

**EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
(DOKTORA TEZİ)**

**T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**

**BİLGİSAYAR GRAFİKLERİNDEKİ
GEOMETRİK PROBLEMLERİN
ÇÖZÜMÜNDE KULLANILAN
YENİ YÖNTEMLER ve GERÇEKLEŞTİRİMLERİ**

Aybars UĞUR

Bilgisayar Mühendisliği Anabilim Dalı

Bilim Dalı Kodu : 619.01.00

Sunuş Tarihi : 17.05.2001

112704

Tez Danışmanı : Doç. Dr. Ahmet KAŞLI

112704

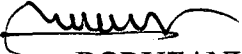
Bornova - İZMİR

Sayın Aybars UĞUR tarafından DOKTORA TEZİ olarak sunulan "**Bilgisayar Grafiklerindeki Geometrik Problemlerin Çözümünde Kullanılan Yeni Yöntemler ve Gerçekleştirimleri**" adlı bu çalışma, "Lisansüstü Eğitim ve Öğretim Yönetmeliği" nin 24 üncü madde (c) ve (d) bentleri ve Enstitü yönergesinin ilgili hükümleri dikkate alınarak tarafımızdan değerlendirilmiş olup, yapılan sözlü savunma sınavında aday oy **barışgi**... ile başarılı bulunmuştur. Bu nedenle Aybars UĞUR'un sunduğu metnin doktora tezi olarak kabulüne oy **barışgi**... ile karar verilmiştir.

17 Mayıs 2001

Jüri Başkanı ; Doç. Dr. Ahmet KAŞLI
 Üye ; Doç. Dr. Yalçın ÇEBİ
 Üye ; Yrd. Doç. Dr. Aylin KANTARCI.....

Bu tezin kabulü, Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 21/5 2001... gün ve ... 21-35... sayılı kararı ile onaylanmıştır.


 Dr. Süleyman BORUZANLI
 Enstitü Sekreteri


 Prof. Dr. Alaettin TAYSUN
 Enstitü Müdürü

ÖZET**BİLGİSAYAR GRAFİKLERİNDEKİ GEOMETRİK
PROBLEMLERİN ÇÖZÜMÜNDE KULLANILAN YENİ
YÖNTEMLER VE GERÇEKLEŞTİRİMLERİ**

UĞUR, Aybars

Doktora Tezi, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Doç. Dr. Ahmet KAŞLI

Mayıs 2001, 193 sayfa

Bu tez projesinde ilk olarak, bilişimsel geometri alanındaki yeni yöntemlerin bilgisayar grafikleri alanındaki geometrik problemlerin çözülmesine katkıları incelenmiştir.

Sonraki aşamada Java programlama dili ile, birçok geometrik hesaplamalar da içeren genel amaçlı ve etkileşimli üç boyutlu grafik yazılımları geliştirilerek İnternet üzerinde kullanıma sunulmuştur. İnternet üzerinden erişen kullanıcılar üç boyutlu nesnelerle etkileşim kurabilmektedirler. Böylece Java programlama dilinin bilgisayar grafiklerinde etkin kullanımı gerçekleştirilmiştir. Üç boyutlu grafik programlamada ve görüntülemelerde ölçümlerden elde edilen değerlerin yardımı ile iki yöntemin karşılaştırılması yapılmıştır.

Java 3D API kullanılarak geliştirilen yazılım, üç boyutlu dönüşüm işlemleri, değişik izdüşüm yöntemleri (paralel izdüşüm, perspektif izdüşüm), arka ve görünmeyen yüzeylerin elenmesi yöntemleri dışında aydınlatma gibi birçok ek özelliği de içermektedir.

Anahtar Sözcükler : Bilgisayar grafikleri, bilişimsel geometri, Java, Java 2D API, Java 3D API, yazılım ölçütleri, üç boyutlu dönüşüm, izdüşüm, görünmeyen yüzeylerin elenmesi, aydınlatma.

ABSTRACT

NEW METHODS FOR SOLVING GEOMETRY PROBLEMS IN COMPUTER GRAPHICS AND THEIR IMPLEMENTATIONS

UĞUR, Aybars

PhD in Computer Eng.

Supervisor: Assoc. Prof. Dr. Ahmet KAŞLI

May 2001, 193 pages

In this thesis, effects of new methods in computational geometry on solving geometric problems in computer graphics are examined firstly.

At the next stage, general purpose and interactive three dimensional graphics programs which include various geometric computations are developed by using Java and stored on the internet. Programs are designed to be executed in Web browsers and users can interact with 3D objects. In this way, efficient usage of Java programming language in computer graphics is provided. Two methods in 3D graphics programming and viewing are compared by using values obtained from measurements.

Software developed by using Java 3D API supports three dimensional transformation operations, different projection methods (parallel projection, perspective projection), back-face and hidden-surface removal methods and others like illumination.

Keywords : Computer graphics, computational geometry, Java, Java 2D API, Java 3D API, software metrics, three dimensional transformation, projection, hidden-surface removal, illumination.

TEŞEKKÜR

Tez çalışmasını yöneten ve başta kaynak ve bilgi olmak üzere her konuda destek veren Ege Üniversitesi Bilgisayar Mühendisliği Bölüm Başkan Yardımcısı ve danışmanım Doç. Dr. Ahmet KAŞLI'ya teşekkür ederim.

Tez çalışması süresince gösterdikleri yakın ilgi ve sağladıkları olanaklar için, Ege Üniversitesi Rektör Yardımcısı Prof. Dr. Fikret İKİZ'e, Ege Üniversitesi Bilgisayar Mühendisliği Bölüm Başkanı Prof. Dr. Şaban EREN'e, Prof. Dr. Serdar KORUKOĞLU'ya, Ege Üniversitesi Mühendislik Fakültesi Dekan Yardımcısı Doç. Dr. Mustafa TÜRKSEVER'e, Doç. Dr. Yalçın ÇEBİ'ye, Doç. Dr. Ata ÖNAL'a, Yrd. Doç. Dr. Mustafa İNCEOĞLU'ya, Yrd. Doç. Dr. Aylin KANTARCI'ya ve Bölüm Başkan Yardımcısı Yrd. Doç. Dr. Aziz Can YÜCETÜRK'e teşekkür ederim.

Bu tezin ortaya çıkmasında doğrudan veya dolaylı katkıda bulunan, konu ile ilgili kaynakların edinilmesinde yardımcı olan, değerli fikir ve önerilerini esirgemeyen herkese teşekkür ederim.

Tez çalışması süresince verdikleri maddi ve manevi destek ve gösterdikleri anlayıştan dolayı, aileme teşekkür ederim.

ŞEKİLLER DİZİNİ

<u>Sekil</u>	<u>Sayfa</u>
4.1 Noktaların birbirini görmesi	45
4.2 Özleri koyu renkli olarak belirtilmiş değişik çokgenler	46
4.3 Bir noktalar kümesinden yıldız-şekilli bir çokgen oluşturulması ..	47
4.4 Sonlu noktalar kümesi ve dışbükey kabuğu	48
4.5 Eklemeli kabuk yönteminde etkin kabuğa s noktasının eklenmesi	49
4.6 Bir düzlemdeki noktalar kümesinin dışbükey kabuğunun hediye sarma yöntemi ile bulunması	52
4.7 Graham taramasında noktaların p_0 'a göre açısız koordinatlarına bakılarak etiketlendirilmesi.....	53
4.8 Graham Taramasının bir örnek üzerindeki adımları.	54
4.9 Farklı dışbükey kabukların birleştirilmesi	57
4.10 Bölge sorgulaması	58
4.11 Izgara (Grid) Yöntemi	59
4.12 Quadtree bölümlenmesi ve Quadtree gösterimi.....	60
4.13 Bir çokgenin kesim çizgileri ile ayrıştırılması ve iki boyutlu ağacı	60
4.14 İkili uzay bölümlenme ve temsil eden BSP ağacı.....	61
5.1 Işık yansıması.....	64
5.2 Nokta ışık kaynağından çıkan ışın yolları	65
5.3 Dağıtık ışık kaynağı ile aydınlatılmış bir nesne.....	66
5.4 Bir yüzeyden yayılan yansımalar.	68

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
5.5 Gelen ışığın yönüne dik ve eğik yüzeyler.	70
5.6 Yayılan yansıma	71
5.7 Yönlü yansıma	72
5.8 Gouraud gölgelendirmesinde normal vektörü hesaplaması	75
5.9 Gouraud gölgelendirmesinde aradeğerleme	75
5.10 Çokgen kenarı boyunca yüzey normali aradeğerlemesi.....	77
5.11 Işın dağıtma yöntemi.....	79
5.12 Işın izleme	81
5.13 Kesişim noktasından çıkan yansıma, kırılma ve gölge ışınları	82
5.14 Kesişim noktalarında diğer ışıklardan özyineli oluşturulan ışınlar ve kullanılan ışın ağacı	83
6.1 Arka yüzeylerin belirlenmesi.....	92
6.2 Yüzey noktalarının derinlik değerlerinin hesaplanması.....	94
7.1 Kenar uzunluğu 1 birim olan bir küpün köşe listesi, kenar listesi ve çokgen yüzey listesi yardımı ile kenar tabanlı sınır temsili.	106
7.2 Çokgen yüzeyli nesne bilgilerini içeren listelerin düzeyleri ve bağlantıları.....	107
7.3 Çokgensel yüzey yaklaşımı ile, dilim sayısı değiştirilerek oluşturulmuş silindirler	108
7.4 Telkafes gösterim ve katı gösterim	109

