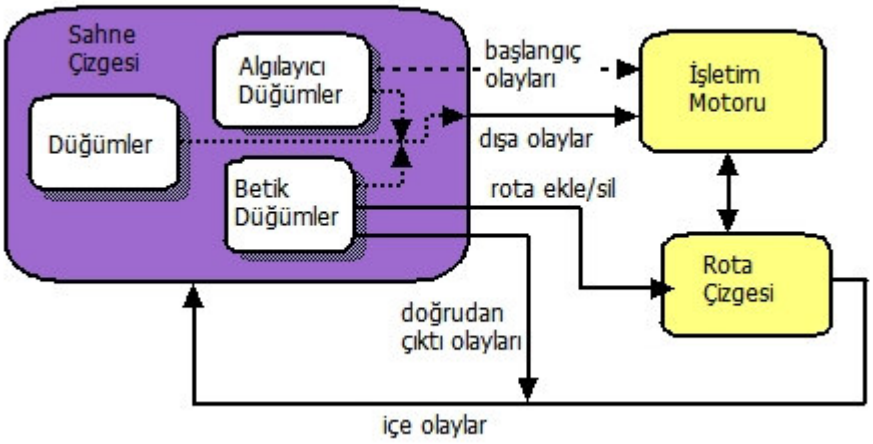
Bir algılayıcı veya betik bir *başlangıç olayı* ürettiği zaman, olay üretildiği sahadan ROUTE'lar ile diğer düğümlere yayılır. Bu diğer düğümler tüm rotalar işlenene kadar yeni olaylar üreterek tepki verir. Bu sürece *olay şelalesi* adı verilir. Bir olay şelalesi içerisinde üretilen tüm

olaylar başlangıç olayı ile aynı zaman imzasına sahip olur. Bunun nedeni tüm olayların beraber hemen olduğunun dikkate alınmasıdır.

Başlangıç olayının şelalesindeki tüm olaylar oluştuktan sonra, olay-sonrası işleme olay şelalesi tarafından uyarılmış eylemleri gerçekleştirir. Tarayıcı aşağıdaki eylem dizisini tek bir zaman imzasında gerçekleştirmelidir:

1. Kamerayı o an bağlı olunan bakış açısının (Viewpoint düğümü) konum ve yönelimine göre güncelle
2. Algılayıcılardan girdileri değerlendir.
3. Rotaları değerlendir.
4. 2 veya 3 adımlarında bir olay üretildiyse 2 adımına gidip devam et.

Şekil 3.3 işletim modelinin kavramsal bir örneğini göstermektedir.



Şekil 3.3: Kavramsal İşletim Modeli

3.10 Genişletilebilir İşaretleme Dili (XML) Kodlama

Genişletilebilir İşaretleme Dili (XML "Extensible Markup Language") Genişletilebilir 3B (X3D "Extensible 3D") 'nin X3D

kodlaması için kendi kendini doğrulamasında kullanılmaktadır. Bu X3D kodlama diğer web dilleriyle birlikte çalışabilirliği arttıran web uyumluluğu sağlar. XML, HTML("Hypertext Markup Language")'ye benzeyen sistemler ve insanlar tarafından okunabilen yapısal veriyi destekler. Açıklık için gereksiz sözler içerir, teknolojilerin modüler bir ailesini temsil eder ve esas olarak Web teknolojileriyle birlikte çalışabilir.

X3D dosya yapısı: Dosyadaki ilk satır UTF-8 metninden oluşmaktadır ve dosyanın bir XML dosyası olduğunu, doğrulamada kullanılacak XML sürümünü ve karakter kodlamasını tanımlar.

Her X3D dosyası aşağıdaki XML dosya tanımlaması ile başlamalıdır:

```
<?xml version="1.0" encoding="UTF-8"?>
```

"UTF-8" tanımlaması X3D dosyası içerisinde Uluslar arası karakterlerin UTF-8 karakter kodlaması kullanılarak gösterilmesini sağlar.

XML başlığından hemen sonra XML DOCTYPE cümlesi gelmelidir. Bu cümle PUBLIC ve SYSTEM kelimeleriyle X3D DTD ("Document Type Definition") bilgisini sağlar.

X3D DOCTYPE tanımlaması şu şekildedir:

```
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D
3.0//EN"
"http://www.web3d.org/specifications/x3d-
3.0.dtd">
```

DTD doğrulamasına başka bir seçenek olarak XML Schema doğrulaması XML Schema ve X3D isim uzaylarının tanımlanmasıyla gerçekleştirilebilir. Aşağıda buna bir örnek verilmektedir:

```
<X3D profile='Immersive'
xmlns:xsd="http://www.w3.org/2001/XMLSchema-
instance"
xsd:noNamespaceSchemaLocation="http://www.we
b3d.org/specifications/x3d-3.0.xsd">
```

DOCTYPE veya Schema tanımlamaları isteğe bağlıdır. Genişletilmiş etiket kümelerini desteklemek için yazarlar temel belge tanımlamalardan farklı belgeleri referans edebilirler. Ancak bu diğer DTD'ler temel X3D DTD/Schema'ya uyumlu olmalıdır. Yazarlar X3D DOCTYPE cümlesini işlevsellik eklemek için genişletme düzenekleri olarak kullanmalıdır.

Daha sonra X3D belge kök etiketi gelmelidir. Bu etiket profil bilgisi ve isteğe bağlı Schema doğrulama bilgisini içerir. X3D belge kök etiketinde isteğe bağlı başlık bölümü vardır ve en sonda zorunlu olan sahne gövdesi yer alır.

ÖRNEK

```
<X3D profile='Immersive'> <!-- X3D kok
etiketi Immersive profili kullaniliyor-->
<head> <!-- Istege bagli baslik bilgisi
baslangici-->
 <meta name='description' content=‘Basit bir
silindir ornegi. '/>
 <meta name=‘dosyaAdi'
content=‘KirmiziKutu.x3d'/>
</head>
<Scene><!-- Zorunlu sahne bilgisi -->
     <!-- Sahne cizge dugumleri buraya
ekleniyor -->
     <Group>
     <Shape>
          <Appearance DEF=‘KIRMIZI'>
          <Material diffuseColor='1 0 0'/>
          </Appearance>
          <!- Varsayilan kirmizi kutu
kenarlar=1,merkeze yerlestirilir-->
          <Cylinder />
     </Shape>
```

```
</Group>
</Scene>
</X3D>
```

XML kodlamada <!-- ve --> karakter serileri yorumları belirlemek için kullanılır.

ÖRNEK

```
<!-- Bu bir X3D yorumudur. -->
```

Bu duruma istisnalar sadece XML CDATA bölümlerinde yer alan, XML ayrıştırması sırasında yok sayılan ve gömülü betik kodu için kullanılan metinlerdedir. Ayrıca çift tırnak ile ayrılmış SFString ve MFString sahaları içerisindeki kısımlar da yok sayılır. Buradaki karakterler özel karakterlerle temsil edilir (< yerine < ve > yerine >).

Özellik değerleri tek tırnak (') veya çift tırnak (“) ile tanımlanmaktadır. XML işleyicilerin kendisi hangi tip verilerle karşılaştığına karar vermektedir. Ne yazarlar ne de ayrıştırıcılar karşılaşılan karakterlerin tipine garanti verebilir.

XML öğeleri (örneğin X3D düğümleri) özellik değerleri içerisinde gözükmeyebilir.

XML özellik değerleri dışında karşılaşılan boşluklar, tablar, satır beslemeler ve satır başları ayırıcı karakterler olarak kullanılır. Virgüller XML öğeleri veya özellik değerleri arasında karıştığı zaman beyaz boşluk olarak yok sayılmaz.

X3D özellik değerleri içerisinde kurallar biraz farklıdır. Virgüller, boşluklar, tablar, satır beslemeler ve satır başları ayırıcı karakter olarak kullanılırlar. Ayırıcı karakterler ve yorumlar toplu olarak *beyaz boşluk* olarak adlandırılır.

Saha, olay, PROTO, EXTERNPROTO ve düğüm isimleri kontrol karakterleri (0x0-0x1f, 0x7f), boşluk (0x20), çift veya tek tırnak (0x22: ", 0x27:'), diyez (0x23: #), virgül (0x2c:,), nokta (0x2e:.), köşeli parantez (0x5b,0x5d: []), ters bölü (0x5c:\) veya parantez (0x7b, 0x7d:{}) içermemelidir . İlk karakter sayı (0x30-0x39), artı (0x2b: +) veya eksi

(0x2d: -) olmamalıdır. Diğer durumlarda isimler ISO 10646 karakter kodlanmış UTF-8 kullanılmış olabilir. X3D büyük küçük harf duyarlıdır. “Sphere” “sphere” kelimesinden farklıdır ve “BEGIN” “begin”'den farklıdır.

Farklı X3D kodlamaları arasındaki çalışabilirliği korumak için aşağıdaki anahtar kelimeler saha, olay, prototip veya düğüm isimleri için kullanılmamalıdır:

- AS
- component
- DEF
- EXPORT
- EXTERNPROTO
- FALSE
- head
- IMPORT
- initializeOnly
- inputOnly
- outputOnly
- inputOutput
- IS
- meta
- NULL
- PROTO
- ROUTE
- Scene
- TO
- TRUE

- USE
- X3D

Cümleler ve Yapılar: Gerekli olan XML başlık ve <X3D> belge kök etiketinden sonra X3D dosyası aşağıdakilerin herhangi bir çeşitlemesini içerebilir:

a. Tek bir isteğe bağlı <head> öğesi, bu öğe bileşen ve üst bilgiyi içerir.

b. Zorunlu tek <Scene> öğesi, bu öğe sahnenin kök düğümü olarak hizmet eder ve bir veya daha fazla sahne çizgesi öğesi içerebilir.

c. 0 veya daha fazla sayıda <ProtoDeclare> veya <ExternProtoDeclare> yapısı

d. 0 veya daha fazla düğüm öğesi

e. 0 veya daha fazla USE yapısı

f. 0 veya daha fazla ROUTE yapısı

g. 0 veya daha fazla IMPORT veya EXPORT yapısı

X3D düğümleri XML öğeleri olarak ifade edilir (etiketlenmiş isimler). Basit düğüm olmayan X3D sahaları, isim="değer" biçiminde XML özelliği olarak ifade edilir.

Diğer X3D düğümlerine saha olarak hizmet eden düğümler içerilmiş XML öğesi olarak ifade edilir (ek etiketlenmiş isimler).

Bir düğüm cümlesi düğüm öğesinin tip isminden, onu takip eden basit saha özelliklerinden oluşur. Bir düğüm örneği DEF özelliği kullanılarak yaftalanabilir.

Her etiket < ve > karakterleriyle başlar ve biter. Bir düğümün gövdesi ilgili açma ve kapama etiketleriyle sınırlandırılabilir. (Örnek: <Sphere><!-- Dugum bilgisi --></Sphere>)

ÖRNEK 1 düğüm ve saha sözdizimini kullanmayı göstermektedir :

```
<Group DEF='OrnekCocukOge'>
  <Shape>
```

```
    <Box/>
    <Appearance>
     <Material diffuseColor='0.6 0.4 0.2'/>
    </Appearance>
  </Shape>
</Group>
```

ÖRNEK 2 Başka bir seçenek olarak kapatma aynı etiketin sonuna "/" işareti koyularak da yapılabilir:

```
<Viewpoint DEF='OrnekTekOge'
description='Merhaba Sozdizimi'/>
<NavigationInfo type='"EXAMINE" "ANY"'/>
```

Bir düğüm öğesi her düğüm tipine uygun sıfır veya daha fazla saha özelliği içerebilir. ProtoDeclare yapısının ProtoBody içerisine yuvalandığı zaman, bir düğümün gövdesi isteğe bağlı olarak IS cümlesi ile başlayabilir, bu cümle bir veya daha fazla *connect* cümlesi içerebilir. Bir düğüm daha sonra sıfır veya daha fazla çocuk düğümü, ROUTE cümleleri, ProtoDeclare veya ExternProtoDeclare cümleleri içerebilir. Bu içerilen düğümler ve cümleler her düğüm tipi için uygun olarak kabul edilmiştir ve herhangi bir sırada olabilir.

Beyaz boşluklar düğümün ismini ve ozellik="deger" çiftlerini birbirinden ayırmalıdır, ancak etiketlerin baş ve sonlarında olması şart değildir.

Düğümler arasındaki ata-çocuk ilişkilerinin XML tabanlı sahne doğrulamasıysa X3D Belge Tip Tanımlaması ("Document Type Definition" DTD) veya X3D Schema ile gerçekleştirilmektedir.

DEF ve USE özellik söz dizimi: USE özelliği DEF ile tanımlanmış bir düğümün referans kopyası olduğunu belirtir. DEF XML tipi ID ile atanır, USE ise XML tipi IDREF ile atanır.

Sahnedeki her DEF (ID) değeri tekil olmalıdır. Tekil olmayan DEF isimleri doğrulama hatalarına sebep olabilir. Bu XML'in kısıtlamasıdır.

```
<Transform DEF='OrnekUSE'
rotation='0 1 0 0.78'
translation='0 2.5 0'>
```

```
<Group USE='OrnekTekOge'/>
</Transform>
```

USE=”herhangiBirIsim” özelliğini içeren düğümler *containerField* ve *class* dışında özellik içeremezler.

containerField özellik söz dizimi: Her düğüm tipi, bu tipin içerebileceği sahaların tip ve ismini tanımlar. Genel olarak, neredeyse tüm saha düğüm ilişkisi X3D'deki iyi tanımlanmış ata-çocuk ilişkisi sayesinde açıktır.

containerField özelliği düğümlerin içerilebileceği düğüm tiplerini belirlemek için kullanılmaktadır. Örneğin Sensor düğümleri Gruplama düğümlerinin children sahaları içerisinde barındırılmaktadır, bu yüzden tüm Sensor düğümleri containerField='children' özelliğini varsayılan olarak içerir.

containerField değeri ihmal edilebilir. İlgili düğümün varsayılan containerField değeri düğümün atasıyla farklı bir ilişkisi var ise değiştirilebilir.

```
<Collision>
<Shape containerField='proxy'>
     <Sphere/>
</Shape>
<Group USE='OrnekCocukOge'/>
</Collision>
```

Prototip ve saha tanımlama söz dizimi: Prototip tanımlaması ProtoDeclare anahtar kelimesi, onu takip eden prototipin isim özelliği, isteğe bağlı arabirim saha tanımlamalarını içeren ProtoInterface ve zorunlu olan gövde tanımlamasını içeren ProtoBody yapılarından oluşur.

Prototip saha cümleleri field anahtar kelimesi, onu sırasıyla takip eden name sahasının değeri, type sahası ve accessType (inputOnly, outputOnly,initializeOnly veya inputOutput değerlerini içerebilir) sahasını içerir.

ÖRNEK Bir prototip tanımlaması (saha tanımlamaları olan ve olmayan):

```
<ProtoDeclare name='YeniDunyaBilgiDugumu'>
<ProtoBody>
     <WorldInfo DEF='OrnekPrototipGovde'/>
</ProtoBody> </ProtoDeclare>
<ProtoDeclare name='YayiciMateryal'>
<ProtoInterface>
     <field name='sadeceRenk'
          type='MFColor'
          accessType='inputOutput'/>
</ProtoInterface>
<ProtoBody>
<!-- Varsayılan diffuseColor degeri 0.8 0.8
0.8 degistiriliyor-->
<Material diffuseColor='0 0 0'>
<!-- su anki dugumun yayiciRenk sahasini
ProtoDeclare atasının sadeceRenk sahasına
bagliyoruz. -->
<IS>
     <connect nodeField='emissiveColor'
          protoField='sadeceRenk'/>
</IS>
</Material>
</ProtoBody>
</ProtoDeclare>
```

Bir saha cümlesi ayrıca eğer erişim tipi (accessType) initializeOnly veya inputOutput ise varsayılan değeri içermelidir. Basit değerler özelliğin değerinde içerilir(Örnek: deger='TRUE'). Düğüm değerleri sahanın tanımlaması içerisinde çocuk düğümler olarak içerilir.

Örnek **<Group DEF='VarsayilanDugumDegeri'/>** ProtoDeclare yapısı için varsayılan deger ataması olmaktadır:

```
<ProtoDeclare name='GrubuYukari2MTasi'>
<ProtoInterface>
     <field name='children'
          type='MFNode'
          accessType='inputOutput'>
          <Group DEF='VarsayilanDugumDegeri'
```

```
                    bboxSize='2 2 2'/>
     </field>
</ProtoInterface>
<ProtoBody>
     <Transform translation='0 2 0'>
     <Group>
     <IS>
     <connect nodeField='children'
          protoField='children'/>
     </IS>
</Group>
</Transform>
</ProtoBody>
</ProtoDeclare>
```

Aynı saha çoklu düğüm prototipleri tarafından kullanılabilir.

IS/connect cümle söz dizimi: Prototip tanımlaması içerisindeki düğüm cümlesi isteğe bağlı olarak <IS> cümlesi içerebilir. <IS> cümlesi de bir veya daha fazla <connect> cümlesi içerebilir. Her <connect> cümlesi ata düğümün arabiriminden *nodeField* özelliğini, onun arkasından ProtoDeclare yapısından *protoField* özelliğini içerir.

Sahaların type ve accessType özellikleri tam olarak eş olmalıdır. nodeField ve protoField isimleri aynı olabilir.

ExternProtoDeclare cümle söz dizimi: ExternProtoDeclare cümlesi ExternProtoDeclare öğesinden, referans edilen prototipin name özelliğinden ve prototipi tanımlayan sahneyi referans eden bir veya daha fazla adresi gösteren URL'den oluşmaktadır. Sıfır veya daha fazla içerilmiş saha tanımlaması prototipin arabirimini sağlamak için takip edebilir.

İçerilmiş saha cümleleri field anahtar kelimesini, onu takip eden name saha değerini, type saha değerini ve saha erişim tipini (accessType) içerir.

URL değerleri çift tırnaklarla belirlenmiş yerel veya uzak dosyaların adreslerini içerir. Her dosya ismi veya adres # işaretiyle dışsal olarak tanımlanmış prototipin ismini içermelidir.

ÖRNEK Dışsal prototip tanımlaması:

```
<ExternProtoDeclare
name='BakisKonumYonelim'
nodeType='Viewpoint'
url='"BakisKonumYonelim
Prototype.wrl#BakisKonumYonelim"
"http://www.myorg.org/Examples/BakisKonumYon
elim Prototype.wrl#BakisKonumYonelim"
"BakisKonumYonelim
Prototype.x3d#BakisKonumYonelim"
"http://www.myorg.org/Examples/BakisKonumYon
elim Prototype.x3d#BakisKonumYonelim"'>
<field name='enabled'
     type='Boolean'
     accessType='exposedField'/>
```

ProtoInstance ve fieldVale cümleleri söz dizimi: ProtoInstance öğesi düğümlerin örneklerini yaratmak için kullanılmaktadır. Örneği yaratılacak olan düğümler sahnede ya ProtoDeclare yapısıyla veya ExternProtoDeclare yapısıyla tanımlanmış olmalıdır.

ProtoInstance sıfır veya daha fazla fieldValue öğesi içerebilir. Bu öğeler ProtoDeclare veya ExternProtoDeclare içerisinde tanımlanmış ilgili sahaların ilk değerlerini sağlamada kullanılmaktadır. Bu fieldValue ilklemeleri erişim tipleri initializeOnly veya inputOutput olan sahalar için kullanılmaktadır.

ÖRNEK

```
<Transform translation='0 -2.5 0'>
<Shape>
     <Appearance>
     <ProtoInstance name='YayiciMateryal'>
          <fieldValue name='sadeceRenk'
               value='0.2 0.6 0.6'/>
     </ProtoInstance>
     </Appearance>
<Text string='"Prototip Soz dizimi"
"Ornekler"'>
```

```
     <FontStyle justify='"MIDDLE"
"MIDDLE"'/>
</Text>
</Shape>
</Transform>
<ProtoInstance name='BakisKonumYonelim'>
<fieldValue name='enabled' value='true'/>
</ProtoInstance>
```

ROUTE cümle söz dizimi: ROUTE cümlesi fromNode, fromField, toNode ve toField özelliklerini içerir. fromNode ve toNode özellikleri XML tipi IDREF şeklindedir. Anlamları önceki DEF değerlerinin eşlenmesidir.

Aşağıdaki anlambilimsel kısıtlamaları geçerlidir:

- `ROUTE` öğeleri sahne çizgesi öğesi değildir.
- ProtoDeclare içerisinde DEF'lenmiş düğümler farklı bir isim uzayı biçimlendirdikleri için ROUTE içerisinde ProtoDeclare içindeki öğeleri referans etmek hatalıdır.

ÖRNEK:

```
<TimeSensor DEF='Saat' cycleInterval='4'
loop='true'/>
<OrientationInterpolator
DEF='Egiren'
key='0 0.5 1'
keyValue='0 1 0 0,0 1 0 3.1416, 0 1 0
6.2832'/>
<ROUTE fromNode='Saat' fromField='fraction'
toNode='Egiren' toField='set_fraction'/>
<ROUTE fromNode='Egiren'
fromField='value_changed'
toNode='OrnekUSE' toField='rotation'/>
```

IMPORT/EXPORT cümle söz dizimi: Aşağıdaki XML söz dizimi X3D'nin IMPORT/EXPORT işlevselliğine uygulanmaktadır.

IMPORT cümlesi IMPORT öğesi, onu takip eden inlineDEF, exportedDEF ve AS özelliklerinden oluşur. EXPORT cümlesi EXPORT öğesi, onu takip eden localDEF ve AS özelliklerinden oluşur.

ÖRNEK 1 IMPORT cümlesinin XML kodlaması:

```
<Inline DEF='I1'
url='"herhangiBirURL.x3d"'/>
<IMPORT InlineDEF='I1'
exportedDEF='rootTransform'
AS='I1Root'/>
<PositionInterpolator DEF='PI'/>
<ROUTE fromNode='PI'
fromField='value_changed'
toNode='I1Root' toField='set_translation'/>
```

ÖRNEK 2 EXPORT cümlesinin XML kodlaması:

```
<Transform DEF='T1'/>
<EXPORT localDEF='T1' AS='rootTransform'/>
```

Betiklerin kod içerisine gömülmesi: Betik kodları XML ayrıştırıcıları tarafından ayrıştırılacak şekilde yerleştirilmemelidir. Bu yüzden bu tip kodlar ayrıştırmayı önlemek için uygun şekilde yerleştirilmesi gerekiyor. Bu tip uygun yerleştirimin tercih edilen yöntemi CDATA yapısı içerisinde, <field/> ve <IS><connect/></IS> tanımlamalarının takip ettiği şekilde yapılmasıdır. CDATA yapısı içerdiği tüm karakterlerin herhangi bir farklı işleme maruz kalmadan, değiştirilmeden kalmasını sağlar.

Eğer hem url sahası hem de bir CDATA cümlesine rast gelinirse, url sahası ilk olarak işlenir. Böylece CDATA yapısı url'ye ek olarak eklenecek değer olarak ele alınır. Bu sıralama çevrim içi çalışan sahnelerin içerisindeki betiklerin değişikliklerinin aynen yansımasını ve yazarın güncellemelerini kolaylaştırır.

ÖRNEK Script düğümü içerisinde CDATA yapısı kullanımı:

```
<Script directOutput='true'>
 <field name='ROOT' type='SFNode'
accessType='initializeOnly'>
    <Transform USE='ROOT'/>
```

```
  </field>
<![CDATA[

  javascript:

  function R ()
  {
    return Math.random();
  }

 function initialize()
  {
    for (i=0; i < 10; i++)
    {
     rand1 = 100*R();
     rand2 = 100*R();
     rand3 = 20*R();
     rand4 = 40*R();
     rand5 = 20*R();
     string =
      'Transform {' +
       '  translation ' + rand1 + ' 0 ' +
rand2 +
       '  children [' +
       '   Shape {' +
'     appearance Appearance {' +
 '   material Material {' +
'    diffuseColor ' + R() + ' ' + R() + ' '
+ R() +
'    }' +
'    }' +
'    geometry Box {' +
'    size ' + rand3 + ' ' + rand4 + ' ' +
rand5 +
'    }' +
'    }' +
' ]' +
```

```
'}';
 newNode =
Browser.createVrmlFromString(string);
ROOT.children[i] = newNode[0];
  }
}
]]>
</Script>
```

3.11 X3D Kullanım Alanları

X3D Birliği, standart için yoğun beklentiler içerisinde olan uygulama marketlerindeki X3D-Tabanlı açık standartların geliştirilmesini hızlandırmak için bazı alanlara odaklanmıştır. Birlik birincil olarak CAD (CDF), Tıbbi (MedX3D) ve Görsel Benzeştirim (XMSF) marketlerine ve bir çok ikincil markete (Coğrafi-Uzaysal, Eğitim, Teknik Yetiştirme ve Belgeleme, Eğlence ve Oyunlar, ...) odaklanmıştır.

CAD (CDF): Milyarlarca dolar CAD ve teknik ürün bilgisine yatırım olarak sağlanmaktadır. Ancak, CAD uygulamaları tarafından yaratılan veriler, işletmelerdeki diğer kullanıcılar arasında paylaşımı zordur. 3B verinin , örneğin CAD mühendislik dosyaları, satış ve pazarlama veya eğitim için diğer uygulamalarla bütünleştirilmesi zaman alıcı ve zor bir iştir.

X3D CAD başlangıç açık standartları müşterilerin karmaşık 3B ve teknik verilere erişmelerini ve bu verileri işletmede diğer genel masaüstü uygulamalarına bütünleştirmelerine izin verecektir. CAD ve mühendislik dışındaki meslekteki kişilerin bu canlandırma,materyal ve kaplama içeren grafiksel veriye erişmesi üretkenliğin artmasına, maliyetlerin azalmasına ve yeni gelir kaynaklarına neden olacaktır. Bu CAD verisinin değerinin arttırıp, diğer alanlardaki maliyetlerin azaltacaktır. Uygulamalar; müşteri görselleştirme, tasarım iletişimi, eğitim, teknik belgeleme, satış ve pazarlama ve müşteri desteğini içerir.

CAD3D Çalışma grubu bir dosya biçimi ve veri aktarım süreci belirlemiştir. Biçim, “CAD Distillation Format” (CDF), CAD verisinin

yayın ve etkileşimli ortam için açık biçime çevrilmesine olanak sağlar. Bu süreç yüzeylerin CAD olmayan ortamlardaki daha genel yapılara çevrilmesine yarayan araçlar barındıran açık ana çatı ardışık düzeni de içerir.

Temel Noktalar

1. Yüksek karmaşık veriden düşük karmaşık veriye doğrudur
2. Düşük maliyetli uyarlama
3. Genişletilebilir, açık teknolojiden kaldıraçlanmış açık bir teknoloji
4. Telifsiz
5. Pazara çıkış zamanı ve maliyetleri düşürmek için, satış ve pazarlama, hizmet süreçleri, üretim ve tasarım arasında iletişim kanalı yaratır.

Uygulama Alanları

1. Elektronik eğitim kılavuzları
2. Canlandırmalar
3. Özel biçim tanımlayıcılar (Örnek araba), ürün incelemeler (Örnek giyim)
4. Mimari, Mühendislik, Yapı: Bakım, güvenlik, ön ve son inceleme, yol haritası

Görsel Benzeştirim ve İletişim: Genişletilebilir modelleme ve benzeştirim ana çatısı ("eXtensible Modeling and Simulation Framework" XMSF) , model tayfı ve yapıcı, sanal ve canlı model içeren benzeştirimler çapında birlikte çalışabilirliği desteklemektedir. Ayrıca var olan benzeştirim ana çatıları ve gittikçe önem kazanan uzaktan öğretim teknolojilerini de bütünleştirmektedir. X3D ve XML web servislerini yeni nesil dağıtık benzeştirimlere olanak sağlamak için kullanmaktadır.

X3D, kendini doğrulayabilen ve güvenilir işleme yeteneği olan "Extensible Markup Language" (XML) kullanarak sahneleri kodlamayı destekler. Gelişmiş X3D yetenekleri Coğrafi-Uzaysal işaretleme, insansı

canlandırma, IEEE Dağıtık Etkileşimli Benzeştirim (“DIS”) protokolünü kullanarak paylaşımlı-durum dağıtımı, prototip yapımı, sunucu üretimli özel arazi, fotoğrafik resim örtme, kartografik veya sözde renkli resimler, algılayıcı gösterim ve varlık hareketi için bütünleştirilmiş fizik, çok sayıda kullanıcı etkileşim yöntemleri ve ölçeklenebilir birbiriyle ilişkili sahnelerin yüklenip, kaldırılması olarak sayılabilir.

XMSF

- Genişletilebilir bir ana çatı uygulanan web tabanlı projeler, yeni nesil modelleme ve benzeştirim (MvB) uygulamalarının ortaya çıkmasına, geliştirilmesine ve birlikte çalışmasına olanak sağlayacaktır
- Bu tip MvB uygulama ana çatıları için işlevsel taktik sistemler desteği eksikliği hissedilen, fakat temel gereksinimdir.
- Genişletilebilir XML altyapı tabanlı diller.

3.12 Xj3D ve Sahne Erişim Arayüzü (SAI)

Kullanıcı kendi kodunu kullanarak X3D sahnesiyle etkileşim kurmak istediğinde, ya Script yerleşik düğümünü veya dışsal uygulamaları kullanacaktır. Bu dışsal uygulamalar Sahne Erişim Arayüzü (SAI “Scene Access Interface”)’nü kullanacaktır. Bu ara yüz, doğrudan sahne çizgesinin kendisini değiştirmeden X3D sahne çizgesini değiştirme protokolüdür.

SAI, tarayıcı ve sahne çizgesini dışsal bir uygulama veya yerleşik Script düğümleriyle değiştirmek için ortak ara yüzdür. Ancak, dışsal uygulama için yazılmış bir kodun hemen betik olarak kullanılması olası değildir. İki ortam da sahne çizgesiyle etkileşim ve erişim için oldukça farklı gereksinim ve yeteneklere sahiptir. X3D SAI belirtimi kodun bağlı olduğu ortam için tek, birleşmiş programatik ara yüz ve kısıtlar sunmaktadır.

Kavramsal olarak SAI X3D sahnelerine aşağıdaki beş erişim şekline izin verir:

- Tarayıcının işlevselliğine erişim

- Tarayıcı eylemleriyle ilgili bildirilerin alınması, örneğin kötü URL, başlama ve kapanma gibi.
- Sahne içerisindeki düğümlerin girdi yetenekli sahalarına olay gönderimi
- Sahne içerisindeki düğümlerin çıktı yetenekli sahalarının gönderilen son değerlerini okuma
- Sahne içerisindeki sahaların değerleri olaylar tarafından değiştirilince bilgilenme

X3D tarayıcı içerisinde SAI hizmetleri kullanılarak erişilebilecek dört ana veri derlemesi vardır: Tarayıcı, yüklü olan sahnenin üst bilgisi, sahne çizgesindeki düğümler, ve düğümlerdeki sahalar. Tanımlama ve belirtimler hizmet terimleri içerisinde çerçevelenmiştir. Bir X3D tarayıcısı, dışsal uygulamaların etkileşimi için bir hizmet kümesi sunar.

X3D SAI belirtiminde bahsedilen hizmetleri kullanan kodlar kullanıcı kodu olarak dikkate alınır. Kullanıcı kodu ya sahne çizgesinin içerisinde veya tarayıcıya dışsal olarak bulunur. Kod bu SAI belirtiminde belirtilen hizmetleri kullanmalıdır, tarayıcıya özgü hizmetleri kullanmamalıdır. Ek olarak, SAI belirtimindeki hizmetler belirli bir tarayıcıya doğal ("native") düğüm genişletmeleri yazmak için tasarlanmamıştır. Bir tarayıcının doğal genişletmeleri gerçekleştirmek için sunduğu kendi malı olan programatik arabirimler SAI belirtiminin bir parçası değildir. Eğer kod bu genişletmeleri kullanıyorsa SAI belirtimi açısından kullanıcı kodu olarak tanımlanamaz.

İçsel eylemlerde görev almak isteyen kullanıcı kodunun sahne çizgesindeki temsilcisi düğüm kapsama düğümüdür. Kullanıcı kodunun yaşam döngüsü kapsama düğümü tarafından yönetilmektedir. Kapsama düğümü canlandığı zaman, kullanıcı kodu da canlanır. Kapsama düğümü kaldırıldığı ve canlı sayılmadığı zaman bu düğüm tarafından kapsanmış olan kullanıcı kodu da sonlandırılmalıdır. Kullanıcı kodunun kapsayan düğüme referansı olması bu düğümün yaşamını devem ettirmesini sağlayamaz. Tarayıcı kapsayan düğüm canlı değilken son söz sahibidir. Kullanıcı kodunun tek bir örneği bir çok kapsama düğümü örneği tarafından paylaşılabilmektedir.

Bir uygulama, X3D tarayıcının parçası olmayan dışsal süreçtir. Bu uygulama X3D tarayıcıya tarayıcının istekleri doğrultusunda bir çeşit bağlantı yapar. Uygulama X3D tarayıcının bir parçası olarak varolmaz. Uygulama X3D tarayıcıdan bağımsız bir şekilde, farklı makineler üzerinde de yer alabilir. Uygulama yeni tarayıcı örneği yaratma işleminden sorumlu olabilir veya kendisini hali hazırda var olan bir tarayıcıya bağlayabilir.

Kullanıcı kodu ve X3D tarayıcı arasındaki tek bağlantının yaşamını oturum tanımlar. Tek bir tarayıcının birden fazla oturuma eş zamanlı olarak hizmet vermesi mümkündür (Örneğin tek bir sahnedeki birden fazla betik düğümleri).

Tek bir uygulama, çok sayıda tarayıcıya bir çok farklı oturum içerebilir, ancak tek betik düğümü içermemelidir. Dışsal uygulamalar ile birçok X3D tarayıcı arasındaki bir çok eş zamanlı oturuma izin verilebilir. Ancak ferdi gerçekleştirimler bu tip çoklu eş zamanlı oturuma kısıtlamalar getirebilir.

Tarayıcı, etkin X3D sahne çizgesi için temel kapsama düzeneğidir. Tarayıcı tüm sahne çizgesini içerir, ayrıca bu sahne çizgesini dinamik olarak değiştirebilmek için en temel çekirdek yetenek kümesini sağlar.

Kullanıcı makinesinde bir çok X3D tarayıcısı eş zamanlı olarak çalışabilir. Bu yüzden her tarayıcı oturum içerisinde tekil tanımlayıcı ile temsil edilmelidir. Bu tanımlayıcı tek tarayıcı örneğinin farklı istekleri için aynı olmalıdır. Bu aynı tarayıcıya erişen iki uygulamanın kesin bir şekilde bilgi paylaşmasını sağlar.

Tarayıcı işlevselliğinin kullanımını gerektiren herhangi bir eylem hizmet isteğini tarayıcı tanımlayıcısıyla belirtmelidir.

Sahne, tek bir X3D sahne çizgesini ve bu çizge hakkındaki tüm bilgiyi temsil eder. Sahne X3D dosyasının programatik eşidir. Düğümler, rotalar, proto tanımlamaları, içe aktarmalar ve dışa aktarmalar, ve geçerli X3D dosyasının içerebileceği her türlü bilgiyi içerebilir. Tarayıcı belli bir zamanda bir veya daha fazla sahne içerebilir. Örneğin bir sahne başka bir sahneyi içermesine olanak sağlayan Inline düğümünü kullanabilir.

Sahnenin canlı veya tarayıcı içerisinde çalışıyor olması gerekmemektedir. Kullanıcı tarayıcıya eklenmemiş yeni bir sahne

yaratabilir ve programatik olarak düğüm, rota gibi bilgilerle doldurabilir. Bu sahne doğrudan kaynak dosyasını yayınlamak için güzel yazıcı gibi araç programlarına veya tarayıcıdaki sahneyi değiştirmek için aktarılabilir.

Sahne çizgesindeki en küçük etkileşim nesnesi düğümdür. Bir düğüm çıkarılabilir, eklenebilir. Her düğüm tekil tanımlayıcı ile tanımlanmıştır. Bu tanımlayıcı oturum boyunca tekildir. Şöyle ki, tek ir tarayıcı birden fazla uygulamaya hizmet sunuyor olabilir, bu yüzden tüm düğüm tanımlayıcıları oturumun yaşamı boyunca tekil ve değişmez olmalıdır. Bu, iki dışsal uygulamanın karışıklık yaşamadan birbirleriyle veri paylaşmasına olanak sağlar, bu paylaşılmış veri ile tarayıcıdan hizmet taleplerinde bulunabilirler.

SAI'nin çoğu işlemi bir düğüme referans edinmekle başlar. Bir düğümün referansını almanın bir çok yolu vardır. Düğüm, DEF yapısıyla isimlendirilip daha sonra uygun hizmet kullanılarak veya sahne çizgesi kök düğümden dolaşılarak alınabilir. Düğüm referansı alındıktan sonra bu düğümün tüm sahalarına erişilebilir.

Bir düğüm referansının maruz kaldığı ve farklı yeteneklere sahip olduğu bir yaşam döngüsü vardır. Yaşam döngüsü şu şekildedir:

- Yaratma: Düğüm tarayıcı içsel araçları tarafından tüm saha değerleri var sayılan değerlerine ayarlanmış şekilde ilklenir.
- Ayarlama: Saha değerleri varsayılan değerlerinden gerekli değerlere ayarlanır.
- Gerçeklenmiş: Düğüm sahne çizgesinde ve/ya betiklemeye katılır.
- Yok edilmiş: Düğüm artık sahne çizgesinin bir parçası değildir ve betik seviyesinde kendisine bir referans yoktur.

Okuma ve yazma için saha erişimi, düğümün durumuna bağlıdır. Çizelge 3.2'de bu durumlar ve yetenekler listelenmiştir.

Çizelge 3.2: Düğümün yaşam döngüsündeki saha erişim yetenekleri

Saha Tipi	Yaratma	Ayarlama	Gerçeklenmiş	Yok edilmiş
initializeOnly	yok	okuma/yazma	yok	yok
inputOnly	yok	yok	yazma	yok
outputOnly	yok	yok	okuma	yok
inputOutput	yok	okuma/yazma	okuma/yazma	yok

Ayarlamadan gerçeklenmiş durumuna geçiş içsel veya dışsal bir şekilde olabilir. Varolan bir hizmet isteği kullanıcının ayarlama durumunun bittiğini bildirmesini, böylece düğümün içsel yapısının gerektirdiği her şeyi tamamlamasını sağlar. Geçiş kullanıcının eylemlerine bağlı olarak içsel de olabilir. Bu noktada kullanıcı düğümün sahalarını ayarlamak yerine referansı ile bir şeyler yapabilir, düğüm kendisini gerçeklenmiş durumuna geçirebilir. Örneğin, kullanıcı bir Box düğümü yaratır, boyutunu ayarlar (size sahası), bir Shape düğümü hazırlar ve Box düğümünü hemen Shape düğümüne ekler, böylece Box düğümünün durumu gerçeklenmiş durumuna geçmek zorundadır, Shape düğümü ise ayarlama durumunu korur.

Düğüm tanımlayıcıları ayrıca boş düğüm temsil etmek için de kullanılabilir. Bir boş SFNode veya MFNode sahası NULL değeri ile temsil edilmektedir. Boş MFNode sahaları için eldeki düğümlerin sayısı sahanın değeri NULL iken sıfır olmalıdır.

Düğümler içerisinde ferdi sahalar vardır. Bir düğümü doğrudan değiştirmek olası değilken, sahalar farklı özelliklerin doğrudan değiştirilmesinin yöntemidir. Düğümün özelliklerini düğümün kendisinden ayrı varlıkları olarak doğrudan değiştirmek mümkün değildir (Sahalar kendilerini kapsayan düğümler dışında var olmazlar).

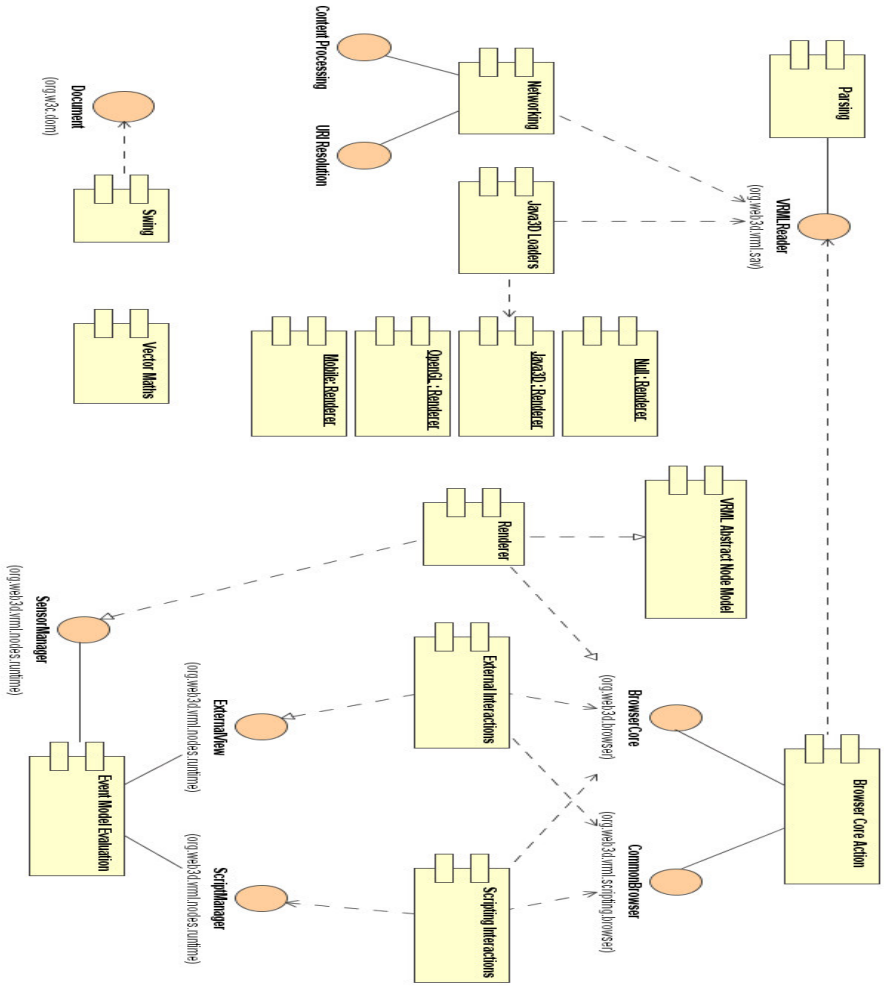
Düğümün çıktı yetenekli sahaları bu sahaya referansı olan uygulamalar tarafından okunabilir. Okunan değer sahanın tipine bağlıdır, kullanılan dil ve protokole uygun bir biçimde sunulmaktadır.