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
7.5 Üç boyutlu nesnelerin görüntülenmesi sırasında gerçekleştirilen işlemler (View Pipeline).....	110
7.6 Java 3D Görüntü Ağacı	117
7.7 Bir küp içeren “Görüntü Ağacı”	119
7.8 Etkileşimli bir küp oluşturan programın applet penceresi.....	119
8.1 Java 3D koordinat sistemi (sağ el)	123
8.2 Java 3D sınıf sıradüzeni.....	124
8.3 "GeometryArray" alt sınıfları.....	128
8.4 "GeometryStripArray" alt sınıfları	129
8.5 İndeks Dizisi tutularak üç boyutlu nesne tanımlama	130
9.1 Kup3B çalıştırıldığında ortaya çıkan appletviewer penceresi	137
9.2 Netscape gibi bir web tarayıcısı kullanılarak appletin yüklenmesi	138
9.3 Küpün perspektif izdüşüm alınarak çizdirilmesi sonucu oluşan görüntü.....	141
9.4 Küpün telkafes gösterimi ile çizdirilmesi	142
9.5 Model3B yazılımı Java 2D sürümü ile oluşturulmuş bir sahne... ..	144
9.6 Model3B ile nesnelere geçirgenlik özelliği verilmesi	145
9.7 Sahnedeki üç boyutlu nesnelerin bilgilerini tutan veri yapısı.....	147
9.8 Java 3D kullanılarak hazırlanan Kup3B programının pencere görünümü.....	148

ŞEKİLLER DİZİNİ (devam)

<u>Sekil</u>	<u>Sayfa</u>
9.9 Java 3D kullanılarak hazırlanan Model3B adlı yazılımın pencere görünümü.....	153
9.10 Model3B yazılımının üç boyutlu grafiksel içerik eklemeye ilişkin bazı özelliklerini gösteren pencere görünümü.....	154
9.11 Kup3B yazılımı kaynak kodundaki bölümlerin satır sayısı yüzdeleri (Java 2D Sürümü)	160
9.12 Kup3B yazılımı kaynak kodundaki bölümlerin satır sayısı yüzdeleri (Java 3D sürümü).....	160
9.13 Model3B yazılımı (Java 2D sürümü) kaynak kodundaki bölümlerin satır sayısı yüzdeleri.....	164
9.14 Model3B yazılımı (Java 3D sürümü) kaynak kodundaki bölümlerin satır sayısı yüzdeleri.....	164
9.15 Java 2D yardımı ile geliştirilen Kup3B (3 işlevli) ve Model3B (5 işlevli) yazılımlarının bölümlerine göre kod satır sayıları.	167
9.16 Java 3D yardımı ile geliştirilen Kup3B (3 işlevli) ve Model3B (10 işlevli) yazılımlarının bölümlerine göre kod satır sayıları.	168
9.17 Java 2D yardımı ile geliştirilen Kup3B (3 işlevli) ve Model3B (5 işlevli) yazılımları arasında bölümlerine göre kod satır sayısı yüzdeleri.	169
9.18 Java 3D yardımı ile geliştirilen Kup3B (3 işlevli) ve Model3B (10 işlevli) yazılımları arasında bölümlerine göre kod satır sayısı yüzdeleri.	169
9.19 Yığılmış sütun grafiği.	170

ŞEKİLLER DİZİNİ (devam)**Sekil****Sayfa**

- 9.20 Tüm yazılımların tüm bölümlerinin üç boyutlu sütun grafiğinde karşılaştırılması. 171
- 9.21 İşlev sayısının artışına göre kaynak kod satır sayısı değişimi 172



ÇİZELGELER DİZİNİ

<u>Cizelge</u>	<u>Sayfa</u>
1.1 Uygulama kısmında kullanılan donanım ve yazılım.....	2
1.2 Yazılımların çalıştırılabileceği en düşük sistem gereksinimleri.....	3
4.1 Dışbükey kabuk bulma algoritmalarının zaman karmaşıklıkları...	61
7.1 Grafik API'lerinde 4 nesil (Sun Microsystems,1999).....	113
9.1 Düğme panelindeki düğmeler ve görevleri	140
9.2 Kup3B yazılımında kullanılan tuşlar ve işlevleri	143
9.3 Kup3B yazılımı Java 3D sürümünde küp üzerinde uygulanabilen fare hareketleri ve işlevleri	148
9.4 Kup3B yazılımlarının geliştirilmesinde Java 2D API ve Java 3D API karşılaştırılması.....	157
9.5 Kup3B yazılımının (Java 2D ve Java 3D sürümleri) bölümlerine göre kaynak kodlarındaki satır sayıları..	159
9.6 Model3B yazılımının bölümlerine göre kaynak kodlarındaki satır sayıları	163

SİMGELER ve KISALTMALAR DİZİNİ

<u>Kısaltmalar</u>	<u>Açıklama</u>
2B	İki Boyutlu
3B	Üç Boyutlu
API	Uygulama Geliştirme Arayüzü
CAD	Bilgisayar Destekli Tasarım
CAM	Bilgisayar Destekli Üretim
CH	Dışbükey Kabuk
CPU	Merkezi İşlem Birimi
GIS	Coğrafi Bilgi Sistemleri
NC	Sayısal Kumanda
VFX	Görsel Efektler
VRML	Sanal Gerçeklik Modelleme Dili

İÇİNDEKİLER

Sayfa

ÖZET	V
ABSTRACT	VII
TEŞEKKÜR	IX
ŞEKİLLER DİZİNİ	XI
ÇİZELGELER DİZİNİ	XVI
SİMGELER VE KISALTMALAR DİZİNİ	XVII
 1. GİRİŞ	 1
2. BİLİŞİMSEL GEOMETRİ	6
2.1 Bilişimsel Geometrinin Anlamı ve Gelişim Süreci	6
2.2 Bilişimsel Geometrinin Önemi ve Yararları	8
2.3 Bilişimsel Geometrinin Kullanıldığı Alanlar	9
2.3.1 Bilgisayar Grafikleri	10
2.3.2 Geometrik Modelleme ve Spline'lar	11
2.3.3 Katı Modelleme	12
2.3.4 Görüntü İşleme ve Örüntü Tanıma	13
2.3.5 Şekillerin Yeniden Oluşturulması	14
2.3.6 Bilgisayarın Görmesi	17
2.3.7 Çizgelerin Çizimi	17
2.3.8 Coğrafi Bilgi Sistemleri	18
2.3.9 Hareket Planlama	20
2.3.10 Robotbilim	21
2.3.11 Tasarım, Üretim ve Ürün Tasarlama	21

İÇİNDEKİLER (devam)

Sayfa

2.4 Bilişimsel Geometrinin Gelecekte Göstereceği Gelişim	23
3. BİLGİSAYAR GRAFİKLERİ	24
3.1 Bilgisayar Grafiklerinin Günümüzdeki Durumu	24
3.2 Bilgisayar Grafiklerinin Kullanıldığı Alanlar	25
3.2.1 Bilim ve Bilimsel Görselleştirme	25
3.2.2 CAD/CAM	27
3.2.3 Eğitim ve Öğretim.....	29
3.2.4 Eğlence	30
3.2.5 Reklamcılık.....	31
3.2.6 Sanat.....	31
3.2.7 Sanal Gerçeklik ve Güçlendirilmiş Gerçeklik.....	32
3.2.8 Web	32
3.3 Bilgisayar Grafiklerinin Geleceği	34
4. GEOMETRİK PROBLEMLERİN ÇÖZÜMÜNDE KULLANILAN ALGORİTMİK YÖNTEMLER.....	36
4.1 Genel Bilgiler.....	36
4.2 Problem, Veri, Algoritma, Veri Yapıları.....	36
4.3 Algoritmaların Karşılaştırılması ve Algoritma Analizi.....	38
4.3.1 İşletim Zamanı (Running Time)	41
4.3.2 Asimptotik Analiz.....	42

İÇİNDEKİLER (devam)

Sayfa

4.4 Birer Birer Ekleme (Incremental Insertion) Yöntemi.....	43
4.4.1 Yıldız-Şekli Çokgenlerin Bulunması	44
4.4.2 Dışbükey Kabukların (Convex Hulls) Bulunmasında Eklemeli Kabuk Yöntemi	47
4.5 Birer Birer Seçme (Incremental Selection) Yöntemleri	49
4.5.1 Seçimli Sıralama	50
4.5.2 Dışbükey Kabukların Bulunmasında Hediye Sarma Yöntemi.....	51
4.5.3 Dışbükey Kabukların Bulunmasında Graham Taraması	52
4.6 Düzlem Tarama Algoritmaları	54
4.6.1 Dışbükey Kabukların Bulunmasında Eklemeli Kabuk Yöntemi.....	55
4.7 Böl ve Çöz Algoritmaları.....	56
4.7.1 Dışbükey Kabukların Bulunmasında Birleştirmeli Kabuk (Merge Hull) Yöntemi	57
4.8 Uzaysal Bölümleme Algoritmaları.....	58
4.9 Dışbükey Kabuk Bulma Algoritmaların Karşılaştırılması.....	61
5. BİLGİSAYAR GRAFİKLERİNDE GERÇEKÇİLİK.....	62
5.1 Aydınlatma Modelleri ve Yüzey Kaplama Yöntemleri.....	62
5.2 Işık Kaynakları	64
5.3 Temel Aydınlatma Modelleri.....	66
5.3.1 Genel Aydınlatma (Ambient Light).....	67
5.3.2 Yayılan Yansıma (Diffuse Reflection).....	68
5.3.3 Yönlü Yansıma (Specular Reflection).....	72
5.4 Basit Aydınlatma : Çokgen kaplama yöntemleri.....	73
5.4.1 Sabit Yoğunluklu Kaplama.....	73