X3D sahne çizgesinde herhangi bir veri olayların kullanımıyla taşınmaktadır. Uygulamalar, X3D sahne çizgesinden olay almak için kaydolabilir ve yeni olaylar üretebilir. Olaylar fani olarak değerlendirilir ve sadece ilgili eylem gerçekleşince üretilir. Olaylar kaydedilmez ve olaylarla ilgisini kaybetmiş partiler olayları alamazlar. Örneğin dünya yüklendikten sonra bağlanan bir uygulama Initialize olayını almayacaktır.

3.13 Xj3D Araç Takımı

Xj3D Web3D Birliğinin VRML97 ve X3D içeriği için tamamen Java'da yazılmış bir araç takımı yaratılmasına odaklanmış Kaynak Çalışma Grubunun (Source Working Group) bir projesidir. Bu araç takımı VRML içeriğinin özel uygulamalarda içe aktarılmasında veya tam bir tarayıcı yaratılmasında kullanılabilir. Xj3D temelinde Java 3D veya JOGL kullanan yüksek seviyeli bir UGA'dir. Böylece kullanılan alttaki UGA'ya bağlı olarak farklı yetenekler ve farklı başarım sonuçları elde edilebilir.

Bu projenin başlangıç güdüsü Java3D API için bir dosya yükleyicisi yaratmak olup, Sun Microsystems tarafından Web3D birliğine kod verilmesiyle başlamıştır. Zamanla projenin gereksinimleri artıp bugünkü haline gelmiştir. Xj3D günümüzde X3D belirtimlerinin doğrulanması ve denenmesi için bir çalışma ortamı sunmaktadır. Ayrıca Java3D kökünden daha uzağa erişip farklı imge oluşturucularla çalışır hale gelmiştir (Bkz. Şekil 3.4).



Şekil 3.4: Xj3D Mimari Modeli (Xj3D, 2004)

Xj3D Java gerektiren bir araç takımıdır. Bu nedenle Xj3D ile çalışabilmek için ilk olarak Java ortamının hazırlanmış olması gerekiyor. Xj3D'nin donanım olarak gereksinimleri asgari şu şekildedir:

- 3D grafik kartı. TNT64 veya daha iyisi. Çoklu doku kaplama gibi karmaşık içerik için Geforce2 veya daha iyisi gerekmektedir.
- 128MB Hafıza
- PII 600MHz ve üstü işlemci

Java ve masaüstü uygulama geliştirmek için şu yazılım gereksinimleri vardır:

- MS Windows, Solaris, Linux, IRIX işletim sistemi
- J2SE JDK 1.4.2

- Java3D 1.3.1
- JOGL 1.1 beta 03
- JOAL 1.1 beta 04
- JInput 1.1 beta 04
- Java Media Framework 2.0
- Mozilla Rhino 1.5R4-1

Gömülü veya mobil sistemler için:

- J2ME CDC
- GL4Java
- JUnit 3.7

Bu gereksinimler karşılandıktan sonra ihtiyaca uygun olan Xj3D kurulumunu indirip bilgisayarda kurmak gereklidir. Java kurulu bir ortamda JRE içermeyen kurulum indirilip kurulmaktadır. Java olmayan sistemler için JRE içeren kurulum kurulmalıdır. Kurulumdan sonra Xj3D uygulamalarda kullanılmaya başlanabilir. Bunun için Xj3D kurulum dizininin altındaki jars dizini içerisindeki jar kütüphaneleri uygulamanın classpath'ine eklenmelidir.

3.13.1 Xj3D Kullanımı

Xj3D bazı kütüphanelerin sınıf içerisine aktarılması ve kütüphane içerisindeki bazı sınıfların kullanılmasıyla uygulamayla bütünleştirilmektedir. Uygulamamıza 3B pencere eklemek için aşağıdaki kodu örnek olarak gösterebiliriz:

```
Container contentPane = getContentPane();
// SAI bilesenini yaratalim
X3DComponent x3dComp =

BrowserFactory.createX3DComponent(null);
// Bu SAI bilesenini penceremize ekleyelim
```

```
    JComponent x3dPanel =

    (JComponent)x3dComp.getImplementation();
    contentPane.add(x3dPanel,
BorderLayout.CENTER);
```

SAI erişimi için gereken tüm sınıflar org.web3d.x3d.sai paketi altında bulunmaktadır. Penceremize SAI bileşenini ekledikten sonra yapacağımız iş eklediğimiz SAI bileşeninin tarayıcı nesnesine dünya yüklemek olacaktır:

```
// Dissal tarayiciyi alalim
ExternalBrowser x3dBrowser =
x3dComp.getBrowser();
```

Dışsal tarayıcıda ilgili hizmetleri kullanarak dünyanın yüklenmesini gerçekleştiririz:

```
// Dosya yukleyerek X3D sahnesi olusturalim
X3DScene mainScene =
x3dBrowser.createX3DFromURL(new String[] {

"KirmiziKutu.x3d" });
// Var olan sahneyi yeni yukledigimiz sahne
ile degistirelim
x3dBrowser.replaceWorld(mainScene);
```

Bu şekilde kendi uygulamamıza KirmiziKutu.x3d dosyası içerisinde tanımlanmış olan içeriği yüklemiş olduk.

```
#X3D V3.0 utf8
PROFILE Interactive
DEF TS TimeSensor {
  cycleInterval 10
  loop TRUE
```

```
}
DEF TG Transform {
  rotation 0 1 0 0.78
  children Shape {
    geometry Box {}
    appearance Appearance {
      material DEF MAT Material {
        diffuseColor 1 0 0
      }
    }
  }
}
DEF PI PositionInterpolator {
  key [ 0 0.25 0.5 0.75 1 ]
  keyValue [
     0 0 0
    -1 0 0
    -1 1 0
     0 1 0
     0 0 0
  ]
}
ROUTE TS.fraction_changed TO PI.set_fraction
ROUTE PI.value_changed TO TG.translation
```

SAI'nin tek amacı dünyaların yüklenmesiyle sınırlı değildir. Asıl işlevi yüklenmiş olan dünyaları dinamik olarak değiştirmek ve dünyada gerçekleşen olayları takip etmektir. Bunun için SAI'nin sahnelerdeki

sahaları değiştirmesi gerekmektedir. Bunun nasıl yapıldığına ilişkin bir örnek aşağıdaki sınıfta gösterilmiştir:

```
import java.awt.*;
import java.util.HashMap;
import javax.swing.*;
import org.web3d.x3d.sai.*;
/**
 * SAI'de sahne yuklemeyi ve saha
degistirmeyi
 * gosteren basit bir ornek
 */
public class SahaDegistirme extends JFrame {
/**
 * Sinifin yapicisi
 */
public SahaDegistirme() {

setDefaultCloseOperation(EXIT_ON_CLOSE);
     Container contentPane =
getContentPane();


     // Tarayici parametrelerini ayarlayalim
     HashMap istenenParametreler = new
HashMap();


     // SAI bilesenini yaratalim
     X3DComponent x3dComp = BrowserFactory
```

```
.createX3DComponent(istenenParametreler);

     // SAI Bilesenini pencereye ekliyoruz
     JComponent x3dPanel = (JComponent)
               x3dComp.getImplementation();
     contentPane.add(x3dPanel,
BorderLayout.CENTER);

     // Bilesenden islemlerimizi yapacagimiz
     // tarayiciyi aliyoruz
ExternalBrowser x3dBrowser =
x3dComp.getBrowser();

     setSize(600, 500);
     this.setVisible(true);

     // Dosya yukleyerek sahneyi
olusturuyoruz
     X3DScene mainScene = x3dBrowser
               .createX3DFromURL(new
String[]                          {
"KirmiziKutu.x3d" });

// Tarayicidaki su anki sahneyi dosyadan
// yukledigimiz sahne ile degistiriyoruz
x3dBrowser.replaceWorld(mainScene);
// DEF ile MAT olarak tanimladigimiz dugumu
getiriyoruz
```

```
X3DNode mat = mainScene.getNamedNode("MAT");
if (mat == null) {
     System.out.println("MAT isimli dugum
          bulunamadi");
     return;
}
// Bu dugum Material dugumu, icerisinde
// diffuseColor sahasini aliyoruz
SFColor color = (SFColor)
mat.getField("diffuseColor");
// Ve kutunun rengini mavi yapiyoruz
float[] blue = { 0, 0, 1 };
color.setValue(blue);
}
}
```

Xj3D yardımıyla dinamik olarak düğümler yaratabilir ve bu düğümleri sahneye ekleyebiliriz. Yüklenmiş olan sahnenin profilinin desteklediği her düğümü rahatlıkla yaratabilir ve sahneye istediğimiz bir noktaya ekleyebiliriz. Örnek olarak verilen dizgede bir kutu düğümü yaratılmakta ve sahneye kök düğüm olarak eklenmektedir.

```
// Kutuyu yerlestirecegimiz Shape dugumunu
yaratalim
X3DNode shape =
mainScene.createNode("Shape");
// Yarattigimiz Shape nesnesinin geometry
sahasini
// alalim
```

```
// bu saha icerisinde dugum(ler) tutmaktadir
SFNode shape_geometry = (SFNode)
          (shape.getField("geometry"));
// bir Box dugumu yaratalim
X3DNode box = mainScene.createNode("Box");
// yarattigimiz Box dugumunun atasini daha
onceden
// elde ettigimiz Shape dugumu olarak
atayalim
shape_geometry.setValue(box);
// Sahnemize Shape dugumunu ekleyelim
mainScene.addRootNode(shape);
```

Bu şekilde Xj3D kullanım alanları çoğaltılabilir. Kullanıcılar Xj3D'nin genişletilebilir yapısı sayesinde rahatça istedikleri gibi biçimlendirebilir ve istedikleri projelerde rahatça kullanabilirler. Yeni dosya biçimlerini, yeni profilleri, yeni bileşenleri ve yeni düğümleri isteğe bağlı olarak rahatça ekleyebilir ve kullanabilirler. Varolan arabirimler kullanmak için yeterli olmazsa kullanıcılar isteklerine göre Xj3D'yi biçimlendirebilirler.

4 YAPAY ZEKA VE GENETİK ALGORİTMALAR

4.1 Yapay Zeka

Zeka, insanın düşünme, akıl yürütme, nesnel gerçekleri algılama, kavrama, yargılama, sonuç çıkarma, soyutlama, öğrenme yeteneklerinin tümü. Ayrıca Soyutlama, öğrenme ve yeni durumlara uyma gibi yetenekler de zeka kapsamı içindedir.Yapay zeka ise, bu özelliklere sahip organik olmayan sistemlerdeki zekadır.

Yapay zeka kabaca; bir bilgisayarın ya da bilgisayar denetimli bir makinenin, genellikle insana özgü nitelikler olduğu varsayılan akıl yürütme, anlam çıkartma, genelleme ve geçmiş deneyimlerden öğrenme gibi yüksek zihinsel süreçlere ilişkin görevleri yerine getirme yeteneği olarak tanımlanmaktadır (Nabiyev, 2003).

Yapay zeka dört kategoriye ayrılabilmektedir (Russel, 2003):

- İnsan gibi düşünen sistemler
- İnsan gibi davranan sistemler.
- Mantıklı düşünen sistemler.
- Mantıklı davranan sistemler.

Yapay zekanın temelleri bir çok farklı alanlardan beslenmektedir. Felsefe, Matematik, Algoritma, Ekonomi, Psikoloji, Bilgisayar Mühendisliği, Sinir bilimleri, Kontrol teorisi ve Sibernetik ve Dilbilim başlıcaları olarak sayılabilmektedir (Russel, 2003).

Yapay Zeka kavramının geçmişi modern bilgisayar bilimi kadar eskidir. Fikir babası, "Makineler düşünebilir mi? " sorusunu ortaya atarak Makine Zekasını tartışmaya açan Alan Mathison Turing'dir. 1943 yılında İkinci dünya savaşı sırasında Kripto Analizi gereksinimleri ile üretilen Elektro-Mekanik cihazlar sayesinde Bilgisayar Bilimi ve Yapay Zeka kavramları doğmuştur.

Alan Turing, Nazi'lerin Enigma makinesinin şifre algoritmasını çözmeye çalışan matematikçilerin en ünlenmiş olanlarından biriydi. İngiltere, Bletchley Park'ta şifre çözme amacı ile başlatılan çalışmalar,

Turing 'in prensiplerini oluşturduğu bilgisayar prototipleri olan Heath Robinson, Bombe ve Colossus bilgisayarları, Boole cebrine dayanan veri işleme mantığı ile Makine Zekası kavramının oluşmasına sebep olmuştu.

Modern bilgisayarın atası olan bu makineler ve programlama mantıkları aslında insan zekasından ilham almışlardı. Ancak sonraları, modern bilgisayarlarımız daha çok uzman sistemler diyebileceğimiz programlar ile gündelik hayatımızın sorunlarını çözmeye yönelik kullanım alanlarında daha çok yaygınlaştılar. 1970'li yıllarda büyük bilgisayar üreticileri olan Apple, Xerox, IBM gibi şirketler kişisel bilgisayar modeli ile bilgisayarı popüler hale getirdiler ve yaygınlaştırdılar. Yapay Zeka çalışmaları ise daha dar bir araştırma çevresi tarafından geliştirilmeye devam etti.

Alan Turing'in adıyla anılan Turing Testi yazılımların insan gibi düşünüp düşünmediğini ölçmektedir. Testin içeriği kısaca şöyledir: birbirini tanımayan birkaç insandan oluşan bir denek grubu birbirleri ile ve bir Yapay Zeka diyalog sistemi ile geçerli bir süre sohbet etmektedirler. Birbirlerini yüz yüze görmeden yazışma yolu ile yapılan bu sohbet sonunda deneklere sorulan sorular ile hangi deneğin insan hangisinin Makine Zekası olduğunu saptamaları istenir. İlginçtir ki,şimdiye kadar yapılan testlerin bir kısmında Makine Zekası insan zannedilirken gerçek insanlar Makine zannedilmiştir.

Loebner Ödülünü kazanan Yapay Zeka Diyalog sistemlerinin dünyadaki en bilinen örneklerinden biri A.L.I.C.E'dir. Carnegie üniversitesinden Dr.Richard Wallace tarafından yazılmıştır. Bu ve benzeri yazılımlarının eleştiri toplamalarının nedeni, testin ölçümlediği kriterlerin konuşmaya dayalı olmasından dolayı programların ağırlıklı olarak diyalog sistemi (chatbot) olmalarıdır.

Türkiye'de de Makine Zekası çalışmaları yapılmaktadır.Bu çalışmalar Doğal Dil işleme, Uzman sistemler ve Yapay Sinir Ağları alanlarında Üniversiteler bünyesinde ve bağımsız olarak sürdürülmektedir. Bunlardan biri, D.U.Y.G.U - Dil Uzam Yapay Gerçek Uslamlayıcıdır (Vikipedi, 2006b).

Yapay zeka'nın araştırma ve çalışma alanları çok geniştir. Bir çok farklı alanda Yapay zeka kullanımıyla karşılaşmak mümkündür. Yapay

zeka ile günümüzde bir çok alanda karşılaşılmaktadır. Bu alanlara aşağıdaki örnekleri verebiliriz:

- Oyunlar (Satranç, dama, strateji, ...)
- Yapay yaşam
- Teorem ispatlama (Prolog, Paralel Prolog, Cebrik Mantık Programlama)
- Doğal dil anlama, işleme ve çeviri
- Bilgi tabanlı sistemler (Bilgi gösterimi, uzman sistemler, bilgi tabanlı benzeştirim, genel bilgi sistemleri, ...)
- Makine öğrenmesi (Bilgi düzeyinde öğrenme, sembol düzeyinde öğrenme, aygıt düzeyinde öğrenme)
- Makine buluşları (Veri madenciliği, bilimsel buluşların modellenmesi)
- Robotik (Görev planlama, robot görmesi)
- Şekil tanıma (Nesne tanıma, optik harf tanıma, ses tanıma, ...)

Yapay zeka'nın temel yaklaşımları Yapay Sinir Ağları, Genetik Algoritmalar, Etmen Sistemler olarak sayılabilir.

4.2 Genetik Algoritmalar

Genetik Algoritmalar evrimsel hesaplamanın bir parçasıdır. Bu alan Yapay Zekâ'nın hızla gelişen bir dalıdır. Genetik algoritmalar Darwin' in evrim teorisinden etkilenerek geliştirilmiştir. Basitçe açıklayacak olursak problemler evrimsel bir süreç kullanılarak bu süreç sonunda en iyi sonucu veren çözüme erişmeye çalışmaktadır. Başka bir ifadeyle çözüm evrimleşmektedir.

Evrimsel hesaplama 1960'larda I.Rechenberg'in Evrim Stratejileri ("*Evolutionsstrategie*") adlı çalışmasında tanıtılmıştır. Daha sonra fikri diğer araştırmacılar tarafından geliştirilmiştir. Genetik algoritmalar John Holland tarafından icat edilmiş ve öğrencileri ve iş arkadaşları tarafından geliştirilmiştir (Holland, 1975). 1992 yılında John Koza, genetik algoritmaları kullanarak programları evrimleştirerek belli işleri yapmakta

kullandı. Bu yönteme “genetik programlama” adını verdi. LISP dilinde programlar Ayrıştırma Ağaçları” (“Parse Tree”) şeklinde ifade edildiği için LISP diliyle geliştirilmiştir. Ayrıştırma Ağaçları genetik algoritmaların çalıştığı temel nesnedir.

4.2.1 Biyolojik Altyapı

Kromozom: Tüm yaşayan organizmalar hücrelerden oluşur. Her hücrede aynı kromozom kümeleri bulunur. Kromozomlar DNA dizileri olup, tüm organizmanın örneği olarak hizmet ederler. Bir kromozom **gen** adı verilen DNA bloklarından oluşur. Her gen belirli bir proteini kodlar. Basitçe, her genin, örneğin göz rengi gibi bir **özelliği** kodladığı söylenebilir. Bir özellik için olası ayarlar, (Örn. Mavi, Yeşil) **alel** olarak adlandırılır. Her gen kromozom üzerinde kendine ait bir konuma sahiptir. Bu konuma **yörünge (“locus”)** adı verilir.

Tüm genetik malzeme kümesine (tüm kromozomlar) **genom** adı verilir. Genom üzerindeki belli gen kümelerine **genotip** adı verilir. Genotipler, doğumdan sonra gelişmeyle fenotiplere - canlının göz rengi, zekâ v.b. fiziksel ve zihinsel özellikleri- dönüşür.

Tekrar Üretim: Tekrar üretim sırasında, yeniden birleşme (veya çaprazlama) ilk önce ortaya çıkar. Atalardan gelen genler yepyeni bir kromozom üretmek için bir araya gelirler. Bu yeni yaratılmış nesil daha sonra mutasyona uğrayabilir. **Mutasyon** DNA elemanlarının değişmesidir. Bu değişimler genellikle atalardan gen kopyalanması sırasındaki hatalardan kaynaklanır. Bir organizmanın uygunluğu (“fitness”) organizmanın yaşamındaki başarısıyla (hayatta kalma) ölçülür.

4.2.2 Arama Uzayı

Eğer bir problemi çözüyorsak, genellikle çözümler arasındaki en iyi olanını arıyoruz demektir. Mümkün tüm çözümlerin uzayına (istenen çözümün aralarından bulunduğu çözümler kümesi) **arama uzayı (durum uzayı)** adı verilir. Arama uzayındaki her nokta bir olası çözümü temsil eder. Her olası çözüm değeri (uygunluğu) ile problem için

"işaretlenebilir". Genetik algoritmalar yardımıyla arama uzayındaki olası çözümler arasından en iyi çözümü araştırırız.