İÇİNDEKİLER (devam)

Sayfa

5.4.2 Gouraud Gölgelemesi	74
5.4.3 Phong Gölgelemesi	76
5.5 Genel Aydınlatma Algoritmaları.....	78
5.5.1 Işın Dağıtma (Ray-Casting) Yöntemi	79
5.5.2 Özyineli Işın İzleme (Recursive Ray Tracing).....	80
5.5.3 Işıma (Radiosity).....	85
6. BİLGİSAYAR GRAFİKLERİNDE GEOMETRİK PROBLEMLER	87
6.1 Bilgisayar Grafiklerinde Geometrik İşlemler	89
6.1.1 Modelleme, Bilgi Toplama ve Basitleştirme.....	89
6.1.2 Gösterim	90
6.1.3 Simülasyon	90
6.2 Görünen Yüzeyleri Belirleme Yöntemleri.....	91
6.2.1 Arka Yüzeylerin Belirlenmesi	92
6.2.2 Arkada Kalan Nesne Parçalarını Belirleme Yöntemleri	92
6.3 Bilgisayar Grafiklerinde Çözüm Bekleyen Problemler.....	94
7. GRAFİK YAZILIMLARI GELİŞTİRMEDE İKİ VE ÜÇ BOYUTLU UYGULAMA GELİŞTİRME ARAYÜZLERİ	96
7.1 Giriş.....	96
7.2 Grafik Yazılımları	96
7.2.1 Grafik Uygulama Yazılımları	97
7.2.2 Grafik Uygulama Yazılımlarında Kullanılan Programlama Dilleri.....	97
7.2.3 Uygulama Geliştirme Arayüzleri.....	98
7.3 İki Boyutlu Uygulama Geliştirme Arayüzleri.....	99
7.3.1 Üç Boyutlu Nesnelerin Bilgisayarda Temsili.....	103
7.3.2 Çokgen Yüzeyli Nesnelerin Bilgisayarda Tanımlanması ..	105

İÇİNDEKİLER (devam)

Sayfa

7.3.3 Üç Boyutlu Nesnelerin Ekranda Görüntülenmesi.....	109
7.4 Üç Boyutlu Uygulama Geliştirme Arayüzleri.....	112
7.4.1 Java 3D Görüntü Ağacı (Scene Graph).....	116
7.4.2 Java 3D ile 3B Sahnelerin Oluşturulması.....	118
7.5 Üç Boyutlu Grafik Yazılımları Geliştirmede 2B ve 3B API'lerin Karşılaştırılması.....	121
8. JAVA 3D VE GEOMETRİ.....	122
8.1 Java 3D'nin Geometri Özellikleri	122
8.2 Java 3D Geometrisi.....	122
8.2.1 Sanal Nesne Tanımlama	125
8.2.2 Geometrik Hizmet Sınıfları	125
8.2.3 Matematiksel Sınıflar	126
8.2.4 Geometri Sınıfları.....	127
8.3 Sonuç	130
9. TEZ KAPSAMINDA GELİŞTİRİLEN YAZILIMLAR.....	132
9.1 Java 2D Kullanılarak Geliştirilen Yazılımlar.....	133
9.1.1 Kup3B (Java 2D sürümü).....	133
9.1.2 Model3B (Java 2D Sürümü).....	144
9.2 Java 3D Kullanılarak Geliştirilen Yazılımlar.....	147
9.2.1 Kup3B (Java 3D sürümü).....	147
9.2.2 Model3B (Java 3D sürümü).....	150
9.3 Model3B Yazılımının Kullanım Alanları	155
9.4 Yazılım Geliştirme Süreci Sonuçları	156
9.4.1 Kup3B Yazılımının Gerçekleştirmede Java 2D ve Java 3D Karşılaştırması.....	157

İÇİNDEKİLER (devam)

Sayfa

9.4.2 Model3B Yazılımının Geliştirilmesinde Java 2D ve Java 3D Karşılaştırması	162
9.4.3 Genel Karşılaştırma.....	166
9.4.4 Yazılım Ölçümü.....	173
10. SONUÇLAR.....	174
10.1 Bilgisayar Grafikleri, Bilişimsel Geometri ve Aralarındaki İlişki	175
10.1.1 Geometrik problemlerin çözümünde kullanılan algoritmik yöntemler	175
10.1.2 Bilgisayar grafiklerindeki geometrik problemler	176
10.2 Grafik Uygulama Geliştirme Arayüzleri	178
10.2.1 Yüksek Düzeyli 3B API'lerin Özellikleri ve Bilgisayar Grafikleri Alanına Etkileri.....	179
10.2.2 Yüksek Düzeyli 3B API'lerin Beklenen Genel Etkileri...	180
10.2.3 Yüksek Düzeyli 3B API'lerin Kullanımı	181
10.2.4 Yüksek Düzeyli 3B API'lerin Eksikleri.....	182

11. İLERİ ÇALIŞMALAR VE ÖNERİLER.....	185
11.1 Doktora Sonrası Yapılacak Çalışmalar.....	185
11.2 Yakın Konularda Çalışma Yapacaklara Öneriler	186
11.3 Araştırmalara Yeni Başlayacaklara Öneriler	187
YARARLANILAN KAYNAKLAR.....	188
ÖZGEÇMİŞ.....	193



1. GİRİŞ

Bilgisayar donanımları ve görselleştirme sistemlerindeki hızlı gelişmeler, geometrik hesaplamaların, bilimin ve mühendisliğin hemen her dalında, tasarım ve üretimden, astrofizığe, moleküler biyolojiye ve akışkanlar dinamiğine kadar her konuda önemli bir yer edinmesini sağlamıştır. Bilişimsel geometrinin önemi, gelecek on yılda daha da artacaktır.

Bilgisayar grafikleri de, içinde çok sayıda bilinen veya potansiyel geometrik problemler barındıran alanların önemlilerinden bir tanesidir. Dışbükey kabuklar (convex hulls) ve Voronoi şemaları (Voronoi diagrams) gibi temel geometrik problemlerin çözümünde başarıya ulaşılmış olmakla birlikte, bilinip de üzerinde çalışılan daha pek çok açık problem vardır. Bunun yanında, gelişmelerle beraber, çözüm bekleyecek pek çok yeni problemin de ortaya çıkacağı tahmin edilmektedir.

Bu tez projesinde ilk olarak, bilişimsel geometri alanındaki yeni yöntemlerin bilgisayar grafikleri alanındaki geometrik problemlerin çözülmesine katkıları araştırılmıştır.

Sonraki aşamada Java programlama dili ile, birçok geometrik hesaplamalar da içeren genel amaçlı ve etkileşimli üç boyutlu grafik yazılımları geliştirilerek İnternet üzerinde kullanıma sunulmuştur. Böylece Java programlama dilinin bilgisayar grafiklerinde etkin kullanımı gerçekleştirilirken, geometri konusunda teori ile uygulamalar arasında bağlantılar kurulmaya çalışılmıştır. Ayrıca, yararlanılan iki boyutlu ve üç boyutlu uygulama geliştirme arayüzleri karşılaştırılmıştır.

Tez projesinin uygulama kısmında Java programlama dilinin kullanılmasının nedenleri arasında, kısa sürede en yaygın ve gelişmiş yazılım geliştirme ortamlarından biri olması sayılabilir. Güncelliğini giderek artıran Java programlama dili, üç boyutlu grafik uygulama

geliştirme arayüzü (API), Java 3D ile de desteklenmektedir. Dördüncü yani en yeni nesil bir uygulama geliştirme arayüzlerinden biri olan Java 3D API, Sun Microsystems, Apple Computer, Silicon Graphics ve Intel gibi birçok büyük şirket arasında oluşan işbirliğinin bir sonucudur. İnternet üzerine üç boyutlu grafiksel içeriklerin kolaylıkla yerleştirilmesini sağlar. Java programlama dilinin iki boyutlu uygulama geliştirme arayüzü Java 2D API de üstün özelliklere sahiptir.

Tez projesinin uygulama kısmında kullanılan donanım ve yazılım ortamları Çizelge 1.1’de belirtilmiştir. İlgili yazılımların yüklenmesi durumunda programların çalıştırılabileceği en düşük donanım da Çizelge 1.2’de yer almaktadır. Bu aynı zamanda Java’nın işletilebileceği en düşük donanımdır.

Çizelge 1.1 : Uygulama kısmında kullanılan donanım ve yazılım

Kullanılan Donanım	
İşlemci ve hız	Pentium III – 500 Mhz.
Ana Bellek	64 Mb RAM
3D Ekran Kartı	NVIDIA TNT 2 Model 64
Sabit Disk	15 Gbyte

Kullanılan Yazılım	
İşletim Sistemi	Windows 98
Programlama Dili	Java 2 Platform
Grafik Arayüzü	Java 3D API (Sürüm 1.2)

Çizelge 1.2 : Yazılımların çalıştırılabileceği en düşük sistem gereksinimleri

En Düşük Sistem Gereksinimleri	
İşlemci ve Hız	Intel Pentium 90Mhz
Ana bellek	24 Mb. RAM
Sabit Disk	50 Mb. Sabit Disk Alanı
İşletim Sistemi	Windows 95 ve üstü
Programlama Dili	Java 2 Platform
Grafik Arayüzü	Java 3D API (Sürüm 1.2)

Java 2 ortamı ayrıca Solaris işletim sistemi üzerine de yüklenebildiğinden, yazılımlar bu ortamlarda da çalıştırılabilecektir.

Bölüm 2’de bilişimsel geometri alanı ana hatları ile incelenmiştir. Önemi, kullanıldığı alanlar ve gelecekteki hedefleri üzerinde durulmuştur.

Bölüm 3’te bilgisayar grafikleri alanı günümüzde geldiği noktada incelenmiş ve kullanıldığı alanlar üzerinde durularak bilgisayar grafiklerinin geleceği ve günlük yaşama etkileri üzerine tahminlemeler yapılmıştır.

Bölüm 4’te geometrik problemlerin çözümünde kullanılan algoritmik yöntemler araştırılmıştır. Bilişimsel geometri ve bilgisayar grafikleri alanlarının tanıtımının ardından, bilişimsel geometrinin problemlerin çözümünde son yıllarda kullandığı teknikler anlatılmıştır.

Bölüm 5'te bilgisayar grafiklerindeki geometrik problemlerin ana kaynağı olan bilgisayar grafiklerinde gerçekçilik konusunda yapılan çalışmalar sunulmuştur. Bilimsel alanlardaki araştırmacılardan genel bilgisayar kullanıcılarına kadar birçok yerden gelen istekler bilgisayar grafiklerinde gerçekçiliğin etkin, hızlı ve kaliteli bir şekilde sağlanmasını gerektirmektedir.

Bölüm 6'da bilgisayar grafiklerindeki geometrik problemler üzerinde durulmuştur. Önceki bölümlerde bilgisayar grafiklerinin ve bilişimsel geometrinin ayrıntılı incelemesinden sonra, bilgisayar grafiklerindeki geometrik problemlerin kaynakları, aşamaları ve çözülmeye çalışılan geometrik problemler incelenmiştir.

Bölüm 7'de, grafik yazılımları geliştirmede kullanılan programlama dilleri ve arayüzler tanıtılmıştır.

Bölüm 8'de grafik yazılımları geliştirmede kullanılan arayüzlerdeki geometri olanakları, günümüzde geldiği noktada, yüksek düzeyli bir API olan Java 3D üzerinde incelenmiştir.

Bölüm 9'da, Java programlama dili kullanılarak geliştirilmiş olan yazılımlar tanıtılmış, çeşitli karşılaştırmalar yapılmış ve ulaşılan sonuçlar verilmiştir. Java 3D API kullanılarak geliştirilen yazılım, üç boyutlu dönüşüm işlemleri, değişik izdüşüm yöntemleri (paralel izdüşüm, perspektif izdüşüm), arka ve görünmeyen yüzeylerin elenmesi yöntemleri dışında etkileşimli kamera hareketleri ve aydınlatma gibi birçok ek özelliği de içermektedir.

Bölüm 10 ve 11 ise yapılan çalışma ve geliştirilen yazılımların ışığında, bilgisayar grafikleri, bilişimsel geometri ve grafik yazılımları geliştirme ve aralarındaki ilişkiler konularında ulaşılan sonuçları, geleceğe yönelik çalışma planlarını ve benzeri konularda tez yapacak kişilere önerileri içermektedir.

Bu tez projesinin önemli ve orijinal bazı noktaları şu şekilde belirtilebilir :

- Bilgisayar grafiklerindeki geometrik problemlerin çözümünde kullanılan yöntemler kapsamlı olarak araştırılmıştır. Bilgisayar grafiklerinde ve diğer alanlarda karşılaşılan geometrik problemlerin çözümleri, çözülme biçimleri ve gelişme süreçleri incelenerek çözüm yolları hakkında genel sonuçlara ulaşılmıştır.
- Java programlama dilinin ve Java 2D API ve Java 3D API arayüzlerinin bilgisayar grafiklerinde etkin olarak kullanımı gerçekleştirilmiştir. Genel amaçlı üç boyutlu etkileşimli grafik yazılımları geliştirilerek İnternet üzerinde kullanıma sunulmuştur. İnternet üzerine üç boyutlu içerik eklemeyi de sağlayan bu yazılımlar, özel amaçlarla biraz değiştirilerek, örneğin eğitim veya e-ticaret alanlarında kullanılabilecektir.
- Kişisel bilgisayarlarda uzun yıllardır kullanılmakta olan iki boyutlu grafik uygulama geliştirme arayüzleri ile günümüzde kişisel bilgisayarlarda iş istasyonları kadar etkin kullanılmaya başlanan üç boyutlu grafik uygulama geliştirme arayüzleri, üç boyutlu grafik programlamadaki kullanımları açısından karşılaştırılmışlardır. Bu karşılaştırmada, geliştirilen yazılımlar temel alınmıştır. 3B (Üç boyutlu) API'lerdeki gelişmelerin bilgisayar grafikleri ve günlük yaşama yapacağı etkiler tahminlenmiş ve daha etkin grafik kullanımı için yapılması gerekenler belirlenmiştir.
- Bilgisayar grafiklerinde problemlerin çözümünden uygulama geliştirmeye kadar geniş bir alanda inceleme ve araştırmalar yapılarak teori ile uygulamalar arasında bağlantılar kurulmuş ve aralarındaki kopukluk giderilmeye çalışılmıştır.

2. BİLİŞİMSEL GEOMETRİ

2.1 Bilişimsel Geometrinin Anlamı ve Gelişim Süreci

“Bilişimsel Geometri” deyiimi, birçok grup tarafından kullanılmaktadır. Deyim olarak ilk defa Marvin Minsky, örüntü tanıma (pattern recognition) ile ilgili olan ve 1969’da yayınlanan “Perceptrons” adlı kitabında söz etmiştir. Günümüzde yaygın olarak kullanılan anlamı şöyledir : Bilişimsel geometri, algoritma teorisinin, başta 2,3 veya diğer sabit boyutlu uzaylardaki girdileri içeren problemlerde, etkin algoritma analizini ve tasarımını kapsayan alt alanıdır. Genelde eğriler ve eğrisel yüzeylerden çok, doğrusal ve düzgün nesneler (noktalar, doğrular, doğru parçaları, çokgenler, düzlemler ve çokyüzlüler) ile ilgilenir. Nokta, doğru ve çokgen gibi geometrik nesneler, pek çok uygulama alanında yaygın olarak kullanılan bileşenlerdir (Mount, 1997).

Bilişimsel geometri alanı, geometrik hesaplamanın algoritmik gereksinimlerini karşılamak üzere, genel araçlar (analitik ve hesaplamaya yönelik) oluşturma görevini yerine getirmek üzere doğmuştur. Amaç, bilgisayar programcılarının geometrik problemlerle uğraştıklarında, yardım için başvurabilecekleri bilgi tabanını oluşturmaktır (Chazelle ve ark., 1996).

Bu alan, 1970’li yılların sonlarından itibaren 1980’li ve 1990’lı yıllar boyunca hızlı bir gelişim göstermiştir. Ayrık Algoritma Tasarımı (Discrete Algorithm Design) alanından türetilmiş olması nedeniyle, bilişimsel geometri alanının problemleri de, ayrık matematiğin doğasına uygun olmuştur. Bilişimsel geometri alanının pek çoğunda “girdi”, sonlu sayıda noktalardan (veya diğer geometrik nesnelerden) oluşmaktadır. “Çıktı” da genelde sonlu sayıda nokta veya doğru parçası içeren yapıdadır (Mount, 1997).

Bilişimsel geometri disiplini, ciddi olarak ilk defa 1975 yılında Shamos'un doktora tezi ile dikkatleri üzerine çekmiştir. Bu yıllarda matematik programlama, teori ve algoritmalar ve bilgisayar destekli tasarım (CAD) grupları tarafından kullanılmakta olan bilişimsel geometri, on yıl sonra 1985'te bir ders kitabı, bir konferans ve periyodik bir dergi üçlüsünü tamamlayarak, her disiplinde olması gereken özellikleri tamamladı. Preparata ve Shamos'un kitabı "Computational Geometry : An Introduction" tümüyle bu konuya ayrılmış ilk kitap niteliğindeydi. Bu, bilişimsel geometri konusunu içeren ilk ACM sempozyumunun yapıldığı tarihe rastlıyordu ve "Discrete and Computational Geometry" dergisinin yayına geçmesinden az önce idi (Chen, 1996).

Bilişimsel geometri, bilgisayar grafikleri gibi uygulama alanlarında karşılaşılan problemlerin çözümünde yararlı olurken, bu alanlar da bilişimsel geometri ile çözülebilecek şekilde problemler üreterek bilişimsel geometrinin gelişmesine katkıda bulunmuşlardır. Yani gelişim, bilişimsel geometri ve kullanıldığı alanlar arasında karşılıklı yarar esasına dayanacak şekilde olmuştur. Bu alan, günümüzde de başarılı bir şekilde ilerlemektedir. Her yıl konu hakkında çok sayıda yayınlar yapılmaktadır ve bilişimsel geometri alanında doktora tezini hazırlayıp mezun olan ilk bilgisayar bilimleri öğrencileri, günümüzde yeni nesil araştırmacıları eğitmektedirler (Chen, 1996).