Çözümü aramak, arama uzayında aşırı noktaları (azami veya asgari) aramak ile aynı anlamdadır. Zaman zaman arama uzayı iyi tanımlanmış olabilir, ama bu arama uzayında sadece bir kaç noktayı biliyor olabiliriz. GA kullanma sürecinde, çözüm bulma süreci diğer noktaları (olası çözümleri) evrim sürdükçe üretir.

Sorun, arama çok karmaşık olabilir. Nereden başlanacağı veya nereye bakılacağı bilinemeyebilir. **Uygun çözüm**ün bulunması için birçok yöntem vardır, fakat bu yöntemler en iyi çözümü üretmeyebilir. Bu yöntemlerin bazıları, *tepe tırmanma ("hill climbing"), yasak arama ("tabu search"), benzetimli tavlama ("simulated annealing")* ve **genetik algoritmalar**dır. Bu yöntemler sonucu bulunan çözümler genellikle iyi çözümler olarak kabul edilir, çünkü sık sık en iyiyi bulmak ve ispatlamak mümkün değildir.

NP-Zor ("NP-Hard") Problemler: "Geleneksel" yolla çözülemeyecek problem sınıfına bir örnek NP ("Non-deterministic Polynomial Time") problemlerdir.

Hızlı (Çokterimli) algoritmaların uygulanabildiği birçok görev vardır. Ancak algoritmik olarak çözülemeyen bazı problemler de vardır.

Çözüm bulmanın çok zor olduğu önemli problemler vardır, fakat çözüm bulununca bu çözümü kontrol etmek kolaydır. Bu gerçek "NP-Complete" Problemleri ortaya çıkarır. NP Nondeterministic Polynomial anlamına gelir ve bunun anlamı çözüm (Nondeterministic algoritma yardımıyla) "tahmin" edilebilir ve kontrol edilebilir.

NP Problemlere örnek olarak tatmin problemini, gezgin satıcı problemini veya sırt çantası problemini verebiliriz. Daha fazla örnek için "A Compendium of NP Optimization Problems" sitesi incelenebilir. (Kann, 2005)

4.2.3 Genetik Algoritma

Genetik algoritmalar Darwin'in Evrim teorisinden esinlenilerek üretilmiştir. Bir problemin çözümü evrimsel süreç kullanılarak çözülmektedir.

Algoritma **toplum** adı verilen ve **kromozomlarla** temsil edilen bir **çözüm kümesi** ile başlamaktadır. Bir toplumdaki çözümler yeni toplumların üretilmesinde kullanılmaktadır. Bu işlem, yeni toplumun eskisinden daha iyi olacağı umuduyla yapılmaktadır.

Yeni çözümler (yavru) üretmek için alınan çözümler uygunluklarına (fitness) göre seçilmektedir. Daha uygun olan tekrar üretim için daha fazla şansa sahiptir.

Bu süreç belli bir durum (örneğin belli sayıda toplum veya en iyi çözümün gelişmesi) karşılanana kadar tekrar edilmektedir.

Basit Genetik Programlama Taslağı

1. **Başlangıç:** n kromozom oluşan rasgele toplum oluşturulur (problemin olası çözümleri)
2. **Uygunluk:** Toplumdaki her x kromozomu için f(x) uygunluk değeri değerlendirilir.
3. **Yeni Toplum:** Aşağıdaki adımlar izlenerek yeni toplum üretilir;
    a. **Seçim:** Toplumdan uygunluklarına göre iki ata seçilir (daha uygun olanın seçilme şansı daha fazladır)
    b. **Çaprazlama:** Çaprazlama olasılığı ile ataları yeni yavru oluşturmak için birbirleriyle eşleştirilir. Eğer çaprazlama yapılmazsa, yavru ataların tıpatıp aynısı olacaktır.
    c. **Mutasyon:** Mutasyon olasılığı ile yeni yavru üzerinde her yörünge için mutasyon işlemi yapılacaktır.
    d. **Kabul:** Yeni yavru, yeni topluma eklenir.

4. **Değiştir:** Yeni toplum algoritmanın tekrar işlenmesinde kullanılır.
5. **Deney:** Eğer bitiş durumu sağlandıysa, durup toplumdaki en iyi çözüm döndürülür.
6. **Döngü:** Adım 2'ye gidilir. (Obitko, 1998)

Yukarıda da görüldüğü gibi, genetik algoritmanın akışı oldukça kolaydır. Birçok parametre ve ayar farklı problemler için farklı şekillerde gerçekleştirme için vardır. Sorulması sorulan ilk soru kromozomun nasıl kodlanacağıdır. Daha sonra çaprazlama ve mutasyon, GA'nın iki basit işleci adreslenecektir.

Bir sonraki soru çaprazlama için ataların nasıl seçileceğidir. Bu farklı birçok yolla yapılabilir, ancak ana fikir daha iyi ataların daha iyi yavrular üreteceği düşüncesiyle seçilmesidir. Bu şekilde en iyi çözümün kaybedilmemesi için seçkinlik, en iyi çözümün değiştirilmeden yeni nesle aktarılması, böylece en iyi çözümün yaşatılması uygulanabilir.

4.2.4 GA İşleçleri

Genetik algoritma taslağında görebileceğimiz gibi, çaprazlama ve mutasyon genetik algoritmanın en önemli kısımlarıdır. Başarım en çok bu iki işleçten etkilenir. Bu işleçlerden bahsetmeden önce kromozomlardan daha fazla bahsetmek gereklidir.

Kromozomun Kodlanması: Bir kromozom temsil ettiği çözüm hakkında bir şekilde bilgi içermelidir. En çok kullanılan kodlama ikili karakter dizisidir. Bu yöntemle kromozom şu şekilde görülmektedir:

Kromozom 1 : 1101100100110110

Kromozom 2 : 1101111000011110

Her kromozom ikili karakter dizisi şeklinde temsil edilmektedir. Karakter dizisindeki her bit çözümün bir özelliğini temsil eder. Bir başka olasılık tüm karakter dizisinin bir sayıyı temsil etmesidir.

Elbette, birçok başka kodlama yöntemi vardır. Kodlama daha çok çözülen probleme bağlıdır. Örneğin bazı problemler için tamsayı veya

gerçek sayı şeklinde kodlamak gerekirken, bazı problemlerde permütasyon şeklinde kodlamaya ihtiyaç vardır.

Çaprazlama: Kodlamaya karar verdikten sonra, çaprazlama işlemiyle devam edebiliriz. Çaprazlama, atalardaki seçili genler üzerinde işlem yapar ve yeni yavrular oluşturur. Bunun en basit şekli, rasgele bir kesme noktası (çaprazlama noktası) seçip, bu noktadan önceki her şeyi ilk atadan, sonraki her şeyi ikinci atadan alıp birleştirerek yavruyu oluşturmaktır.

Çaprazlama aşağıdaki şekilde gösterilebilir: (| kesme noktasıdır):

Çizelge 4.1: Çaprazlama Örneği

Kromozom 1	11011 | 00100110110
Kromozom 2	11011 | 11000011110
Yavru 1	11011 | 11000011110
Yavru 2	11011 | 00100110110

Çaprazlamanın birçok yolu mevcuttur, örneğin birden fazla kesme noktası seçilebilir. Çaprazlama daha da karmaşık olabilir ve tamamen kromozomların kodlanmasına bağlıdır. Özel problemler için yapılmış özel çaprazlamalar genetik algoritmanın başarımını arttırabilir.

Mutasyon: Çaprazlama işlemi gerçekleştirildikten sonra, mutasyon işlemi yapılır. Mutasyonun amacı, toplumdaki tüm çözümlerin çözülen problemlerin bir yerel uygun değerine düşmesinin önüne geçmektir. Mutasyon işlemi çaprazlama sonucu oluşan yavruyu rasgele değiştirmektedir. İkili kodlamada rasgele seçilmiş bir kaç biti 1'i 0'a, 0'ı 1'e şeklinde değiştirmek bir mutasyondur (Bkz. Çizelge 4.2).

Çizelge 4.2: Mutasyon örneği

Asıl Yavru 1	1101111000011110
Asıl Yavru 2	1101100100110110
Mutasyon Geçirmiş Yavru 1	1100111000011110
Mutasyon Geçirmiş Yavru 2	1101101100110110

Mutasyon tekniği (çaprazlama tekniği de) kromozomların kodlamasına çoğunlukla bağlıdır. Örneğin permütasyon şeklinde kodlamada mutasyon rasgele seçilen iki genin yer değiştirmesi olarak gerçekleştirilir.

4.2.5 GA'nın Parametreleri

Çaprazlama Olasılığı: Bu parametre çaprazlamanın ne kadar sıklıkla yapılacağını belirtir. Eğer herhangi bir çaprazlama yoksa yavrular ataların aynısı olacaktır. Eğer bir çaprazlama yapılırsa yavrular ataların parçalarından oluşur. Eğer çaprazlama olasılığı %100 ise yavrular tamamen çaprazlama ile yapılır. Eğer %0 ise yavrular ataların kromozomlarının aynısına sahip olurlar. (Bu yeni toplumun aynı olduğu anlamına gelmez)

Çaprazlama, yeni kromozomların eski kromozomların iyi parçalarını alıp daha iyi olacakları düşüncesiyle yapılır, ancak eski toplumun bazı parçalarının bir sonraki nesle aktarılması da iyidir.

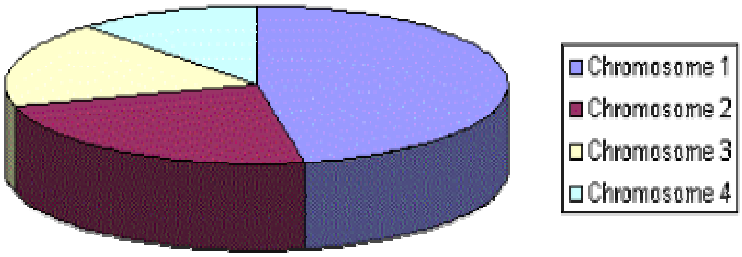
Mutasyon Olasılığı: Kromozom parçalarının ne kadar sıklıkla mutasyon geçireceğini belirtir. Eğer mutasyon yoksa yavrular çaprazlamadan hemen sonra değiştirilmeden üretilir (veya doğrudan kopyalanır). Eğer mutasyon varsa, yavruların kromozomlarının bir veya daha fazla parçası değişir. Eğer mutasyon olasılığı %100 ise tüm kromozom değişecektir. %0 ise hiçbir şey değişmez. Mutasyon genellikle GA'nın yerel aşırılıklara düşmesini engeller. Mutasyonlar çok sık oluşmamalıdır, çünkü GA rasgele aramaya dönüşebilir.

Toplum Büyüklüğü: Toplumdaki kromozom (birey) sayısını belirtir. Eğer çok az birey varsa, GA'nın çaprazlama yapacağı olasılıklar azalacaktır ve arama uzayının çok küçük bir kısmı araştırılacaktır. Eğer çok fazla birey varsa, GA bayağı yavaşlayacaktır. Araştırmalar bazı sınırlardan sonra çok büyük toplumların kullanılmasının yararlı olmadığını göstermiştir, bu problemin daha hızlı bir şekilde çözülmesine yardımcı olmamaktadır.

4.2.6 Seçim

GA Akış şemasından da bildiğimiz gibi, kromozomlar toplum içerisinden çaprazlama için ata olmak için seçilmektedir. Problem bu kromozomları nasıl seçileceğidir. Darwin'in evrim teorisine göre en iyi olan yeni yavruyu yaratmak için yaşamına devam eder. Seçim düzeneklerine rulet tekeri seçimi, Boltzman seçimi, turnuva seçimi, sıralama seçimi, sabit durum seçimi ve diğerleri verilebilir.

Rulet Tekeri Seçimi: Atalar uygunluklarına göre seçilirler. Daha iyi kromozomlar, daha fazla seçilme şansına sahip olanlardır. Toplumdaki tüm kromozomların yerleştirildiği bir rulet tekerini hayal edelim. Rulet tekeri üzerindeki kromozomun yerinin boyutu kromozomun uygunluğuyla orantılıdır (Bkz. Şekil 4.1). Daha uygun olan kromozom daha geniş bir kısma sahip olur.



Şekil 4.1: Rulet Tekeri Kromozom Dağılımı (Obitko, 1998)

Bir bilye rulet tekerine atılmakta ve bilyenin durduğu yerdeki kromozom seçilir. Daha uygun olan kromozomlar böylece daha fazla sayıda seçilecektir.

Süreç aşağıdaki algoritma ile anlatılabilir.

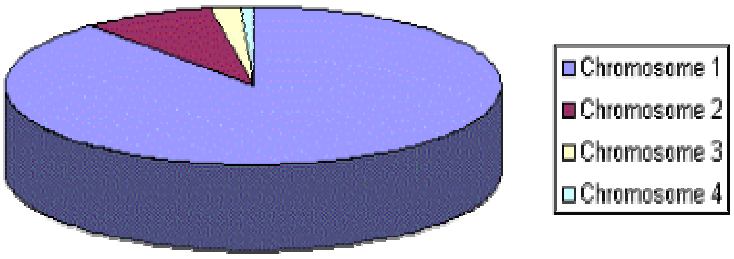
- **Toplam:** Toplumdaki tüm kromozomların uygunluk toplamını hesaplar - S.
- **Seçim:** (0,S) aralığından rasgele bir sayı üretilir - r.
- **Döngü:** Toplum üzerinden gidip 0'dan itibaren uygunlukların toplamını al - s, s r'den büyük olduğu zaman dur ve bulunduğumuz yerdeki kromozomu döndür.

Elbette, aşama bir her toplum için bir kez yapılmaktadır.

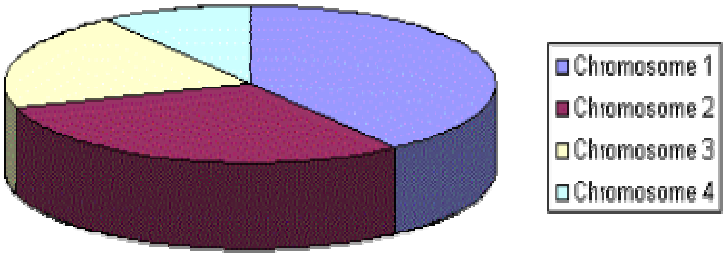
Sıralama Seçimi: Bir önceki seçim düzeneğinde uygunluk değerleri arasında büyük farklar oluşunca problemler ortaya çıkacaktır. Örneğin, eğer en iyi kromozomun uygunluğu diğer tüm kromozomların toplamının %90'ı ise diğer kromozomların seçilme şansı çok azalacaktır.

Sıralama seçimi ilk önce toplumu sıralar ve her kromozom uygunluk değeri olarak sırasını kullanır. En kötü 1 uygunluğunu, ikinci kötü 2,…., en iyi N (toplumdaki kromozom sayısı) uygunluğunu alır.

Şekil 4.2 ve 4.3'te görüldüğü üzere, uygunluk sıraya göre belirlendiği zaman durum değişmektedir.



Şekil 4.2: Sıralamadan önce (uygunluk çizgesi) (Obitko, 1998)



Şekil 4.3: Sıralamadan sonra (sıra numaraları çizgesi) (Obitko, 1998)

Bu şekilde tüm kromozomların seçilme şansı olacaktır. Ancak bu yöntem daha yavaş yakınsama neden olabilir, çünkü en iyi kromozomlar birbirlerinden çok farklı değillerdir.

Sabit Durum Seçimi: Bu özel bir ata seçme yöntemi değildir. Bu tip seçimin ana fikri, toplumun var olan kromozomlarının büyük bir kısmının yeni nesle aktarılmasıdır.

Sabit durum seçimi şu şekilde çalışmaktadır. Her yeni nesilde yüksek uygunluk değerine sahip kromozomlar yeni yavruları oluşturmak için seçilir ve düşük uygunluk değerine sahip yavrular kaldırılarak yerlerine bu yeni oluşturulan yavrular koyulur. Toplumun geri kalan kısmı aynen yeni nesle aktarılır.

Seçkinlik: Seçkinliğin ana fikri daha önce açıklandı. Çaprazlama ve Mutasyon yöntemleriyle yeni bir nesil oluştururken, en iyi kromozomları kaybetme olasılığımız vardır. Seçkinlik, en iyi kromozomların (ya da bir kısmının) ilk önce kopyalanıp yeni nesle aktarıldığı yöntemin adıdır. Geri kalan kromozomlar yukarıda anlatılan yöntemlerle üretilir. Seçkinlik GA'nın başarımını hızlı bir şekilde arttırabilir, çünkü bulunan en iyi çözümün kaybolmasını önler.

4.2.7 Kodlama

Kromozomların kodlanması bir problem çözümüne başlarken sorulması gereken ilk sorudur. Kodlama problemin kendisine yoğun şekilde bağlıdır.

İkili kodlama: İkili kodlama en çok kullanılan yöntemdir, çünkü ilk GA araştırmaları bu kodlama yöntemini kullanıldı ve görece basit bir yöntemdir. **İkili kodlamada**, her kromozom bit (0 veya 1) karakter dizilerinden oluşmaktadır (Bkz. Çizelge 4.3).

Çizelge 4.3: İkili kodlanmış kromozom örnekleri

Kromozom A	101100101100101011100101
Kromozom B	111111100000110000011111

İkili kodlama, fazla olasılıkta kromozomlar verir, bunlara düşük sayıda alel içerenler de dahildir. Ancak, bu yöntem çoğu problem için doğal bir kodlama değildir ve çaprazlama ve/ya mutasyondan sonra düzeltmeler yapılması gerekir.

Örnek Problem: Sırt çantası problemi

Problem: Elimizde değeri ve boyutu verilmiş olan nesneler var. Sırt çantasının kapasitesi verilmiştir. Elimizdeki nesneler sırt çantasına azami sayıda çantanın kapasitesini aşmayacak şekilde yerleştirilmelidir.

Kodlama: Her bit, şeyin sırt çantasında olup olmadığını belirtiyor.

Çaprazlama Yöntemleri:

Tek Noktalı Çaprazlama: Tek bir kesme noktası seçilir, ilk atanın kromozomundan kesme noktasına kadar baştan itibaren alınır ve geri kalan kısım ikinci atanın kesme noktasından sonraki kısmıyla birleştirilip yavrunun kromozomu oluşturulur (Bkz. Şekil 4.4; **11001**011+11011**111** = **11001111**).

Şekil 4.4: Tek Noktalı Çaprazlama (Obitko, 1998)

İki Noktalı Çaprazlama: İki kesme noktası seçilir, kromozomun başından ilk kesme noktasına kadar olan ikili karakter dizisi ilk atadan, iki kesme noktası arasındaki kısım ikinci atadan ve ikinci kesme noktasından sonraki kısım tekrar ilk atadan alınarak yeni yavru oluşturulur (Bkz. Şekil 4.5; **11**001**011** + 11**011**11 = **11011111**).

Şekil 4.5: İki Noktalı Çaprazlama (Obitko, 1998)

Tek biçimli çaprazlama: Bitler atalardan rasgele olarak seçilip kopyalanır (Bkz. Şekil 4.6 1**10**010**11** + **1**10**111**01 = 11011111).

Şekil 4.6: Tek Biçimli Çaprazlama (Obitko, 1998)

Aritmetik çaprazlama: Bazı aritmetik bit işlemleri atalar üzerinde uygulanarak yeni yavru oluşturulur (Bkz. Şekil 4.7 VE işlemi 11001011 + 11011111 = 11001001).

Şekil 4.7: Aritmetik Çaprazlama (Obitko, 1998)

Mutasyon Yöntemleri:

Bit ters çevirme: Seçilen bitler terslerine çevrilir. (Bkz. Şekil 4.8 Yeni Bit= (NOT)Eski Bit işlemi 1**1**001001 => 1**0**0010001)

Şekil 4.8: Bit Ters Çevirme İşlemi (Obitko, 1998)

Permütasyon Kodlama: Permütasyon kodlama, gezgin satıcı problemi veya görev sıralama gibi sıralama problemlerinde kullanılabilir. **Permütasyon kodlamada,** her kromozom **sıra**'da konum belirten numara karakter dizisinden oluşur (Bkz. Çizelge 4.4).

Çizelge 4.4: Permütasyon kodlama ile kodlanmış kromozom örnekleri

Kromozom A	1 5 3 2 6 4 7 9 8
Kromozom B	8 5 6 7 2 3 1 4 9

Permütasyon kodlama, sıralama problemleri için yararlıdır. Bazı problemlerde bazı çaprazlama ve mutasyon türleri için kromozomların tutarlılığı için (örneğin içerisinde gerçek sırayı tutan) düzeltmeler yapılması gerekmektedir.