Bernard Chazelle ve geometrik hesaplama konusunda deneyimli bir grup bilgisayarlı bilim adamı, araştırmacı, mühendis ve matematikçi bir araya gelerek 1996 yılında bir rapor yayınlamışlardır. "*Application Challenges to Computational Geometry : CG Impact Task Force Report*" adlı bu raporun amacı, bilişimsel geometri alanında olması gereken canlandırmanın yolunu açmaktır. Gelecekte bilişimsel geometrinin, topluluklar bazında tartışılır bir konu haline geleceği umulmaktadır.

Bu raporda, bilişimsel geometrinin gelişim süreci anlatılmakta ve elde edilen fırsatlar değerlendirilmektedir. Bilişimsel geometri, uygulamacıların algoritmik gereksinimlerini karşılayabilir mi? Yeni problem kaynaklarına sahip uygulama alanlarına girmeli midir? Bilişimsel geometri, kendi potansiyelini canlandırabilir mi? Çeşitlilikler gösteren büyük geometrik hesaplama dünyasında anahtar rol oynayabilir mi? gibi sorulara yanıt aranmaktadır.

Bilişimsel geometrinin gelişim süreci boyunca, temel geometrik problemlerin (dışbükey kabuklar, Voronoi şemaları, düşük boyutlu optimizasyon gibi) bazılarının çözümünde başarıya ulaşılmıştır. Zamanla bu tür problemlere yenileri de eklenmiştir. Bilişimsel geometri, dinamikliğini, üretkenliğini ve yapıcılığını hiçbir zaman kaybetmemiş olmasına rağmen her zaman hak ettiği yerin altında görülmüştür (Chazelle ve ark., 1996).

2.2 Bilişimsel Geometrinin Önemi ve Yararları

Bilişimsel geometrinin pratikte yararlı olmasının en önemli nedeni, iki ve üç boyutlu Öklit (Euclid) uzayının, aynı zamanda gerçek fiziksel nesnelerin düzenlendiği alan da olmasındandır. Bu nedenle pek çok alanda geometrik problemler vardır ve bu geometrik problemler için etkin algoritmalar geliştirilmektedir. Bu problemleri yüksek hızlı bilgisayarlarda etkin olarak çözmek için hızlı algoritma tekniklerinin uygulanması gerektiği kadar, yeni geometrik araçların da geliştirilmesi gerekmektedir. Yani bilişimsel geometri, sadece var olan birtakım teoremlerin bilgisayar programlarına çevrilmesi olayı değildir.

Günümüzde başta bilgisayar grafikleri, görüntü işleme, şekillerin yeniden oluşturulması, geometrik modelleme, katı modelleme, bilgisayarın görmesi, coğrafi bilgi sistemleri, robotbilim, üretim ve ürün tasarlama, moleküler biyoloji, astrofizik olmak üzere pek çok alanda

geometrik hesaplamalardan yararlanılmaktadır. Bilgisayar donanım ve yazılımlarında görülen gelişmelere paralel olarak, bilişimsel geometri, önümüzdeki on yılda önemini daha da artıracaktır.

Bilişimsel geometri, uygulamalardaki pratik problemlerin anlaşılmasında ve çözülmesinde yeni tekniklerin kullanılmasını sağlar, geometrik hesaplamaların yapılmasında yeni ve geliştirilmiş yöntemler sunar ve bilişimsel geometri, geometrik hesaplamaların algoritmik gereksinimlerine ayrılmış olan tek alandır (Chazelle ve ark., 1996).

2.3 Bilişimsel Geometrinin Kullanıldığı Alanlar

Bilişimsel geometrinin kullanıldığı alanların en önemlileri şunlardır :

- Bilgisayar Grafikleri (Computer Graphics)
- Geometrik Modelleme ve Spline'lar (Geometric Modeling and Splines)
- Katı Modelleme (Solid Modeling)
- Görüntü İşleme ve Örüntü Tanıma (Image Processing and Pattern Recognition)
- Şekillerin Yeniden Oluşturulması (Shape Reconstruction)
- Bilgisayarın Görmesi (Computer Vision)
- Çizge Çizimi (Graph Drawing)
- Coğrafi Bilgi Sistemleri (Geographical Information Systems = GIS)
- Hareket Planlama (Motion Planning)
- Robotbilim (Robotics)
- Tasarım, Üretim ve Ürün Tasarlama (Design, Manufacturing and Product Design)

Bu alanlar dışında dayanıklılık (robustness), moleküler biyoloji (molecular biology) ve astrofizik (astrophysics) gibi birçok alanda

kullanılmakta, bu alanlardaki problemlerin çözülmesine katkıda bulunmaktadır. Böylece birçok alanda önemli bir yer edinmiştir.

2.3.1 Bilgisayar Grafikleri

Bilgisayar grafiklerinin, bilişimsel geometri tekniklerinin en önemli uygulama alanı olduğu söylenir. Her iki alanın da gelişim süreçleri arasında büyük benzerlikler vardır. Her ikisinde de tarama hattı (sweep-line) ve alan bölümlleme (area subdivision) algoritmaları gibi bazı yöntemler birbirinden bağımsız olarak aynı zamanda ortaya çıkmıştır. Her iki alan da hesaplama modelleri farklı olmasına rağmen benzer problemler üzerine odaklanmışlardır. Örnek olarak, gizli yüzeylerin elenmesi veya başka bir ifadeyle görünen yüzeylerin belirlenmesi bilgisayar grafiklerindeki temel problemlerdendir. Bu problem aynı zamanda bilişimsel geometri konusunda çalışan pek çok araştırmacıyı da motive eden bir unsur olmuştur. Ayrıca her iki alanda yaşanan gelişmeler benzeri problemler için farklı gereksinimlerin ortaya çıkmasına yol açmıştır. Bu nedenle her iki alan arasındaki benzerlik ve farklılıkların belirlenmesi önemlidir. Bilişimsel geometri temelde, kesin sonuçları üretmek üzere ideal geometrik varlıkların etkin olarak nicel gösterimi ve işlenmesi ile ilgilenir. Bilgisayar grafikleri bu hedefleri kısmen paylaşmakla birlikte grafik uygulamacıları, ışık ve nesneler arasındaki etkileşimi ve bu etkileri oluşturan ortamı da modellerler. Grafik araştırmacıları ve uygulamacıları bunun dışında;

- i) Geometrik gösterimlerde çoğunlukla kesinlik yerine sonlu bir duyarlık kullanırlar,
- ii) Örnekleme stratejileri ve sayısal yöntemler yerine nadiren kapalı formda çözümler bulurlar,

iii) Algoritmalarındaki tasarımlarında işletim süresi ve çözüm kalitesi arasında bir tercih gerekliliği çoğunlukla ve açık bir şekilde göze çarpmaktadır,

iv) Genelde kendi önerdikleri algoritmaların gerçekleştirimini yaparlar.

Bilgisayar grafiklerinde hesaplamaya dayalı en yoğun işlemler yüzey kaplama (rendering) alanında yapılmaktadır. Yüzey kaplama, bazıları önemli ölçüde hesaplama darboğazı oluşturan geometrik problemlerle dolu bir alandır. Bilgisayar grafikleri modelleme ve simülasyon gibi iki temel alt alana bölünebilir. Modelleme, yüzey tanımlamaları ve özellikleri ile, içinde bulundukları ışık miktarı ve ortamla ilgili iken, simülasyon, bu bilişimsel modelleri davranışları tahminlemek üzere uygular. En önemli simülasyon çeşitlerinden biri yüzey kaplamadır. Bu yöntemde, sahnenin tanımı ve ışık kaynakları, bir veya bir dizi belirlenmiş bakış noktasına göre yapay resmi oluşturmak üzere işleminden geçirilir. Yüzey kaplama konusu, zengin bir bilişimsel geometri problem kaynağıdır. Görünürlük işlemini de içerir. Bu işlem sırasında, bakış noktasından görünen yüzey ve/veya yüzey parçaları belirlenir. Ardından gölgelendirme işlemi yapılır. Bu aşamada görünen her yüzeydeki örnek noktalardan renk değerleri hesaplanır.

2.3.2 Geometrik Modelleme ve Spline'lar

Model, yapay olarak oluşturulmuş ve bir başka nesneyi kolaylıkla gözlemlemeyi sağlayan bir nesnedir. Gözlemlemeyi etkinleştirmek için bina, gemi, araba gibi üç boyutlu fiziksel nesnelerin göreceli boyutları ve genel görünüşleri korunarak gerçek boyutlarında olmayan fiziksel modelleri oluşturulur. Kimyada molekül içindeki atomların, diğer atomlarla olan göreceli dizilişleri de molekül modelleri sayesinde gözlemlenebilir. Moleküllerin bunlar dışındaki özellikleri molekül modeline konulmayarak belli özelliklere odaklanılması sağlanır. Bilim ve

mühendisliğin değişik alanlarında yaygın olarak kullanılan matematiksel modeller, sayısal veri ve denklemler aracılığı ile modellenmiş bir olayın davranışlarını temsil eder. Bilgisayar ortamında hazırlanan ve “Bilgisayar Modeli” olarak adlandırılan modeller çok sayıda veri içerirler. Bilgisayar modeli, mühendislik çizimleri de dahil birçok alanda kullanılır. Geometrik modelleme, nesne modelinin geometrik kısmını içeren bir alt kümesidir. Nesnelerin bakış noktasından görünen parçalarını belirlemek ve her bir noktanın rengini bulmak gibi geometrik problemler bu konunun bir parçasıdır. Katı modelleme, geometrik modellemenin bir dalı olup fiziksel katı nesnelerin tam gösterimini içerir. Geometrik modelleme, bilgisayar destekli tasarımdan (CAD) bilgisayar grafiklerine, bilimsel görselleştirmeden (scientific visualization) görüntü işlemeye kadar birçok alanda kullanılmaktadır.