Örnek Problem: Gezgin satıcı problemi

Problem: Şehirler ve bu şehirler arasındaki uzaklıklar verilmektedir. Gezgin satıcı tüm bu şehirleri dolaşmak zorundadır. Fakat gereğinden fazla dolaşmamalıdır. En küçük dolaşma uzunluğunu verecek olan şehir dolaşma sırası bulunmalıdır.

Kodlama: Kromozom şehirlerin gezgin satıcının dolaşacağı sırasını tutar.

Çaprazlama yöntemleri (Tek Noktalı Çaprazlama): Bir kesme noktası seçilir, kesme noktasına kadar ilk atadan, kesme noktasından sonraki kısımlar da ikinci atadan olmak üzere permütasyonlar kopyalanır. Aynı sayılar olmayan sayılarla değiştirilerek tutarlı yeni yavru elde edilir. Bundan çok farklı, daha fazla sayıda yöntem de uygulanabilir.

(**1 2 3 4 5** 6 7 8 9) + (**4 5 3 6 8** 9 7 2 1) = (**1 2 3 4 5** 6 8 9 7)

Mutasyon Yöntemi (Sıra Değiştirme): İki sayı seçilir ve yerleri değiştirilir.

(1 **2** 3 4 5 6 **8** 9 7) => (1 **8** 3 4 5 6 **2** 9 7)

Değer kodlama: Gerçek sayılar gibi karmaşık değerlerin kullanıldığı problemlerde doğrudan değer kodlama kullanılabilir. İkili kodlamanın bu tip problemler için kullanılması problemlerin zorlaşmasına neden olacaktır.

Değer kodlamada, her kromozom bazı değerlere eşittir. Değerler problemle ilgili herhangi bir şey olabilir. Gerçek sayılar, karakterler veya herhangi nesneler (Bkz. Çizelge 4.5).

Çizelge 4.5: Değer kodlama ile kodlanmış kromozom örnekleri

Kromozom A	1.2324 5.3243 0.4556 2.3293 2.4545
Kromozom B	ABDJEIFJDHDIERJFDLDFLFEGT
Kromozom C	(geri), (geri), (sağ), (ileri), (sol)

Değer kodlama bazı özel problemler için iyi bir seçimdir. Ancak, bu tip kodlamada probleme özgü yeni çaprazlama ve mutasyon yöntemleri geliştirmek gereklidir.

Örnek Problem: Bir sinir ağı için ağırlıkları bulma

Problem: Bir sinir ağı belirlenmiş mimariyle birlikte verilmektedir. Ağdan beklenen değeri almak için sinir ağındaki sinirler arasındaki ağırlıklar istenmektedir.

Kodlama: Kromozomlardaki gerçek değerler sinir ağındaki ağırlıkları temsil eder.

Çaprazlama yöntemi: İkili kodlamadaki tüm çaprazlamalar kullanılabilir.

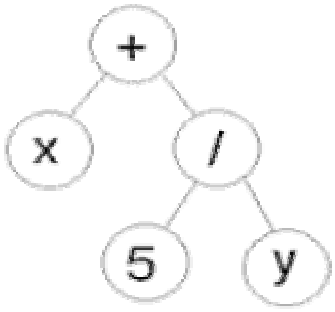
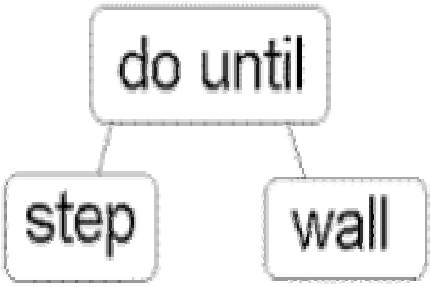
Mutasyon Yöntemi (Küçük bir sayı ekleme -Gerçek sayı kodlama için-): Seçilen değerlere küçük bir sayı eklenir (veya çıkarılır).

(1.29 5.68 **2.86 4.11** 5.55) => (1.29 5.68 **2.73 4.22** 5.55)

Ağaç Kodlama: Ağaç kodlama genellikle evrimleşen program veya ifadeler için kullanılmaktadır. Örneğin genetik programlama için

Ağaç kodlamada her kromozom bazı nesnelerin ağacıdır, örneğin işlevler veya programlama dilindeki komutlar gibi (Bkz. Çizelge 4.6).

Çizelge 4.6: Ağaç kodlama ile kodlanmış kromozom örnekleri

Kromozom A	Kromozom B
	

(+ x (/ 5 y))	(do_until step wall)

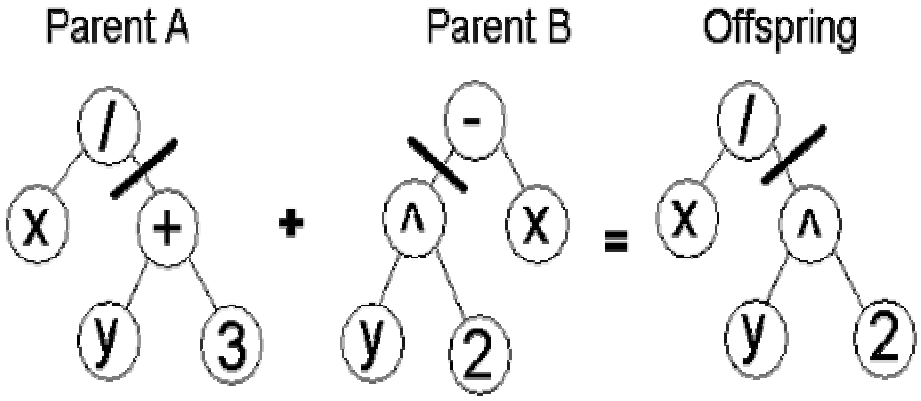
Ağaç kodlama evrimleşen programlar veya ağaç şeklinde kodlanabilecek herhangi diğer yapılar için uygundur. LISP programlama dilinde programların ağaç şeklinde temsil edilmesi nedeniyle LISP bu iş için en çok kullanılan dildir. LISP'te bu ağaçlar kolayca ayrıştırılıp, çaprazlama ve mutasyon kolayca yapılmaktadır.

Örnek Problem: Verilen değer çiftlerini yaklaştıran fonksiyonu bulma

Problem: Girdi ve çıktılar verilmektedir. Görev tüm girdiler için en iyi çıktıları veren fonksiyonun üretilmesidir.

Kodlama: Kromozomlar ağaçta fonksiyonlar şeklinde temsil edilir.

Çaprazlama: Bir kesme noktası her iki atada da seçilir, atalar bu noktada bölünür ve kesme noktasının altında kalan parçalar değiştirilerek yeni yavrular oluşturulur (Bkz. Şekil 4.9).



Şekil 4.9: Ağaç kodlamada kromozomların çaprazlanması (Obitko, 1998)

Mutasyon (İşlev veya Numara değiştirme): Seçilen düğümler yer değiştirilir.

5 ÖNCEKİ ÇALIŞMALAR

5.1 İnternet Üzerinde Üç Boyut ve Web3D Çalışmaları

Web3D çalışmaları birkaç bölümde incelenebilir. Sanal fuarlar, sanal müzeler, sanal mutfaklar, sanal elbise tasarımı gibi çalışmalar daha çok sanal ortam, sanal gerçeklik konularına dayanan çalışmalardır. İnsanların bazı şeyleri sayısallaştırıp çevrimiçi olarak insanların hizmetine sunmasıdır. Ayrıca insanların sanal sosyalleşmesi olarak adlandırabileceğimiz İnternet üzerinde sanal dünyalar, rol yapma oyunları ve çevrimiçi üç boyutlu sohbet ortamları da Web3D çalışmalarına örnek olarak gösterilebilir.

Ürün tanıtımı için VRML yüksek derecede olanak sunmaktadır. Ürünü düz, iki boyutlu resimlerle anlatmak yerine etkileşimli bir şekilde üç boyutlu olarak sunmak daha fazla etki yaratmaktadır. Bu iş alanı için VRML kullanmak gerçek anlamda bir çok zorluklar çıkarabilmektedir. Dauner, Landauer, et. al. “3D Product Presentation Online: The Virtual Design Exhibition” (3B Çevrimiçi Ürün Sunumu : Sanal Tasarım Fuarı) adlı makalelerinde bu zorlukları tartışmıştır. Gerçekleştirdikleri bu sanal fuarda yedi adet içsel tasarım firmasının ürünlerini estetik kalitede sunmuşlardır (Dauner , 1998) . Bu çalışmaya önayak olan çalışmalardan biri Matsushita Sanal Mutfak projesidir. Bu proje Japonya'da insanlara mutfaklarını seçme ve düzenlemede yardımcı olmuştur. Türkiye'de İnternet üzerinden hazır mutfak modellerinin nasıl görüldüğünü müşterilerin incelemesi için buna benzer bir proje iki boyutlu olarak Teba şirketinde de yapılmıştır. Kullanıcılar bu sanal mutfakta istedikleri şekilde mutfağı hazırlayıp nasıl duracağını önceden inceleme olanağı bulmuşlardır.

Sanal gerçeklik alanında bir başka çalışmada Barbieri, et. al. tarafından gerçekleştirilmiş olan ve bilgisayar biliminin tarihçesini üç boyutlu olarak sunan Bilgisayar Bilimi 3B sanal müze çalışmasıdır. Bu çalışmalarını İtalya'da gerçekleştirip, insanların etkileşimli bir şekilde bilgisayar biliminin aşamalarını modern arayüz kullanımıyla incelemesine olanak sağlamıştır (Barbieri , 2001).

Bu çalışmaların bir benzeri bir başka çalışmada (George Lepouros, et. al.) gerçek bir müze içerisinde modern üç boyutlu etkileşimli müzenin sanal bir modeli gerçekleştirilmiştir. Böylece kullanıcılar müzeye gitmeden müzeyi ve içerisindekileri rahatça bilgisayarlarının yardımıyla inceleme olanağı bulmuşlardır. Çalışmalarında on farklı müzeyi sanal ortam olarak gerçekleştiren araştırmacılar sonuç olarak bu tip sanal müzelerin gerçek müze gezmesinden çok daha farklı deneyimlere olanak sağladığını belirtmişlerdir (Lepouras, 2001).

Sanal gerçeklik ve sanal ortam çalışmaları ağırlıklı olarak müzelerin sanal ortamda gezilebilir ve etkileşimli bir şekilde gerçekleştirilmesi üzerinde yoğunlaşmıştır. Bu bir bakıma insanoğlunun çok değerli bilgilerini mikrofilmlere saklamasına benzetilebilir. Yıpranma ihtimali olan tarihi eserleri korumanın ve sonraki kuşaklara aktarmanın belki de en iyi yolu bu eserleri yok olmadan önce sayısallaştırıp tüm insanoğlunun hizmetine sunabilmektir.

Bazı çalışmalarda insanların İnternet üzerindeki deneyimlerini en iyi şekle getirmek için farklı anlayışlar üretmeye yönelmiştir. Örneğin Breiteneder, et. al. çalışmaların iki buçuk boyutlu web bilgi görselleştirmesi diyebileceğimiz “Lookmark” adlı bir anlayış sunmuşlardır. Lookmarklar aynı masaüstünde belgeleri dizdiğimiz gibi web bilgisini de dizmeye dayanmaktadır.Lookmark Java ve Java3d kullanılarak gerçekleştirilmiştir (Breiteneder, 2002).

Önemli bazı çalışmalar hikaye anlatma ve görselleştirme diyebileceğimiz alanda olmaktadır. Örneğin bir tarihi binanın yaşam hikayesini size önüne geldiğiniz zaman üç boyutlu olarak anlatan sistem verilebilir. Bu çalışmanın bir başka nedeni üç boyutlu müzelerde, ortamlarda yeterli yardım olmadığı durumlarda etmen kullanımıyla bu yardımı gereksinim duyanlara sunmaktır. Buna örnek olarak Chittaro, et. al'un üç boyutlu ortamlarda dolaşan insanlara yardımcı olmak için etmen önerisini sunduğu çalışma incelenebilir (Chittaro, 2003).

Bu çalışmaların en çok yapıldığı yerlerden biri Amerika'daki “Naval Postgraduate School” (NPS)'dur. Don Brutzman başta olmak üzere X3D ve Web3D üzerine çalışmalar ve tezler üretilmektedir. Don Brutzman X3D konusunda eğitici makaleler üretmesinin yanı sıra, belirtimlerin üretilmesine katkı sağlamaktadır (Brutzman, 2005). Bu

okulda yapılan en önemli çalışmalardan biri askeri araçlar barındıran bir X3D-VRML üç boyutlu nesne deposu sunmalarıdır. Benim tezimde kullandığım helikopterde bu depodan alınmıştır (Savage, 2006).

Amerika'daki orduyu inceleyerek ihtiyaçlarına uygun benzeştirim programları yapan MOVES enstitüsünde NPS'nin Web3D ile çalışan en önemli merkezdir. Çalıştığı konular 3B Görsel Benzeştirim ve Ağlanmış Sanal Ortamlar, Bilgisayarla üretilmiş otonomi ve Hesaba dayalı idrak ("Computational Cognition"), İnsan başarım mühendisliği ve "Immersive" teknolojiler, Oyun tabanlı benzeştirimin anlaşılması ve Analiz, Savaş modelleme ve analiz olarak sayılabilir (MOVES, 2006).

Web3D ile yapılmış olan önemli benzeştirim çalışmalarından biri de Ay'a inmeyi görselleştiren Ivan Klima'nın "Lunar Landing in Web3D" çalışmasıdır. Bu çalışmada yazar Ay aracının aya inişini benzeştirmek için VRML yardımıyla bir sistem tasarlamıştır. Armstrong ve Aldrin'in aya iniş sırasında kullandıkları araç ve ortam bilgilerinden yararlanılarak bu üç boyutlu benzeştirim programı gerçekleştirilmiştir(Klima, 2000).

Web3D uzaktan öğrenme ve eğitim için de önemli bir teknoloji haline gelmektedir. Bu konuda gösterilebilecek en önemli örnek Leonardo projesidir (Tornincasa, 2002). Bu proje tasarım mühendislerine laboratuar ve çalışmalarda öğrenmeleri gereken bilgileri İnternet üzerinde Web3D kullanılarak nasıl öğretilebileceğini örneklemiştir. Günümüzde tasarım mühendislerinin başarılı olabilmesi için gereken temel eğitim ve öğrenim için belli standartlar vardır. Bu standartları uzaktan öğrenmek isteyen kişilere de sağlamak Web3D ile gerçekten kolaydır. Web3D uzaktan öğrenme için idealdir. Çünkü öğrencilerin fenomen ve nesnelerin gerçekçi görüntülerini çalışmaları ve anlatıcıya çevrimiçi sorular sormaları kolaylaşmaktadır. Web3D kullanımının bir diğer avantajı da pahalı araç ve gereçler satın alınmadan sanal gerçeklik tabanlı çoklu ortam benzeştirimleri yardımıyla bu araç ve gereçleri öğrenmeleri, kullanmalarıdır.

5.2 Yol Planlama ve Genetik Algoritma Çalışmaları

Nikolos et. al. insansız hava taşıtları için üç boyutlu yol planlama çalışması yaptılar. (Nikolos, 2001). Bu çalışmalarında evrimsel algoritmaları kullanarak üç boyutlu bir ortamda, yeryüzü gibi bazı kısıtlar

ve istenen özelliklere göre yol eğrisini hesapladılar. Bu çalışmada vardıkları önemli sonuçlardan biri yolun doğru parçaları şeklinde olmasından ziyade eğrisel bir şekilde olması daha iyi sonuçlar üretiyor.

Hui et. Al. yol bulmanın en çok kullanıldığı alanlardan olan oyunlarda yapay zekanın önemini incelediler(Hui, 2004). Çalışmalarında rasgele yön bulma, engel aşma, hareket öncesi plan gibi konularda yorumlar ürettiler. Oyunlarda yol bulma için en çok Bezier eğrileri ve planlama için Sonlu Durum makineleri kullanılmaktadır.

Hall ilgili çalışmasında (Hall, 2004) yol bulmada varolan teknolojilerin araştırılması, nesne dolu üç boyutlu ortamların tasarlanması, bu ortamın gerçekleştirilmesi, ortam üzerinde testlerin uygulanması ve yol bulma algoritmalarının verimliliği konularında araştırma ve proje yapmıştır. Çalışma sonucunda engeller içeren üç boyutlu ortam yaratılmış, etmenler bu ortamda yol bulmuşlardır, engelleri aşıp hedeflerine ulaşmışlardır.

Reese vd. özellikle oyunlarda yol bulma konusunda çalışma yaptılar (Reese, 1999). Bu çalışmalarında yol bulma tekniklerinin başarımını etkileyen etkenlerin bir sınıflandırmasını yapmışlardır. Bu etkenler oyunun dinamizmi, oyuncuların ve ortamın geometrisi, hareketin tahmin edilebilirliği (edilemezliği), hareket bilim ve zamansal sınırlandırmalar, etkileşim kuralları ve gerçek zamanlı başarım olarak sayılabilir.

5.3 Gezgin Satıcı Problemi (GSP)Çalışmaları

Obitko (Obitko, 1999) genetik algoritmaları tanıtmak için hazırladığı çalışmasında gezgin satıcı probleminin genetik algoritmalarla çözümünü de örneklemiştir. Bu çalışma genetik algoritma eğitiminde kullanılmak üzere bilgilendirici bir çalışmadır. Örnekler Java Appletlerinin kullanımıyla gerçekleştirilmiştir.

Dorigo vd. çalışmalarında (Dorigo, 1997) Karınca Kolonisi Sistemi (KKS) kullanarak dağıtık bir algoritma yardımıyla GSP'ni çözmeye çalışmışlardır. Karınca adı verilen işbirlikçi etmenler GSP için iyi çözümler bulmaya çalışmaktadırlar. Feromen adı verilen bir maddeyle doğrudan olmayan bir iletişim yapan karıncalar bu maddeyi yolların

üzerinde saklayarak en çok geçilen, en iyi yolun bulunmasında yardımcı olmaktadırlar. Çalışmada KKS deneyler yardımıyla anlaşılmaya çalışılmaktadır.

Lutton vd. (Lutton, 1984) çalışmalarında Metropolis Algoritması adını verdikleri algoritmayla olası GSP yollarının dizisini üretmektedirler.

Fogel çalışmasında otomatik keşfe dayanan makine öğrenmesi olarak Evrimsel Eniyilemeyi önermektedir. Doğal bir ortamı GSP yardımıyla yapay olarak benzeştirmektedir. Üç farklı uyumlandırabilir yöntem tanıtılmakta ve analiz edilmektedir. Çeşitli kapsamlar doğrultusunda evrimsel uyarlanma gösterilmektedir (Fogel, 1988).

6 YAZILIM SİSTEMİ TASARIMI VE GERÇEKLEŞTİRİMİ

Projenin gereksinimlerini şu şekilde sıralayabiliriz:

- Bir helikopter olacak (Herhangi bir üç boyutlu uzayda dolaşan birim olarak genelleştirilebilir)
- Bu helikopterin dolaşması gereken (hepsini bir kere) n sayıda hedef var. (Çizgelerde Hamilton yolu)
- Helikopter bu hedefleri en verimli şekilde dolaşmalıdır. Aldığı yol en az düzeyde olmalıdır.

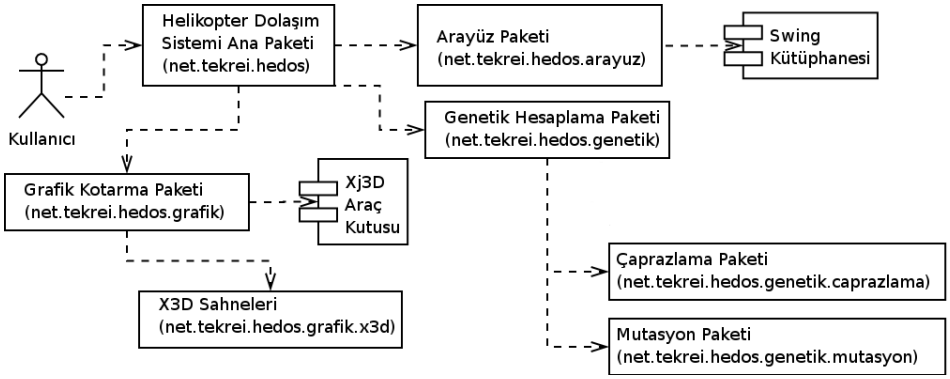
Yukarıdaki istekler incelendiği zaman bu problemin tipik bir Gezgin Satıcı Problemi olduğu görülebilir. Ama en önemli farkı bu dolaşma işinin iki boyutlu şehirler arasında olmayıp, üç boyutlu herhangi bir uzayda olabilmesidir. Bu problemi çözebilmek için ve farklı kriterlere göre sonuçları değerlendirebilmek için Genetik Algoritmaların kullanılmasına karar verilmiştir.

6.1 Tasarım

Helikopter Dolaşım Sistemi (HeDoS) genetik algoritmaların gerçekleştirildiği bir Genetik Hesaplama Birimi, Xj3D ve X3D görevlerinin gerçekleştirildiği Grafik Kotarma (“Handling”) birimi ve grafik kullanıcı arayüzünün gerçekleştirildiği Arayüz Biriminden oluşmaktadır. Her birimler sadece ilgili görevlerini gerçekleştirmektedir. Grafik kotarma biriminde Genetik hesaplama biriminin sorumluluğunda olan herhangi bir iş yapılmaz. Sistemin ana mimari yapısı şekil 6.1'de incelenebilir. Mimariden de anlaşılacağı gibi sistem birbiriyle haberleşen, sorumlulukları dağıtılmış birimlerden oluşmaktadır.

Şekil 6.1: Temel Mimari Birimler

Şekil 6.2'de birimleri paket temelli ve bağımlılıklar şeklinde görebilirsiniz. Kullanıcı alttaki paketleri bilmez, sadece ana paket altındaki (net.tekrei.hedos) ana sınıfı (HedosFrame) çalıştırmak suretiyle programı çalıştırır. Arayüzler Java'nın Swing Uygulama Geliştirme Arayüzü kullanılarak geliştirilmiştir. Genetik hesaplama paketi görevlerine göre 3 alt pakete ayrılmıştır. Genetik algoritmaya bağlı olarak bu 3 paket kendi üzerine düşen ilgili aşamaları gerçekleştirirler.

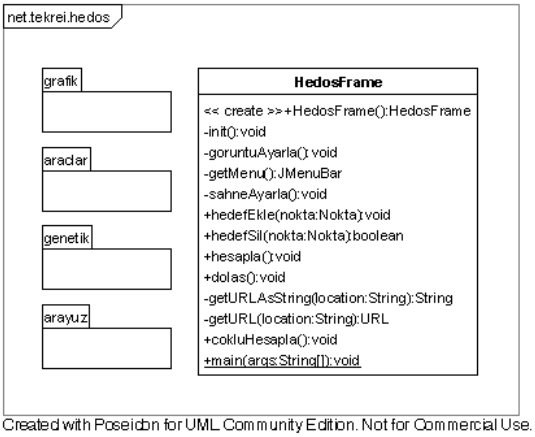


Şekil 6.2: Mimari model ve bağımlılıklar

Grafik kotarma paketi Xj3D araç kutusunu kullanarak yüklenen önceden hazırlanmış X3D sahneleri üzerinde çalışma zamanındaki dinamik değişikliklerin kullanıcı talebine ve genetik hesaplamanın sonucuna göre uygulanmasını gerçekleştirir.

6.2.1 net.tekrei.hedos Paketi

Bu pakette (Şekil 6.3.) çalıştırılacak olan HedosFrame sınıfı ve alt paketler vardır. En üst düzeydeki pakettir. HedosFrame sınıfı programın çalışması için gereken "main" metodunu ve temel başlangıç, kullanıcı etkileşim metotlarını içerir. Ara yüz paketinde SolPanel sınıfından gelen bazı çağrıları diğer sınıflara yönlendirme işlemini de gerçekleştirir. Bu sınıf JFrame sınıfından türetilmiş olup pencere oluşmasını sağlamaktadır.



Şekil 6.3: net.tekrei.hedos paketi

Bu sınıf hedef ekleme (hedefEkle), silme (hedefSil); hesaplama başlatma (hesapla, cokluHesapla) ve dolaşma başlatma (dolas) metotlarını içerir. Bu metotlar alt paketlere çağrı metotlarıdır. SolPanel sınıfındaki düğmeler ve kullanıcı etkileşim araçlarıyla tetiklenirler.

goruntuAyarla metodu arayüzün dili, sağ tarafta gösterilecek olan X3D tarayıcısının ilklenmesi işlemleri ve arayüz yerleştirilmesi işlemlerini gerçekleştirir. Son olarak önceden hazırlanmış olan helikopterin sanal dünyasını yükler.

sahneAyarla metodu ilk olarak yerleştirilecek olan ve ayarlar dosyasında tanımlanmış hedefleri sahneye yerleştirme işlemini gerçekleştirir.

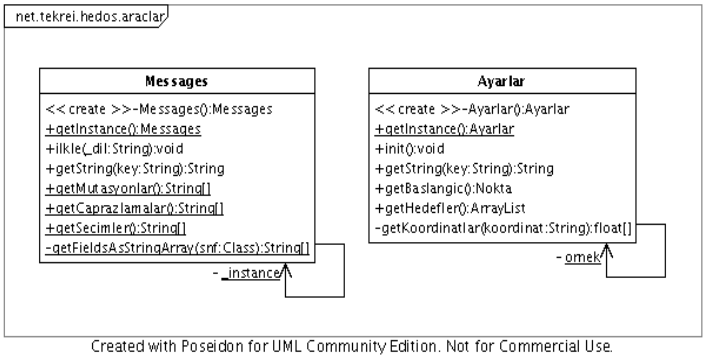
6.2.2 net.tekrei.hedos.arayuz Paketi

Şekil 6.4'te modeli görülen bu paketin amacı grafik kullanıcı ara yüzünü (GKA) soyutlamak ve diğer sınıflarla bağımlılığını azaltmaktır. Bu pakette sol taraftaki genetik parametrelerinin ve kullanıcının sahne ile etkileşiminin bulunduğu paneli gerçekleştiren tek bir sınıf vardır. Bu sınıf Java Swing UGA JPanel sınıfından türetilmiştir. Bu sınıf kullanıcıdan gelen etkileşim taleplerini HedosFrame üzerinden (ilgili parametreleri sistem üzerinde tek bir örneği ("Singleton") saklanan Ayarlar sınıfına kaydederek) ilgili birimlere aktarmaktadır. İlgili birimler de gelen talebi cevaplandırarak kullanıcı etkileşimini sağlamaktadır. Sınıftaki kodun büyük bir çoğunluğu GKA gerçekleştirilmesi işlemi için kullanılmaktadır. Kod tamamen Swing tabanlıdır.

Şekil 6.4: net.tekrei.hedos.arayuz paketi

6.2.3 net.tekrei.hedos.araclar Paketi

Şekil 6.5'te modeli görülen bu paketin amacı programdaki bazı başlangıç ve mesaj dili ile ilgili ayarlamaları gerçekleştirmektir. Tek örnek tasarım deseni kullanılarak gerçekleştirilmiş iki sınıf içermektedir.



Şekil 6.5: net.tekrei.hedos.araclar paketi

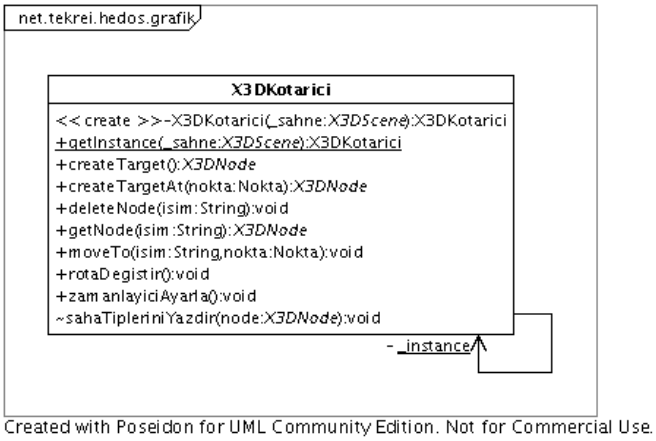
Messages sınıfı programın arayüz ve mesaj dilini ayarlayan metotlar ve programda kullanılan combobox içeriklerini barındırmaktadır.

Ayarlar sınıfı aynı dizinde bulunan *ayarlar.properties* dosyasını açılışta okuyarak, içerisinde tanımlanmış olan bazı başlangıç değerlerini okuyup ayarlamakla ilgilenmektedir. Bu dosyadan genetik algoritma parametreleri ve başlangıç hedeflerini okuyarak gerekli durumlarda bu bilgileri ekran üzerinde göstermek için ve hedeflerin eklenmesi için kullanılmaktadır.

6.2.4 net.tekrei.hedos.grafik paketi

Şekil 6.6'da görülen bu paketin amacı Xj3D bağlantısını sağlamak ve X3D sahnesinin yüklendikten sonra dinamik olarak değiştirilmesini kotarmaktır. İçerisinde şu anki gereksinimler doğrultusunda bu işi yapan X3DKotarici isimli tek bir sınıf vardır. Bu sınıf tamamen Xj3D metodlarını çağırma işlemi yapmaktadır. Görevleri metot isimlerinden de anlaşılacağı üzere sahneye hedef düğüm ekleme, hedef düğüm silme,

düğüm yeri değiştirme ve rota ayarlayıp bu rota üzerinde hareketin başlatılması ve bitirmesi şeklindedir.



Şekil 6.6: net.tekrei.hedos.grafik Paketi

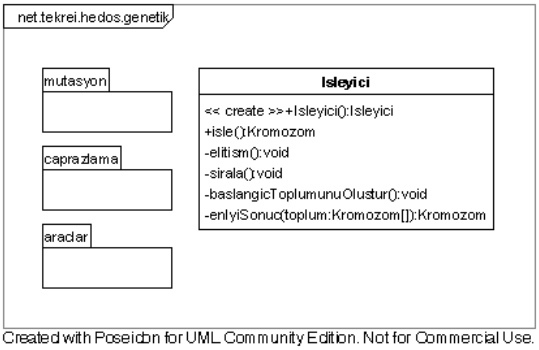
Sınıfın kullanımını kolaylaştırmak için sınıf Tek örnek tasarım deseni kullanılarak çalışma zamanı boyunca bir kere örneklenecek şekilde gerçekleştirilmiştir. Bu şekilde bu sınıfın tek bir şekilde kullanılmasını ve farklı sınıfların farklı örnekleri kullanmamasını sağlıyoruz.

6.2.5 net.tekrei.hedos.genetik paketi

Bu paket (Bkz. Şekil 6.7) genetik hesaplamaların yapılması için gereken paket ve sınıfları barındırmaktadır. Vitrin ("Façade") tasarım desenine özgü sadece bir sınıfın dışarıdan bilinip erişilebildiği bir yapısı vardır. En üst pakette bulunana Isleyici sınıfı arabirim olarak kullanılmaktadır, HedosFrame sınıfı hesapla düğmesine tıklandığı zaman bu sınıfın isle() metodunu çağırarak hesaplama işlemini başlatmaktadır. isle metodunun sırasıyla kullanılacak olan alt paketlerdeki kullanıcının seçimlerine bağlı olarak Caprazlayici, Mutator sınıflarının örneklerini oluşturuyor. Daha sonra tipik genetik algoritmayı işletmeye başlıyor. Hesaplamada aşağıdaki genetik algoritma aşamaları sırasıyla uygulanıyor:

1. Başlangıç toplumu oluşturulur.
2. Uygunluk hesaplanır.
3. Seçkin birey seçilir.
4. Çaprazlayıcı seçilen bireyler üzerinde çaprazlama işlemini gerçekleştirir.
5. Mutator mutasyon işlemini gerçekleştirir.
6. Eğer bitiş kriterleri sağlanmadıysa 2. aşamaya geri dönülür.
7. En iyi sonuç metodun çıktısı olarak döndürülür.

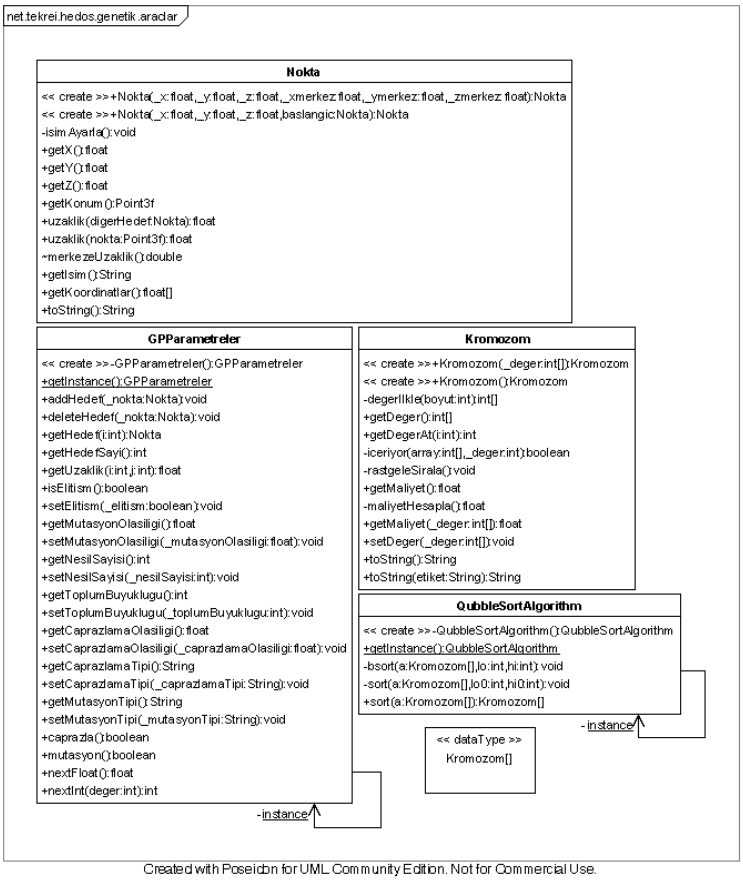
Yukarıdaki algoritmanın ilgili kısımları ilgili sınıflar tarafından kullanıcının seçimlerine göre yapılmaktadır. Algoritma tamamlandığında kullanıcının seçimlerine uygun olan en kısa yol üretiliyor, kullanıcı daha sonra bu yolu *Dolaş* talebiyle üç boyutlu alanda helikopterin hedefleri dolaşmasını izleyerek görebiliyor.



Şekil 6.7: net.tekrei.hedos.genetik Paketi

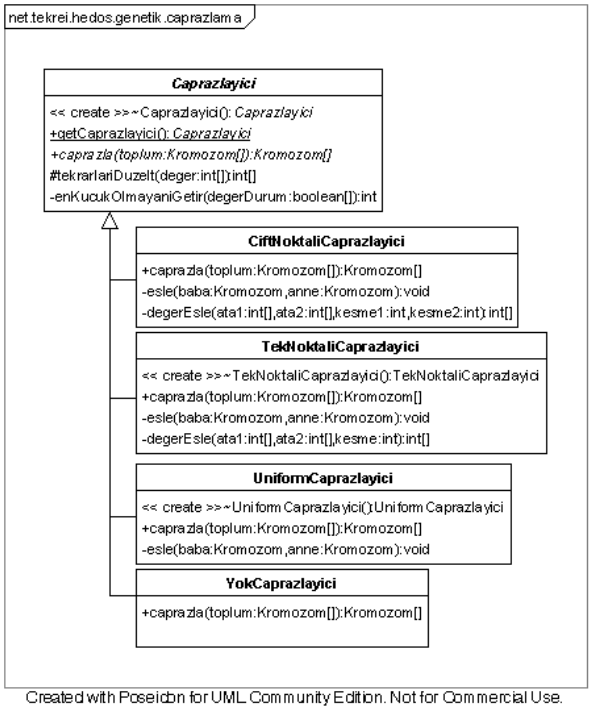
Araçlar alt paketi: Adından da anlaşıldığı gibi bu sınıf genetik algoritmada gereksinim duyulan temel araçları barındırmaktadır. Üç boyutlu konumu temsil eden *Nokta*, bireylerin kromozom yapılarını tutan *Kromozom* ve sıralama işleminde gereksinim duyulan hızlı sıralama algoritmasını gerçekleştiren *QubbleSortAlgorithm* sınıflarını içermektedir. Ayrıca kullanıcının seçimlerinin program boyunca ilgili

sınıflarca rahatça kullanılabilmesi için tek örnek tasarım deseni kullanılmış olan *GPParametreler* sınıfını da barındırmaktadır.



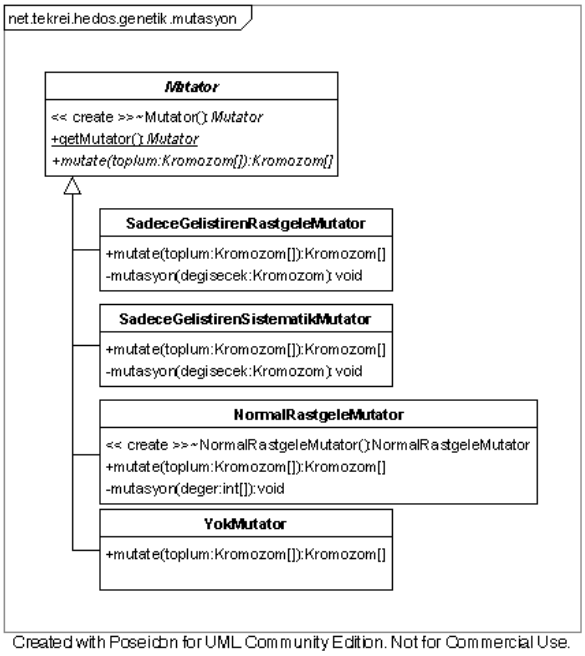
Şekil 6.8: net.tekrei.hedos.genetik.araclar Paketi

Çaprazlama alt paketi: Bu paket genetik algoritmanın çaprazlama aşamasında kullanılan ve bireylerin birbirleriyle eşleşmesini sağlamak için gereksinim duyulan sınıfları Soyut Fabrika tasarım deseni kullanılarak gerçekleştirmektedir. Caprazlama isminde soyut bir sınıftan her çaprazlama tipi için ayrı somut sınıflar türetilmektedir. Kullanıcının seçimine göre Isleyici sınıfı içerisinde örneklenen ilgili Caprazlama sınıfı kullanılmaktadır.



Şekil 6.9: net.tekrei.hedos.genetik.caprazlama Paketi

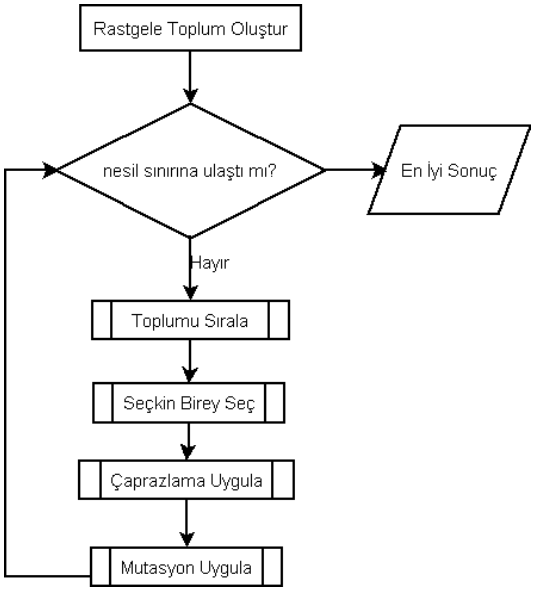
Mutasyon alt paketi: Bu paket genetik algoritmanın mutasyon aşamasında kullanılan ve çaprazlama sonucu oluşmuş olan bireylerin mutasyon geçirmesini sağlayan sınıfları gerçekleştirmektedir. Bu pakette çaprazlama paketinde olduğu gibi kolaylık olması amacıyla soyut fabrika tasarım deseni kullanılarak gerçekleştirilmiştir. Mutator isimli soyut sınıftan her mutasyon tipi için farklı somut sınıflar türetilmiştir. Bu somut sınıflar Isleyici içerisinde Mutator kullanılarak kullanıcının seçimine göre örneklenmektedir. Seçime göre örneklenen sınıf genetik algoritmada kullanılmaktadır.



Şekil 6.10: net.tekrei.hedos.genetik.mutasyon Paketi

6.3 Gerçekleştirim

Kullanıcının girebildiği n sayıda bireyden oluşan bir toplum kullanılmaktadır. Kromozomları kodlamak için 4.2.4. bölümde anlatılan Permütasyon Kodlaması kullanılmaktadır. Kromozomun genleri olarak dolaşma sırasına göre hedef numaraları tutulmaktadır. Şehir eklenmesi ve çıkarılması mümkündür ancak her seferinde hesaplamayı tekrar yapmak gerekmektedir. Şekil 6.11'de algoritmanın gerçekleştirildiği isle metodunun akış şeması incelenebilir.



Şekil 6.11: isle metodunun (genetik algoritma) akış şeması

Uygunluk Hesaplama Formülü: Bireylerin ne kadar başarılı olduğunu bulmamız uygunluklarını bulmamızla mümkündür. Uygunluk formülü de bireyin toplum içerisindeki uzaklık sıralamasındaki yeridir. Bu nedenle her bireyin uzaklığının hesaplanması gerekmektedir. Bu hesaplama aşağıdaki formüle göre gerçekleştirilmektedir:

Uzaklık[j] = $\sum_{0}^{n-1} |P_{i+1} - P_i|$

$$P_{i+1} - P_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}$$

(Üç boyutlu düzlemde iki nokta arasındaki öklid Uzaklığı (Standart Uzaklık ölçütü))

j: Bireyin toplum içerisindeki konumu

i: Bireyin hedef konumu (kromozom içerisindeki her bir genin konumu)

$P_{i+1} - P_i$: Birbirini takip eden iki hedef arasındaki uzaklık

x,y,z : Noktaların koordinat düzlemindeki konum bilgisi

Bu şekilde uzaklık hesaplandıktan sonra hızlı kabarcık sıralama (“quick bubble sort”) algoritması yardımıyla toplum bireyleri uzaklıklarına göre artan bir sırada sıralanmaktadır. Bireylerin bu sıralamadaki konumları uygunlukları olarak kabul edilmektedir. 1. sıradaki birey en uygun bireydir. Son sıradaki bireyde en kötü bireydir.