Spline eğrileri, parçaların sınırlarında belirlenmiş süreklilik şartlarını sağlayan polinom parçaları ile biçimlendirilen bileşik eğrilerdir. Spline oluşturma problemi, verilerden interpolasyon (aradeğerleme) veya yaklaşım yolu ile sonlu elemanlar oluşturulmasını içerir ve geometrik modellemenin en önemli araştırma problemidir. Spline yüzeyleri ise birbirine dik olan iki Spline eğrileri kümesi ile tanımlanır. Spline'lar, bilgisayar destekli tasarımdan bilgisayar grafiklerine, verilerin görselleştirilmesinden geometrik modelleme ve görüntü işlemeye kadar birçok alanda kullanılmaktadır. CAD uygulamalarında özellikle otomobil gövdelerinin, uçak, gemi ve uzay aracı yüzeylerinin tasarımında kullanılır. Ayrıca bir sahnedeki nesnelerin veya kameranın animasyon yolunu yani hareketini tanımlamada da kullanılmaktadır.

2.3.3 Katı Modelleme

Geometrik modellemenin bir alt alanı olup üç boyutlu nesnelerin bilgisayarda tam olarak gösterimi ve üzerlerinde yapılan işlemler ile

ilgilenir. Yirmibeş yıl kadar önce geometri, yüzeyleri düzlemlerle ve koni, küre, silindir, simit gibi ikinci dereceden basit denklemlerle tanımlanan katı nesnelerle sınırlıydı. Bu tür nesneler mekanik tasarımda uzun yıllar yeterli görülmüştür. Bununla birlikte gemi ve uzay aracı inşa endüstrisinden gelen sürekli baskı, günümüzde bu teknolojiyi nesnelerin NURBS (Nonuniform Rational B-Splines) gibi yöntemlerle oluşturulmasına kadar ulaştırmıştır. Bu tür yöntemler günümüzde yaygın olarak kullanılmaktadır (Chazelle ve ark., 1996).

Katı modellemenin amacı, katı fiziksel nesnelerin üç boyutlu şekillerinin bilgisayarda temsili ve işlenmesidir. Katı modelleme, sistem gerçekleştirimi ve algoritma geleneği olan uygulamaya dayalı bir alandır. Önemli uygulama alanları arasında tasarım, üretim, bilgisayarın görmesi, bilgisayar grafikleri ve sanal gerçeklik sayılabilir. Bu alanın kaynakları arasında sayısal çözümleme, cebirsel hesaplama, yaklaşım teorisi, uygulamalı matematik, küme topolojisi, cebirsel geometri, bilişimsel geometri ve veritabanı vardır.

2.3.4 Görüntü İşleme ve Örüntü Tanıma

Görüntü işleme, fotoğraflar ve televizyon görüntüleri gibi var olan resimleri değiştirmek ve yorumlamak için birçok teknikleri kullanır. Görüntü işlemenin en önemli iki uygulama alanı (1) resim kalitesini artırmak (2) robotbilimde kullanıldığı gibi makinelerin görsel bilgiyi algılamasını sağlamaktır.

Bir örüntü tanıma sistemindeki en önemli iki problem özellik belirleme (feature extraction) ve sınıflandırmadır (classification). Sayısal bir resimden vektör veya şekil ölçümlerinin belirlenmesi problemi üç alt probleme ayrılabilir : (Goodman ve O'Rourke, 1997)

- Resmi parçalama yani ilgilenilen nesnelerin zeminden ayrılması problemi. Bu problem, sabit boyutlu uzayda n tane noktayı $m \ll n$ olacak şekilde m tane doğal gruba parçalama işlemidir. Burada kümelenendirme analizi (cluster analysis) yöntemlerinin önemi büyüktür.
- Parçalara ayrılmış resimdeki nesnelerin bulunması problemi.
- Tespit edilen nesnelerin şekillerinin belirlenmesi.

Sınıflandırmada nesnelerin özelliklerine bağlı olarak kullanılan birçok araç bulunmaktadır. Hough dönüşümü, çokgensel yaklaşım, çokgensel parçalama bunların en önemlilerindendir. Sınıflandırma problemi, özellik vektöründen sınıflandırma ile ilgili olarak etkin karar verme kurallarını tasarlamayı içerir. En güçlü karar verme kuralları, özellik vektörlerinin dağılımı üzerinde hiçbir varsayımda bulunmayan ve aynı zamanda parametrik olmayan kurallardır. Bunların en çok bilineni, en yakın komşu (NN, nearest neighbor) kuralıdır (Goodman ve O'Rourke, 1997).

2.3.5 Şekillerin Yeniden Oluşturulması

Birçok uygulama alanında, nesnelerin iki boyutlu kesitlerinden üç boyutlu modellerinin oluşturulmasına gereksinim duyulmaktadır. Örneğin tıbbi görüntüleme, mikroskop incelemesi, jeoloji ve uzay üretimi gibi alanlarda şekillerin yeniden oluşturulmasından önemli ölçüde yararlanılmaktadır. Bu amaçla iki yaklaşım geliştirilmiştir. Birincisi, görüntü işleme tekniklerinden esinlenerek veriyi kesit görüntülerinin üst üste konulması ile elde edilmiş üç boyutlu görüntü olarak görür. Bu yaklaşım, hacim veya üç boyutlu nokta (voxel) tabanlıdır; her noktanın görünürlüğü için interpolasyon uygulanır veya aynı (x,y) koordinatlarına sahip tüm noktalar sütununa dayanır.

İkinci yaklaşım, daha çok sınır duyarlıdır; kesitlerin sınırlarının interpolasyonunu alarak nesnenin çokyüzlü bir modelini oluşturmaya çalışır. Dilim elemanlarını dikkate alır ve daha sonra sıradaki her ardışık dilim çifti için tüm katman interpolasyonlarını birleştirir. Karışık yöntemler de kullanılabilir : Örnek olarak yapılandırılmamış bir noktalar bulutunu yüzeye uyduracak hale getirebilir (Chazelle ve ark., 1996).

Bellek gereksinimlerinden dolayı pek çok sınırlaması olmasına rağmen, ilk yaklaşım geçen on yılda daha çok tercih edilmiştir. UCLA tıp merkezinde, bir insan kafatası bu yöntemle tekrar oluşturulmuştur : Kafatası 800,000 üçgenden oluşmuştur ve bu sayı bilgisayar ekranında görselleştirmede sorun çıkartmayacak derecede yeterince küçük fakat işlemek için ise büyüktür. Görüntünün pratik olarak oluşturulması için etkin geometrik çözümler, seksenlerin sonlarında ortaya çıkmaya başlamıştır (Chazelle ve ark., 1996).

Bu tür yöntemlerin çoğu, bilişimsel geometriciler tarafından geliştirilmiş yeni veri yapılandırma tekniklerini kullanırlar. Örnek olarak Voronoi Şemasına dayalı bir yöntem INRIA'da geliştirilmiştir. Bu yöntem ardışık dilim çiftlerinin Delaunay üçgenleştirmesi ile ilgili hesaplamaları içerir ve her çift elemanın diğerine izdüşümünü alarak, ardışık dilimler arasındaki boşluğu dört kenarlı hale getirir (Chazelle ve ark., 1996).

Yazılım iki şirket (NOESIS ve CRIL) tarafından ticarete kazandırılmıştır. "NOESIS" tıbbi görüntüleme ve endüstri için genel amaçlı yazılım üretirken "CRIL" üç boyutlu tıbbi görüntülerin (yeniden oluşturma algoritmasının da dahil olduğu) işlenmesi için büyük bir kütüphane tutmaktadır; ftp ile ticari olmayan sürümüne erişilebilir ve 300 farklı bölgede kullanılmaktadır. Program, nöron araştırmalarında kullanılmaktadır. Yazılımın kullanıldığı diğer bazı uygulamalar da şunlardır : (Chazelle ve ark., 1996)

- Radyasyon terapisi planlama yazılımının klinik kullanımı (DKFZ Alman kanser araştırma merkezi)
- Köpekbalığı böbreğinin yeniden oluşturulması (Max Planck Institut für Malekulare Biologie)
- Köpek kalbi için sonlu eleman yöntemi (Purdue Üniversitesi)
- Kalp hareket modeli (Yale Üniversitesi)
- Beyin MRI (Colorado Üniversitesi)
- Vücut yüzeyi potansiyeli haritalandırması (Dalhousie Üniversitesi)
- Aşınan yüzeylerin hacim hesaplaması (Utah Supercomputing Institute)
- Kemik yeniden yapılandırılması (Center for Medical Robotics & Computer Assisted Surgery, Carnegie Mellon Üniversitesi)
- Tıbbi verilerden hızlı prototip oluşturulması (Yüksek Teknoloji Üniversitesi, Kıbrıs)
- Sonlu eleman yöntemi ile ağ oluşturulması (Waikoto Üniversitesi, Yeni Zelanda)
- Beyin yüzeylerinin parçalarının döşenmesi (Washington Üniversitesi)
- Atardamarların MRI taramalarının yeniden oluşturulması (Waterloo Üniversitesi)

Son olarak İsrail’de alternatif bir yaklaşım izlenmektedir. Bu yaklaşım bilgisayarın görmesindeki nesne tanıma problemleri için geliştirilmiştir ve “geometric hashing” düşüncesine dayanır. Bu yazılım paketi değişik karmaşık katıların yeniden yapılandırma problemleri üzerinde yoğun olarak test edilmektedir (Özellikle tıbbi verilerin görüntülenmesinde ve arazi yükseltileri çevre verisinden alanın yeniden oluşturulmasında). Dayanıklı ve karmaşık durumlara uygulanabilir

olduğu kanıtlanmıştır ve doğru sonuçlar verdiği anlaşılabilmektedir (Chazelle ve ark., 1996).