Çaprazlama çeşitleri: Farklı çaprazlama yöntemleri gerçekleştirilmiştir. Kullanıcının seçimine göre bu çaprazlamalardan ilgili olanı çalışmaktadır. Çaprazlama yapılmadan önce rasgele bir değer üretilip çaprazlama olasılığı ile karşılaştırılmaktadır. Çaprazlama olasılığından küçük bir değer üretilmemişse çaprazlama işlemi yapılmaz.

- Tek Noktalı: Bu çaprazlama şeklinde rasgele seçilen bir noktaya göre iki kromozom eşlenmekte ve yeni bireyler üretilmektedir.
- Çift Noktalı: Bu çaprazlamada tek bir nokta yerine iki nokta rasgele seçilip bu iki noktaya göre kromozomlar eşlenmektedir.
- Tek Biçimli (“Uniform”): Bu çaprazlama şeklinde birinciden, ikinciden, birinciden, ikinciden, ... şeklinde gen alımıyla çaprazlama işlemi gerçekleştirilmektedir.
- Yok : Bu seçildiği zaman toplum değiştirilmeden geri döndürülmektedir.

Mutasyon çeşitleri: Kullanıcının seçimine göre aşağıdaki mutasyonlar çalışabilmektedir. Mutasyon içerisinde rasgele bir değer üretilip mutasyon olasılığıyla karşılaştırılmaktadır. Eğer mutasyon olasılığından düşük bir değer üretilmemişse mutasyon gerçekleştirilmez.

- Normal rasgele: Bu mutasyonda rasgele seçilen iki genin yeri değiştirilmektedir.
- Sadece Geliştiren rasgele: Bu mutasyonda rasgele seçilen iki genin yeri değiştiriliyor, eğer sonuç önceki bireyden daha iyiyse bu değişim kalıcı bir hale getiriliyor.
- Sadece Geliştiren sistematik: Bu mutasyonda sırayla tüm genler sistematik olarak değiştirilip en iyi sonucu üreten mutasyon kalıcı bir şekilde bırakılıyor.

- Yok: Bu seçildiği zaman herhangi bir mutasyon yapılmıyor.

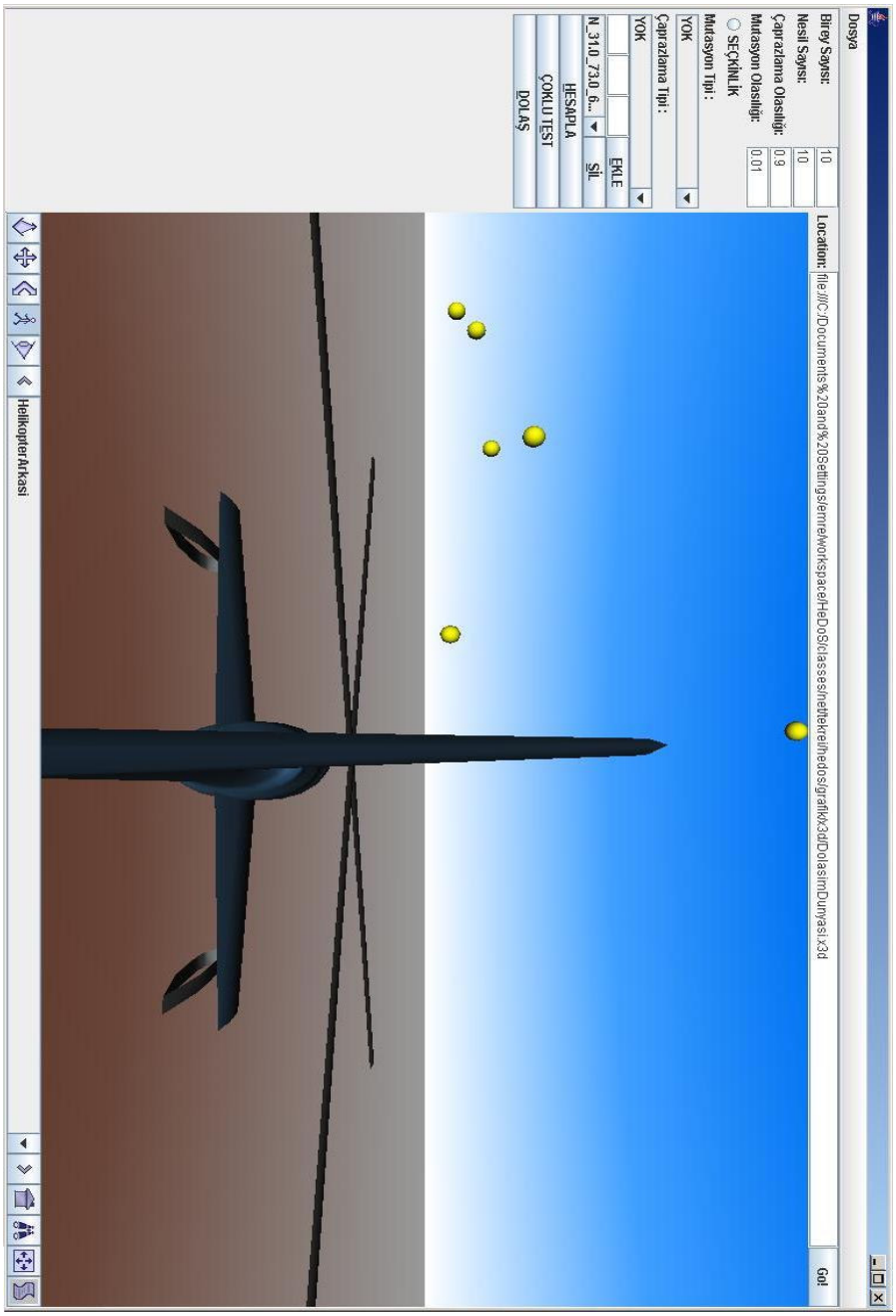
Seçkinlik: Seçkin seçiminin yapılıp yapılmayacağı kullanıcının arayüzden seçimine bağlıdır. Seçkinlik yapılması istendiği zaman her nesildeki en iyi birey bir sonraki nesle aktarılmaktadır. Her yeni nesil başlangıcında bu seçkin birey en kötü birey ile (sıralamada sonuncu sırada yer alan) değiştirilmektedir. Seçkin birey seçimi ile amaçlanan algoritmanın başarısını arttırmak ve en uygun bireyin yok olmasını engellemektir.

7 YAZILIMIN İŞLETİLMESİ VE DEĞERLENDİRİLMESİ

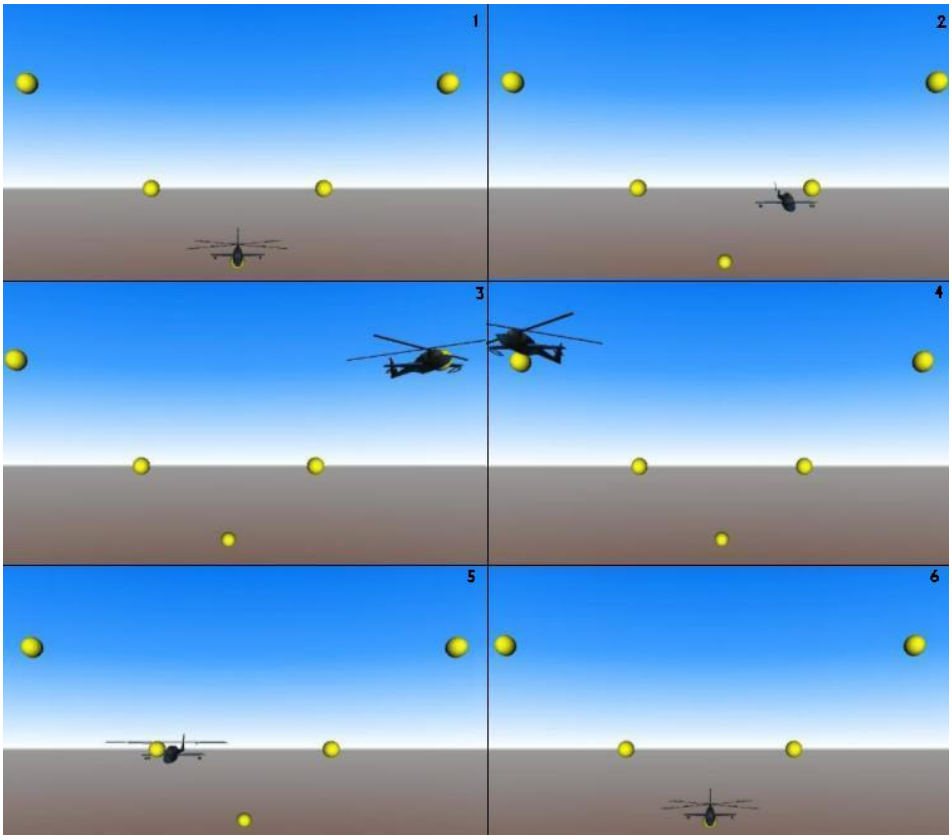
Yazılım Java Web Start şeklinde gerçekleştirilmiştir. Kullanıcıların sadece Java 1.5 veya daha üzeri bir JDK kullanarak başka bir şey kurmalarına gerek kalmadan her platformda kendiliğinden yüklenen kütüphaneler şeklinde işleri kolaylaştırılmıştır. Kullanıcılar kendilerine sunulan web adresinde link olan JNLP dosyasını çalıştırarak (http://yzgrafik.ege.edu.tr/~tekrei/hedos/) programı başlatabilirler. Program yaklaşık olarak 12 MB'lık kütüphaneleri bilgisayarlarına otomatik olarak indirecektir. İndirme bittikten sonra dosya kullanımı nedeniyle gereksinim duyulan uygulamanın güvenli olduğunu doğrulayan pencereyi onaylamaları gerekmektedir. Bu onaylamadan sonra aşağıda resmi görülen ana pencere açılacaktır. Bazı başlangıç değerleri kullanıcılara kolaylık için varsayılan olarak ayarlanmıştır.

Kullanıcı soldaki panelden istediği değişiklikleri yaparak genetik algoritmanın işletilmesini değiştirip farklı tercihler sonucu farklı deneyimler elde etmektedir. Ayrıca sahnenin altındaki düğmeleri kullanarak sahne üzerinde farklı noktalara ilerleme, bakış açısını değiştirme işlemlerini gerçekleştirebilmektedir. İlgili seçimleri yaptıktan sonra kullanıcı Hesapla düğmesine basarak genetik algoritmanın girdiği değerlere göre hesaplanmasını ateşlemektedir. Hesaplama işlemi bittikten sonra Dolaş düğmesini etkinleştirerek helikopterin bulunan en iyi yolu sahne üzerinde dolaşarak göstermesini sağlayabilir. Kullanıcı deneyimlerini düğüm ekleyerek, çıkararak, parametreleri ve kriterleri istediği gibi değiştirerek arttırabilir. Kullanıcının bakış açısı ilk açılışta helikopterin bakış açısıdır (Bkz. Şekil 7.1). Kullanıcı aşağıdaki menüden Helikopterin açılıştaki noktasını seçerek uzak bir noktadan helikopter hareketini izleyebilir (Bkz. Şekil 7.2).

Düğüm (Hedef) eklenmesi soldaki panelden x,y ve z koordinat değerlerini ilgili metin kutularına girerek ve Ekle düğmesine basarak gerçekleştirilmektedir. Düğüm silmek için ilgili combobox'tan silinecek düğüm seçilerek Sil düğmesine basılır. Bu şekilde sahne üzerindeki düğümler (hedefler) eklenebilir, silinebilir. Düğüm değişikliklerinden sonra hesaplama işleminin tekrar yapılması gerekmektedir.



Şekil 7.1: Programın ilk çalıştırılmasındaki görüntüsü



Şekil 7.2: Seçimlere göre hesaplanmış yolu dolaşan helikopter

7.1 Yazılımın Değerlendirilmesi

Genetik algoritmanın başarısını ölçmek ve en uygun genetik algoritma parametreleri ve kriterleri için yazılımı farklı seçimler için ürettiği en kısa yol ve üretim süresi bazında denemek gerekti. **Bu denemeler istatistiksel doğruluğu bir nebze de olsa elde edebilmek için yüz kere aynı değerlerle çalıştırmak ve bu değerlerin aritmetik ortalamasını almak şeklinde gerçekleştirildi.** Bu denemeler tarafımızdan dört başlığa ayrıldı.

Programın hesaplama süresi ve grafiksel imge oluşturma süresi farklı makinelerde doğal olarak farklı sonuçlar üretecektir. Bu bölümdeki denemeler için aşağıdaki özelliklere sahip bir makine kullanılmıştır:

- AMD Athlon 3200+ 64 Bit İşlemci (2.01 GHz)
- 2 GB RAM
- Nvidia GeForce 6600 Ekran Kartı
- Windows XP İşletim Sistemi

Elbette bütün kriterler birbirine bağlıdır. Bu etkilerin ölçülmesi tamamen algoritmanın saf bir şekilde çalışmasını denemekten başka bir amaç gütmemektedir. Kriterlerin farklı kombinasyonları aşağıdaki deneylerden çok farklı sonuçlar doğurabilir.

7.1.1 Birey ve Nesil Sayısının Etkisi

Bu bölümdeki amaç birey sayısının ve nesil sayısının değişiminin genetik algoritma üzerinde yarattığı farklılığı bulmak ve en uygun birey ve nesil sayılarına ulaşabilmektir.

Birey sayısının etkisini hesaplamak için aşağıdaki kriterler sabit tutularak birey sayısı değiştirildi:

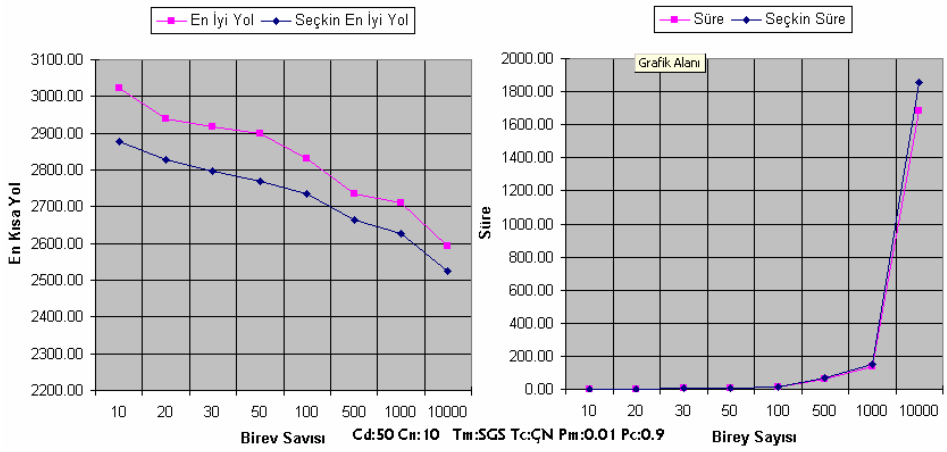
- C_d (Düğüm sayısı): 50
- C_n (Nesil sayısı): 10
- T_m (Mutasyon tipi): SGS (Sadece geliştiren sistematik)
- T_c (Çaprazlama tipi): ÇN (Çift noktalı)
- P_m (Mutasyon olasılığı): 0.01
- P_c (Çaprazlama olasılığı): 0.9

Çizelge 7.1, Şekil 7.3 yardımıyla birey sayısının artışının en kısa yolu iyileştirdiği ancak belli bir sayıdan sonra süredeki artışın göz önüne alınması gerektiği ortaya çıkmaktadır. Bu nedenle birey sayısının düğüm ve nesil sayısını da bağlı olmak şartıyla uygun bir değer seçilmesi genetik algoritmanın verimini arttıracaktır. Yukarıda belirtilen kriterler için uygun değer 1000'dir. İnceleme sonucu seçkinliğin etkin olması

durumunda en kısa yolun daha iyi olduğu gözlenmektedir.

Çizelge 7.1: Birey sayısının etkisi

Birey Sayısı	En Kısa Yol	Süre (ms)	En Kısa Yol (Seçkinlik)	Süre (ms) (Seçkinlik)
10	3021.24	1.72	2877.47	1.40
20	2940.67	2.50	2828.52	2.82
30	2918.29	3.75	2796.35	4.85
50	2899.35	5.63	2767.85	6.40
100	2831.27	11.72	2735.55	12.97
500	2734.63	62.36	2663.71	68.75
1000	2708.95	136.42	2627.26	148.90
10000	2592.08	1683.90	2526.04	1858.28



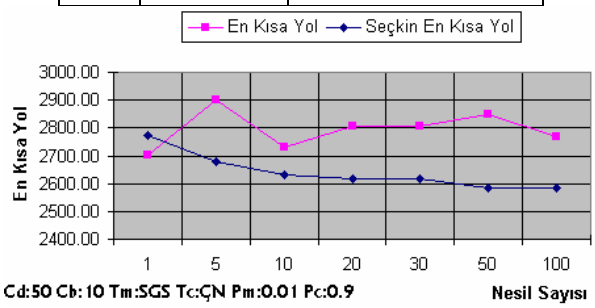
Şekil 7.3: Birey sayısının etkisi

Nesil sayısının etkisini hesaplamak için de yukarıdaki kriterlerden farklı olarak birey sayısı 10'da sabitlendi. Nesil sonundaki en kısa yollar tabloda ilgili yerlere yazıldı.

Çizelge 7.2 ve Şekil 7.4 incelendiği zaman nesil ilerledikçe en kısa yol değişmektedir. Verimli bir sonuç alabilmek için ise seçkin seçiminin ayarlanması gerekmektedir. Yoksa her nesilde elde edilen en kısa yol kaybolmaktadır.

Çizelge 7.2: Nesil sayısının etkisi

Nesil	En Kısa Yol	En Kısa Yol(Seçkinlik)
1	2701.78	2772.03
5	2899.38	2677.52
10	2731.22	2632.96
20	2807.69	2617.29
30	2807.69	2617.29
50	2846.98	2583.48
100	2770.11	2583.48



Şekil 7.4: Nesil sayısının etkisi

7.1.2 Düğüm Sayısının Etkisi

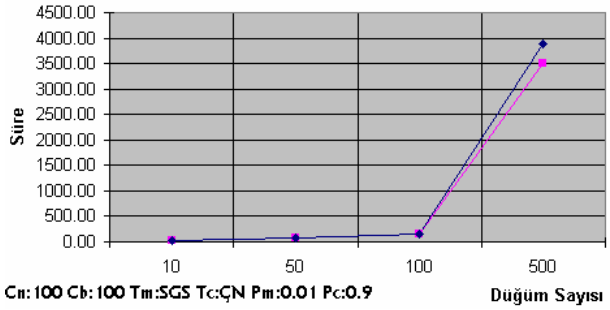
Düğüm sayısının etkisini hesaplamak için aşağıdaki kriterler kullanıldı ve düğüm sayısı değiştirildi:

- C_n : 100
- C_b (Birey Sayısı): 100
- T_m : SGS
- T_c : ÇN
- P_m : 0.01
- P_c : 0.9

Çizelge 7.3, Şekil 7.5 yardımıyla düğüm sayısının artışının süreyi 100 düğümden sonra arttırdığı gözlenmiştir. Bu gözlem sonucunda 100 düğümlük problemler için sürenin makul kalacağı öngörülmektedir.