Son olarak tıbbi uygulamalar, geometrik yeniden oluşturma ve analiz problemleri zenginliği ile bu alanı motive etmektedir. Örnek olarak bir organın hacminin tahminlenebilmesi için yeniden oluşturulması gerekebilir. Tıp alanındaki geometrik problemlerdeki ilerlemeler, bilgisayar destekli cerrahi gibi alanlarda büyük değişimlere yol açacaktır (Chazelle ve ark., 1996).

2.3.6 Bilgisayarın Görmesi

Bilgisayarın görmesi ile ilgili problemlerin pek çoğu doğal olarak bilişimsel geometriye de girer. Ancak bilişimsel geometricilerin bu problemlerle uğraşmama nedeni ise anlayamamaktadır. Belki bu ilgisizliğin yanıtını uygun matematiksel formüllerin bulunamamasında aramak gerekecektir. Bilgisayarın görmesi konusunda, soruların biçimlendirmesindeki uygun soyutlama düzeyinin bulunması araştırma çabalarının önemli bir kısmını oluşturur. Bilgisayarın görmesi aynı zamanda bilgisayarın insan beyninden geri kaldığı az rastlanır özelliklerden (ses tanıma gibi) birisidir : İnsanlar gerçekten de görme konusunda bilgisayarlardan ileridirler. Bunun doğrudan yararı, insanların çıktıya bakarak programın kalitesini etkin olarak sınayabilmeleridir. Bu alanın geometrik algoritmaları kullanan başlıca iki alt alanı vardır : Model-tabanlı tanıma (veya desen eşleme) ve iki boyutlu görüntülerinden üç boyutlu görüntülerin elde edilmesi (özellikle stereopsis ve structure from motion) (Chazelle ve ark., 1996).

2.3.7 Çizgelerin Çizimi

Çizge çizimi, yazılım mühendisliği, veritabanı sistemleri, görsel arayüzler ve bilgisayar destekli tasarım gibi bilgisayar alanındaki önemli

uygulamalardaki geometrik gösterimlerin oluşturulmasına yöneliktir. Çizgelerin çizilmesi üzerinde yapılan araştırmalar, ayrık matematik (topolojik çizge teorisi, geometrik çizge teorisi), algoritma (çizge algoritmaları, veri yapıları, bilişimsel geometri, VLSI) ve insan-bilgisayar etkileşimi (görsel diller, grafiksel kullanıcı arayüzleri, yazılım görselleştirme) gibi birbirinden çok farklı alanları kapsamaktadır (Goodman ve O'Rourke, 1997).

2.3.8 Coğrafi Bilgi Sistemleri

Coğrafi bilgi sistemleri (GIS), bilişimsel geometri araştırmalarını destekleyici bir şekilde artarak kullanılmaya başlanmıştır. Bir GIS, karmaşık planlama ve yönetim problemlerinin çözümünde uzaysal-başvurulu verilerin yakalanmasında, yönetiminde, işlenmesinde, analizinde, modellenmesinde ve görüntülenmesinde kullanılmak üzere tasarlanmış donanım, yazılım ve prosedürlerdir (Chazelle ve ark., 1996).

Asıl önemli olan uzaysal-başvurulu kısmıdır. Bu kısım değişik özellikler/nesneler hakkındaki farklı kaynaklardaki farklı türlerdeki veri setlerini birleştiren bir veri tabanı sistemidir. Sistemin işlem yükünün çoğunu veri setlerini genel bir forma dönüştürmek oluşturur (Chazelle ve ark., 1996).

Veri uzaysal bir yere atanabilir fakat haritada veya bilgisayar ekranında veri setlerinin aynı anda görüntülenmesi ile birleştirilebilir. Görüntülemek üzere veri setlerini seçme mekanizması eklendiğinde karşımıza ilkel bir GIS çıkar (Chazelle ve ark., 1996).

Doğal kaynak yönetimi konusunda birçok sistem geliştirilmiştir. Bu sistemlerden bazıları çok sayıdaki ve büyük kağıtlardaki harita envanterlerinin kullanımı ile elde edilemeyecek yönetim kararlarının verilmesinde oldukça başarılı görülmüştür. GIS, genişleme eğilimindeki

pazarlama ve hükümet uygulamalarında kullanılır (Chazelle ve ark., 1996).

Şüphesiz ilkel bir GIS'ta eksik olan şey şudur : “Bir GIS, veriyi işleyen ve görüntüleyen temel bir araçtır. Arındırmaz, bakımını yapmaz ve makul olmamasına bile bakmaz. Bilişimsel geometricilerin, geometrik verilerin analizini destekleyen algoritmaları ve veri yapılarını sağlama yönünden katılımları olur (Chazelle ve ark., 1996).

GIS ve bilişimsel geometri birbirlerine paralel olarak gelişmişlerdir. Bilişimsel geometriciler GIS'ı bilişimsel geometrinin pratik gerçekleştiriminin örneği olarak görme eğiliminde iken tarihsel açıdan bu tam doğru değildir. GIS ve bilişimsel geometri toplulukları pek çok konuda geometrik hesaplama anlayışlarını bağımsız olarak geliştirmişlerdir (Chazelle ve ark., 1996).

Bazı anahtar örneklerde gelişim bilişimsel geometri alanından önce GIS alanında olmuştur (Chazelle ve ark., 1996). Örneğin bilgisayarlı kartografiya 1960'larda doğmuştur. Pek çok temel kavram (harita katmanları, topolojik yapı...) Kanada hükümetinin “The Land Inventory” kolunda veya Harvard Laboratuvarlarının Bilgisayar Grafikleri ve Uzay Analizi Bölümünde yapılan çalışmalara dayanmaktadır. ESRI'nin ARC/INFO'su en popüler ticari GIS sistemidir ve Harvard Lab. Teknolojisinde başlatılmıştır (Chazelle ve ark., 1996).

Bilişimsel geometri ve GIS arasındaki etkileşim son yıllarda artmıştır. İki yılda bir düzenlenen uzay verileri yönetimi üzerine uluslararası sempozyumda bilişimsel geometri yayınları kabul edilmektedir. ESRI, Caliper Corp. ve diğer şirketler bilişimsel geometri topluluklarından kiralama yapmakta ve araştırma sonuçlarını almaktadırlar. Bu alana ilişkin genel bir şikayet bilişimsel geometri konusundaki yayınların gecikmesi ve ders kitabı eksikliğinin araştırma

sonuçlarını GIS gibi diğer alanların benimsemesini engellemesidir (Chazelle ve ark., 1996).

GIS alanında doğan geometrik problemlere bakıldığında iki şey akılda tutulmalıdır. İlki, bilişimsel geometricilerin kendilerini geometrik problem çözücüler olarak görmelerine rağmen herhangi bir özel problemin çözümünden çok, geliştirilen uzay, şekil ve geometrik hesaplama kavramlarına önem vermeleridir. İkincisi, geometrinin coğrafi bilgi sistemlerinin sadece bir bölümünü oluşturmasıdır. Standart GIS hizmetleri, otomatik harita üretimi, harita analizi, envanter, uzaysal analiz ve uzaysal karar destek olarak belirtilebilir (Chazelle ve ark., 1996).

2.3.9 Hareket Planlama

Hareket planlama robotbilimin temel prensiplerindendir. Çok değişik şekilleri olmakla birlikte en basit biçimi şöyledir : Birbirine eklemeler, menteşeler ve değişik bağlantılarla bağlı veya bağımsız hareket eden birçok katı nesneden oluşan B robot sistemi ile, engellerle dolu iki veya üç boyutlu V ortamı verildiğinde, B'nin Z_1 başlangıç konumundan Z_2 hedef konumuna çarpışma olmadan ulaşabileceği bir yol olup olmadığının belirlenebilmesi ve bu tür bir hareketin planlaması hareket planlama olarak adlandırılır. Planlama sisteminde engellerin yeri ve şekli ile B'nin şeklinin verildiği varsayılır. Bu basitleştirilmiş ve tümüyle geometrik düzende, eksik bilgi, algılamada ve hareketlerde yanlışlık ve hareketli engeller olmadığı da varsayımlar arasına konulabilir. 1980'lerin başından günümüze kadar hareket planlama, robotbilim ve bilişimsel geometri çalışmalarında önemli yer tutmaktadır (Goodman ve O'Rourke, 1997).