Çizelge 7.3: Düğüm sayısının etkisi

Düğüm	Süre (ms)	Süre (ms)(Seçkinlik)
10	14.37	15.93
50	67.97	73.90
100	159.53	165.01
500	3503.44	3887.96



Şekil 7.5: Düğüm sayısının etkisi

7.1.3 Olasılıkların Etkisi

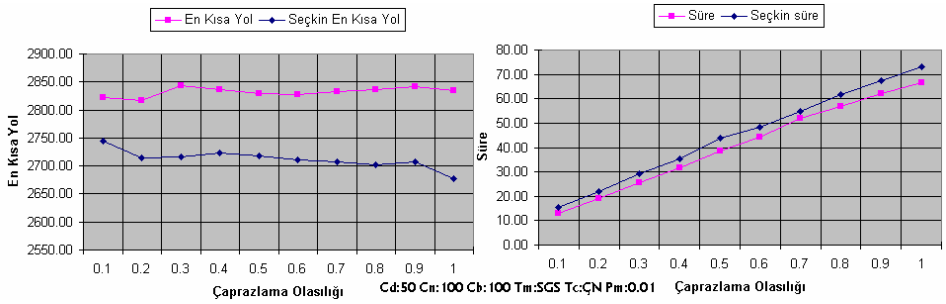
Olasılıkları hesaplamak için hesaplanan olasılık değiştirilerek diğer olasılık sabit tutulmuştur. Diğer kriterler için aşağıdaki sabit değerler kullanılmıştır:

- C_d : 50
- C_n : 100
- C_b : 100
- T_m : SGS
- T_c : ÇN

Çizelge 7.4, Şekil 7.6 yardımıyla çaprazlama olasılığının artışı ile beraber özellikle seçkinliğin seçilmesinde iyi sonuçlar döndürdüğü gözlenmiştir. Bu gözlem sonucunda süredeki değişimi de göz önüne alarak en iyi olasılık değerinin 0.5'ten büyük değerler olduğu ortaya çıkmıştır.

Çizelge 7.4: Çaprazlama olasılığının etkisi

En Kısa Yol	Süre (ms)	En Kısa Yol (Seçkinlik)	Süre (ms) (Seçkinlik)
2822.21	12.97	2744.17	15.63
2816.49	19.22	2715.15	22.02
2843.40	25.47	2715.61	29.21
2835.54	31.71	2723.32	35.46
2830.03	38.44	2717.87	43.75
2827.14	44.37	2709.99	48.13
2832.75	52.03	2708.01	55.00
2836.89	56.87	2702.70	61.56
2841.55	62.18	2706.74	67.52
2833.72	66.41	2677.10	73.12



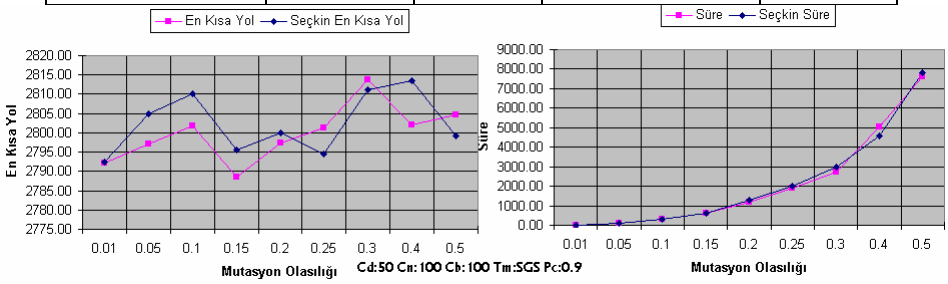
Şekil 7.6: Çaprazlama olasılığının etkisi

Çizelge 7.5, Şekil 7.7 yardımıyla çaprazlama olasılığının artışının en iyi süre/sonuç başarımının 0.1 ve 0.2 arasında olduğu gözlenmiştir.

Çizelge 7.5: Mutasyon olasılığının etkisi

Mutasyon Olasılığı	En Kısa Yol	Süre (ms)	En Kısa Yol (Seçkinlik)	Süre (ms) (Seçkinlik)
0.01	2792.24	8.28	2792.52	11.56
0.05	2796.99	77.34	2805.01	79.06
0.1	2801.78	288.12	2810.19	290.94
0.15	2788.56	642.81	2795.58	641.87
0.2	2797.25	1158.28	2800.06	1283.12
0.25	2801.18	1905.00	2794.62	2025.62
0.3	2813.74	2750.00	2811.16	2975.01

0.4	2802.07	5050.63	2813.51	4573.90
0.5	2804.55	7602.65	2799.24	7833.76



Şekil 7.7: Mutasyon olasılığının etkisi

7.1.4 Tercihlerin Etkisi

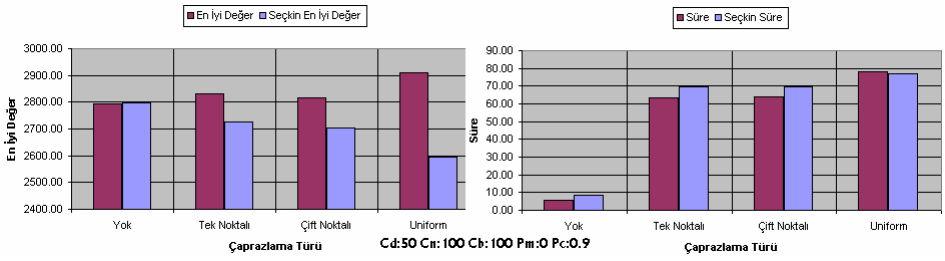
Kullanıcının mutasyon ve çaprazlama tercihlerine göre değişim gözlenmiştir. Mutasyon denenirken çaprazlama yapılmamış, çaprazlama denenirken mutasyon yapılmamıştır. Aşağıdaki değerler bu deney sırasında kullanılmıştır:

- C_d : 50
- C_n : 100
- C_b : 100

Çizelge 7.6, Şekil 7.8 yardımıyla çaprazlama tercihlerinin etkisi incelenebilir. Çaprazlama tercihlerinin denenmesinde mutasyon yapılmamıştır. Bu gözlem sonucunda en iyi sonucun seçkinliğe bağlı olduğu görülmektedir. Seçkinlik varken en iyi sonucu Uniform üretirken, seçkinliğin olmadığı durumda çaprazlama olmaması daha iyi sonuçlar üretmektedir.

Çizelge 7.6: Çaprazlama tercihinin etkisi

Çaprazlama Tipi	En Kısa Yol	Süre (ms)	En Kısa Yol (Seçkinlik)	Süre (ms) (Seçkinlik)
Yok	2794.43	5.94	2795.88	8.29
Tek Noktalı	2832.89	63.28	2725.70	69.54
Çift Noktalı	2815.12	63.75	2705.07	69.37
Uniform	2911.57	77.97	2593.40	77.04

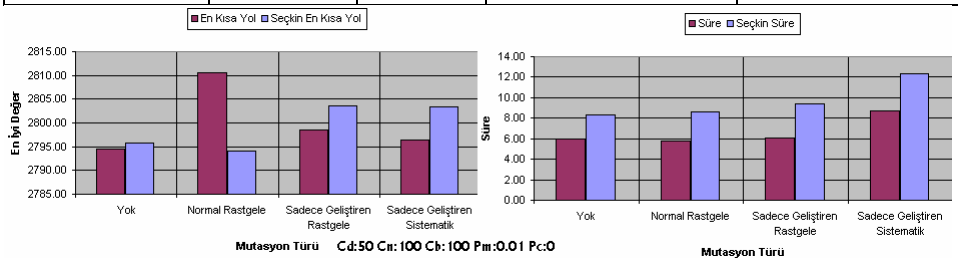


Şekil 7.8: Çaprazlama tercihinin etkisi

Çizelge 7.7, Şekil 7.9 yardımıyla mutasyon tercihlerinin etkisi incelenebilir. Mutasyon tercihlerinin denenmesinde çaprazlama yapılmamıştır. Bu gözlem sonucunda en iyi sonucun Sadece geliştiren sistematik mutasyonda olduğu görülmektedir. Seçkinliğin mutasyonda kötü sonuçlara neden olduğu görülmektedir.

Çizelge 7.7: Mutasyon tercihinin etkisi

Mutasyon Tipi	En Kısa Yol	Süre (ms)	En Kısa Yol (Seçkinlik)	Süre (ms) (Seçkinlik)
Yok	2794.43	5.94	2795.88	8.29
Normal Rastgele	2810.58	5.77	2794.14	8.59
Sadece Geliştiren Rastgele	2798.54	6.09	2803.57	9.38
Sadece Geliştiren Sistematik	2796.35	8.75	2803.43	12.34



Şekil 7.9: Mutasyon tercihinin etkisi

8 SONUÇ

Yapay Zeka ve Bilgisayar Grafikleri alanındaki bu tez çalışmasında aşağıda belirtilen iki konu ve birbirleri ile etkileşimleri üzerine araştırma yapılmıştır:

İnternet üzerinde üç boyut: İnternet üzerinde üç boyut beklediği ilgiyi görmeye yeni başlamaktadır. Web3D birliğininin ve bu birlikte üyelikleri bulunan Yumetech, Media Machines gibi bu alanın lokomotifi olan firmaların yoğun çabaları bu konuya olan ilgiyi arttırmaktadır. Bu süreç uzun zamandır devam etmesine rağmen 2006 yılında zirveye varmıştır. En son olarak Media Machines firmasının ticari bir ürünlerini (Flux) açık kaynak ve serbest olarak dağıtacaklarını açıklamaları bu sürecin gelişimi açısından çok önemlidir. Bu bilgilerin ışığında tez çalışmasında geldiği son noktada Web3D ve uygulanabilirliği bir bakıma denenmiştir. Henüz geliştirme aşamasında olan X3D standardı ve programlara bütünleştirmeye yarayan Xj3D'nin kullanımı hakkında elde edilen sonuçlar şu şekilde özetlenebilir:

- X3D artık olgunlaşmaya başlamıştır. Süreç X3D'nin standartlaştırılmasından daha çok X3D'nin yaygınlaştırılması ve kullanımının arttırılmasına odaklanmıştır. Bunu sağlamak için Web3D birliği ve lokomotif firmalar kullanım için yazılımlar geliştirmektedir.
- X3D bir programlama dilinin verebildiklerini tam olarak vermese de genişletilebilirliği ve SAI desteği sayesinde bu problemi aşmaktadır.
- Xj3D X3D dosyalarını üç boyutlu bilgi saklama ortamı olarak kullanmak isteyenler için bir çok kolaylık sağlamaktadır.
- Xj3D 1.0 sürümünü yeni çıkarmasına rağmen son 1 yılda hızlı geliştirme sürecinden payını almış, hataların kısa sürede temizlendiği bir açık kaynak yazılımı olmuştur.
- X3D ve Xj3D'nin yaygınlaşması önündeki önemli engellerinden biri modellerin İnternet üzerinde gösterilmesinde sıkıntı yaratan veri yolu büyüklüğü kısıtlarıdır. Bunu aşmak için sıkıştırma yetenekleri geliştirilmektedir.

- Varolan İnternet tarayıcıların henüz varsayılan olarak üç boyutlu grafikleri doğrudan desteklememeleri bir eksikliktir. Bunun aşılması gerekmektedir.

Gezgin satıcı probleminin üç boyutlu çözümü ve genetik algoritmalar: Gezgin satıcı problemi günlük yaşamda bir çok kullanım alanına uyarlanabilecek yapıdadır. Bir postacının mektupları dağıtırken izlediği yol, seyyar satıcıların en kısa sürede en çok sokağı dolaşmak için izledikleri yol bu tip problemlere birer örnektir. Bir çok uygulama alanına sahip olan bu problemin etkin çözümü de önemli bir araştırma konusudur. Üç boyutlu olarak dolaşma problemi olarak kurtarma robotlarının veya kurtarma helikopterlerinin üç boyutlu uzay olarak tabir edebileceğimiz alandaki hareketlerinin verimli bir şekilde sağlanmasında bu çözüm yöntemleri kullanılmaktadır.

Gerçek yaşamdaki kurtarma operasyonlarında en kısa sürede en az maliyetle helikopter kullanımını sağlayan çözüm sunacak bir yazılım altyapısı geliştirmek, bu çalışmanın somut hedeflerinden birisidir. Bu problem bir gezgin satıcı problemi olarak temsil edilmektedir. Çözümde, bilgisayar bilimlerinin özellikle eniyileme konusunda başarısını kanıtlamış olan genetik algoritmalar yaklaşımından yararlanılmıştır. Bu algoritmanın kodlanmasından sonra, HeDoS değişik seçeneklere göre (mutasyon olasılığı, çaprazlama olasılığı, birey sayısı, nesil sayısı, düğüm sayısı, mutasyon çeşidi, çaprazlama çeşidi) denenmiş, elde edilen değerler yorumlanmıştır:

- Algoritmadaki seçenekler algoritmanın sonucunu dolayısıyla en kısa yolu oldukça etkilediğinden tüm genetik algoritma uygulamalarına benzer şekilde iyi ayarlanmaları gerekmektedir.
- Genetik algoritmalar en iyi sonucu kesin olarak üretmemektedir. Ancak çok kısa sürede kabul edilebilir uygun ve güzel sonuçlar elde edilmektedir.
- Genetik algoritmaların verilen sabit süre içerisindeki başarısı gelişen donanım ve yazılım teknolojisi sayesinde artmaktadır.

Yapay zeka ve Bilgisayar Grafiklerinin Bütünleştirilmesi:

- Yapay zeka açısından bilgisayar grafikleri anlaşılırlığı arttırmakta, yöntemlerin somut hale gelmesini sağlamaktadır.

- Tüm bunlar deney sonuçlarının etkin bir şekilde incelenmesini sağlayarak çözüm kalitesini arttırmaktadır.
- Bilgisayar grafiklerinin otomatik sistemlerde ve gerçek yaşamda daha verimli kullanımına öncülük etmektedir.

Hazırlanan web tabanlı görsel araç, üç boyutlu uzayda belirtilen tüm noktaları en etkin şekilde dolaşmayı sağlayan turu bulmak isteyen kullanıcılar açısından oldukça yararlı olacak sonuçlar üretmektedir. Dolaşma benzeştirimi üç boyutlu grafiklerle de desteklenmektedir.

Bu çalışma, geliştirilen yazılımın kullanıcılar için yararlı olmasının yanında, aynı zamanda yapay zeka ve bilgisayar grafiklerinin, etkin olarak kullanılmalarının avantaj sağladığı bir problem üzerinde altyapı gerçekleştirimi olarak da değerlidir.

KAYNAKLAR DİZİNİ

Barbieri, T., Garzotto, F., Beltrame, G., Ceresoli, L., Gritti, M., Misani, D., 2001, From Dust to Stardust: a Collaborative 3D Virtual Museum of Computer Science, in Proceedings ICHIM 01, Milano, Italy

Breiteneder, C., Eidenberger, H., Fiedler, G., Raab, M., 2002, Lookmark: A 2.5D Web Information Visualization System, Proceedings of EURASIA-ICT Conference, Teheran, Iran

Brutzman, D., 2005, X3D Production Methods using XML, Available at:http://people.cs.vt.edu/~bowman/3dui.org/web3d2005/X3D_IT_Tutorial/eng_ves_brutzman3.pdf

Chittaro, L., Ranon, R., Ieronutti, L., 2003, Guiding visitors of Web3D worlds through automatically generated tours, Proceeding of the eighth international conference on 3D Web technology, 27-38

Cult3D, 2006, Cult3D, Available at: http://www.cult3d.com/Cult3D/default.asp

Dauner, J., Landauer, J., Stimpfig, E., Reuter, D., 1998, 3D product presentation online: the virtual design exhibition, Proceedings of the third symposium on Virtual reality modeling language, 57-62

Dorigo, M., Gambardella, L.M., 1997, Ant colony system: a cooperative learning approach to the travelingsalesman problem, Evolutionary Computation, IEEE Transactions on, 1(1):53-66

ECMA, Standard ECMA-363, Available at: http://www.ecma-international.org/publications/standards/Ecma-363.htm

Fogel, D.B., 1988, An Evolutionary Approach to the Traveling Salesman Problem, Biological Cybernetics, 60:139-144

Hall, M., 2004, Pathfinding in an Entity Cluttered 3D Virtual Environment, Available at: http://www.comp.leeds.ac.uk/fyproj/reports/0405/HallM.pdf

Holland, J., 1975, Adaption in Natural and Artificial Systems, University of Michigan Press

Hui, Y.C., Prakash, E.C., Chaudhari, N.S., 2004, TENCON 2004. 2004 IEEE Region 10 Conference, B(2):306-309

Kann, V., 2005, A compendium of NP optimization problems, Available at: http://www.nada.kth.se/~viggo/problemlist/compendium.html

Klima, O., 2000, Lunar Landing in Web3D, ACM SIGGRAPH Computer Graphics, 34(2):57-58

Lepouras, G., Charitos, D., Vassilakis, C., Charissi, A., Halatsi, L., 2001, Building a VR-Museum in a Museum, Third International Virtual Reality Conference, Laval, France

Lutton, J.L., Bonomi, E., 1984, The N-City Travelling Salesman Problem: Statistical Mechanics and the Metropolis Algorithm, SIAM Review, 26:551-568

MOVES, 2006, The MOVES Institute, Available at: http://www.nps.navy.mil/moves/

Nikolos, I.K., Tsourveloudis, N., Valavanis, K.P., 2001, Evolutionary algorithm based 3-D path planner for UAV navigation, in Proc. CD-ROM 9th Mediterranean Conf. Control Automation Dubrovnik, Croatia

Obitko, 1998, Introduction to genetic algorithms with Java applets, Available at: http://cs.felk.cvut.cz/~xobitko/ga/index.html

Obitko, Marek and Slavík, Pavel., 1999, Visualization of Genetic Algorithms in a Learning Environment, Spring Conference on Computer Graphics, SCCG'99, p. 101-106.

OpenGL, 2006, OpenGL Web Site, Overview Page, Available at: http://www.opengl.org/about/overview/

Parrallel, 2006, Internet Space Builder, Available at: http://www.parallelgraphics.com/products/isb/

Reese, B., Stout, B., 1999, Finding a pathfinder, AAAI 99 Spring Symposium on Artificial Intelligence and Computer Games

Savage, 2006, SAVAGE library of dynamic 3D military models, Available at: http://savage.moves.nps.navy.mil/Savage/

Tornincasa, S., 2002, The Leonardo WEBD project: an example of the Web3D technology applications for distance training and learning, XIV Congreso Internacional de Ingeniería Grafica INGEGRAF, Santander

Vikipedi, 2006, Vikipedi Özgür Ansiklopedi 3ds Max bilgilendirme sayfası, Erişim: http://tr.wikipedia.org/wiki/3D_Studio_MAX

Vikipedi, 2006a, Vikipedi Özgür Ansiklopedi 3ds Max bilgilendirme sayfası, Erişim: http://tr.wikipedia.org/wiki/Autocad

Vikipedi, 2006b, Vikipedi Özgür Ansiklopedi Yapay Zeka Tarihçesi, Erişim: http://tr.wikipedia.org/wiki/Yapay_Zeka#Tarih.C3.A7e

Web3d, 2004, Web3D Consortium Comparison of X3D versus VRML, Available at: http://www.web3d.org/x3d/x3d_vs_vrml.html

Web3d, 2006, What is Web3D?, Available at: http:/www.web3d.org/about/overview/

Wikipedia, 2006, Wikipedia The Free Encyclopedia Blender (Software) page, Available at: http://en.wikipedia.org/wiki/Blender_%28software%29

Wikipedia, 2006a, Wikipedia The Free Encyclopedia Java 3D page, Available at: http://en.wikipedia.org/wiki/Java_3D

Xj3D, 2004, Xj3D Architecture, Available at:

http://www.xj3d.org/arch/architecture.html

EKLER

Ek 1 Terimler Sözlüğü

Ek 1 Terimler Sözlüğü

Cross Platform: Birden fazla farklı platformda çalışabilen yazılım

Audiovisual: Öğretimde kullanılan yardımcı araç, kulak ve göze aynı anda hitap eden sistem; kitaptan başka ögretim araçları ile ilgili;görsel işitsel

Immersive: Sanal gerçeklik terimi olarak kullanıcının sanal dünya ile tamamen çevrildiği hissini yaratan ortam.

Sahne Çizgesi: Özellikle vektör tabanlı grafik değiştirme programları ve modern oyunlar tarafından kullanılan genel veri yapısıdır. Nesne tabanlı bir yapıdır. Çizge veya ağaç yapısında düğümlerden oluşur. Grafiksel sahnenin uzaysal ve mantıksal (şekil bilgisi, davranış bilgisi, dönüşüm bilgisi) bilgisini içerir.

Gezgin Satıcı Problemi (GSP): Bir eniyileme problemi olan bu problemde bir çizge üzerine yerleştirilmiş noktalar ve aralarındaki maliyetler göz önüne alınarak her düğüme yalnız bir kere uğramak şartıyla en uygun maliyetle çizgedeki tüm düğümlerin dolaşılması olarak tanımlanabilir. Bu problemin çözümü bir Hamilton Döngüsü olarak da görülebilir.

ÖZGEÇMİŞ

Tahir Emre KALAYCI

Adres : Elektrik Mh. Filiz Sk. Kanatlı Apt. D:3 Antakya/HATAY
Tel : (232) 3478978 , (326) 2234491
E-Mail: tahir.kalayci@yahoo.com

Kişisel Bilgiler

Milliyeti : T.C.
Doğum Yeri / Tarihi : Antakya / 07.07.1981

Eğitim

2003 – Ege Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği A. D.

1999 – 2003 Ege Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Mesleki İlgi Alanları

Yapay Zeka
Güvenlik ve Kriptoloji
Nesne Tabanlı Tasarım
3D Grafik Uygulamaları