2.3.10 Robotbilim

Robotbilim, çok değişik düzenlerdeki fiziksel nesnelerin hareketlerinin bilgisayar tarafından kontrolünün sağlanması ile ilgili alandır. Fiziksel nesneler üç boyutlu uzayda dağılım gösterdiklerinden geometrik gösterimleri ve hesaplamaları robotbilimde önemli rol oynamaktadır. Sonuçta bu alan, bilişimsel geometrideki pratik problemler için zengin bir alandır. Aralarında büyük farklar olmakla birlikte robotbilimdeki araştırmalar ile bilişimsel geometri araştırmaları birbirine bağlı problemler üzerinde çalışırlar. Robotbilim araştırmacıları ağırlıklı olarak pratikte iyi işleyen yöntemlerin geliştirilmesi ile ve bütünleştirilmiş sistemler haline dönüştürülmesi ile ilgilenirler. Bilişimsel geometri araştırmacıları ise temeldeki eleman düzenleri, kombinasyonu ve karmaşıklık gibi kavramlar ile ilgilenirler. Robotbilimde parçaları tutma, parçaları sabitleme, malzemeleri makineye verme, parçaları birleştirme ve yol ve hareket planlama gibi birçok problem vardır (Goodman ve O'Rourke, 1997).

2.3.11 Tasarım, Üretim ve Ürün Tasarlama

Bilişimsel geometri tekniklerinin bilgisayar destekli tasarım ve üretim alanındaki problemler üzerine uygulamaları konusunda yapılan son çalışmalar genelde kalıp tasarımı, sayısal kontrollü makineler ve doğruluk kontrolü ile ilgilidir (Goodman ve O'Rourke, 1997).

Kalıp döküm ve enjeksiyon işlemleri, çok değişik alanlardaki ürünlerin seri üretiminde kullanılır. Burada, parçanın CAD modelinden kalıbın tasarlanması anahtar adımdır. Çünkü işlemin hızını ve ürünün son halinin kalitesini belirleyen budur (Goodman ve O'Rourke, 1997).

Sayısal kontrollü (NC) üretimde makine parçaları, CAD modelinden alınan bilgiye dayalı olarak bilgisayar kontrolü altında

üretir. Bu tür üretimi gerçekleştiren makinelere örnek olarak freze makinesi ve torna tezgahı verilebilir. Buradaki önemli nokta, aracın parçaya erişimi ve belirli optimizasyon kriterini sağlayacak araç yollarının bulunmasıdır (Goodman ve O'Rourke, 1997).

Üretilmiş bileşenin ideal modeldeki şekil, boyut ve özelliklerinin konumlarının doğruluğunun kontrolü de bilişimsel geometri tekniklerini gerektirir. Bu alandaki Bilişimsel geometri araştırmaları değişik şekillerin toleranslarını (doğruluk, düzgünlük, eğrisellik veya silindiriklik) denetleyen etkin algoritmalar üzerine odaklanmıştır (Goodman ve O'Rourke, 1997).

Ürün tasarımında da bilişimsel geometri önemli yer tutmaktadır. Araştırma alanında Bath üniversitesi birkaç yıl önce genel cebirsel yüzey elemanları ile modelleme konusunda çalışmalar yapılmaya başlanmıştır. Cebirsel yüzeylerin parametrik yüzeylere göre matematiksel avantajları vardır ve gelecek vadeden etkin bir araştırma alanı olarak ortaya çıkmaktadır. En bilinen ticari proje olarak Boeing'in 777 uçak tasarımı gösterilmektedir. Bu tasarım tümüyle geometrik ve katı modelleme (yanında mühendislik analizi, gider analizi ve geometrik olmayan diğer teknikler) kullanılarak gerçekleştirilmiştir. Boeing 777, CATIA üzerinde modellenmiştir. CATIA, Fransa tarafından üretilip ABD'de IBM tarafından pazarlanan ticari bir sistemdir. CATIA, 25 yıllık bir teknolojiye dayanmasına rağmen ancak yakın zamanlarda üretimde kullanılması araştırmaların üretimde gecikmeli olarak kullanımına ilginç bir örnektir (Chazelle ve ark., 1996).

Bilişimsel geometrinin gelişiminde genel olarak doğrusal nesneler üzerine yoğunlaşmıştır. Ancak üretimde böyle bir sınırlandırma yapmak kabul edilebilir değildir. Bu nedenle bu boşluğu kapatacak şekilde katı modelleme çalışmaları yürütülmektedir. Üretimde katı modellemede günümüzde çalışılan konulardan biri, değişen yaklaşımlara göre verilen

kısıtları içeren yüzeylerin nasıl tanımlanacağı ile ilgilidir (Chazelle ve ark., 1996).

2.4 Bilişimsel Geometrinin Gelecekte Göstereceği Gelişim

Bilişimsel geometri alanında çalışan kişiler, sadece karmaşıklığa (complexity) değil, kod sağlamlığı (code robustness), duyarlılık (precision), CPU zamanları (CPU times), standartlar (standards), “Benchmark’lar” (Benchmarks), ve dağıtıklık (distribution) konularına da önem vermeye başlamışlardır. Geometrik problemler, algoritmik tabanda sadece bilişimsel geometri yardımıyla anlaşılabilirler. Mühendisliğin pek çok alanındaki -şu anki ve gelecekteki- darboğazlarını aşmada anahtar rolü üstlenecek kişiler de bu konuda çalışanlar olacaktır. Ayırık geometriciler bile, sadece sayılabilir geometrik yapılar üzerine yoğunlaşmışlardır (Chazelle ve ark., 1996).

Bilişimsel geometri dışarıdan bakıldığında, ulaşılmayı bekleyen yeni ufuklar olarak görülmektedir. Şu ana kadar yapılmakta olan çalışmalar oldukça önemlidir. Teorik ve pratik, çeşitli alanlarda pek çok önemli açık problem bilişimsel geometri yardımı ile çözülmeyi beklemektedir (Chazelle ve ark., 1996).

3. BİLGİSAYAR GRAFİKLERİ

3.1 Bilgisayar Grafiklerinin Günümüzdeki Durumu

Bilgisayar yazılım ve donanımlarında son yıllarda görülen gelişmeler, bilgisayar grafikleri ve animasyon işlemlerindeki kalite ve hızı oldukça artırmıştır. Bu şekilde günlük yaşamdan değişik bilim dallarına kadar her alanda etkisini göstermeye başlayan bilgisayar grafikleri alanına duyulan ilgi ve gereksinim giderek artmakta; bu alan standartların oluşması ve yeni kavramların eklenmesi ile her geçen gün zenginleşmektedir (Uğur, 1996).

Bilgisayar grafikleri bilgisayar üzerinde grafiksel görüntüler oluşturmaya yönelik çalışmaların yapıldığı bir alandır. Günümüzde bilgisayarlar, çizim yapmak, görüntü, grafik ve animasyonlar oluşturmak ve bunlar üzerinde işlemler yapabilmek için çok kullanışlı ve güçlü araçlar haline gelmişlerdir.

Yeni 3B (üç boyutlu) grafik hızlandırıcılar (nVidia'nın GeForce ve S3'ün Savage2000 yongaları gibi), dönüştürme (transformation), aydınlatma (lighting), kurulum (setup) ve özellikle kaplama (rendering) aşamalarındaki performansları ile düzgün görüntüler elde etmek için kullanılan yöntemlerden (antialiasing) doku eşlemeye kadar birçok işlemi başarı ile gerçekleştirmektedirler. Her geçen gün üç boyutlu kartlar tarafından gerçekleştirilen grafik işlemlerine (alpha blending, environment mapping, fogging, shading, Gouraud shading, lens flaring, texture mapping, mip mapping) yenileri eklenmektedir. Günümüzde PC'lerde üç boyut teknolojisinin geliştirilmesinde itici güç olarak bilgisayar oyunları gösterilmekle beraber yakın bir gelecekte 3B uygulamaların işletim sisteminin bir parçası olması ile iş uygulamaları da üç boyutlu olacaktır.

3.2 Bilgisayar Grafiklerinin Kullanıldığı Alanlar

Yeni bin yıla girdiğimiz bugünlerde bilgisayar grafikleri birçok alanda yaygın olarak kullanılmaktadır. Bu alanlardan en önemlileri aşağıda belirtilmiştir.

3.2.1 Bilim ve Bilimsel Görselleştirme

Bilim adamları, mühendisler, tıp personeli, iş analistleri ve diğerleri çok miktarlarda veriyi analiz etmek durumunda kalırlar. Bu gibi alanlardaki veri setlerinin grafiksel görüntülerinin oluşturulması bilimsel görselleştirme olarak adlandırılır. Bilimsel görselleştirmenin birkaç örneği şu şekilde verilebilir :

- Uzay çalışma ve araştırmalarında uzaya ilişkin fizik yasaları ve hareketler bilgisayara yüklenir. Sonra uzay araçlarının yörüngeleri bulunur, uzay araçlarının içindeymiş veya dışından bakıyormuş gibi görüntüler elde edilir. Kalkış, iniş, çarpışma gibi animasyonlar oluşturularak uzay araçlarını tehlikeye atmadan yapılan hesaplamaların doğruluğu sınanır. Gök cisimlerinin hareketlerinin animasyonları da yapılan araştırmaların görselleştirilmesi açısından yararlıdır.
- Kemiklerin, kasların, organların, vücuttaki sistemlerin üç boyutlu olarak görselleştirilmesi ve hareket ettirilmesi tıbbi araştırmalarda kullanılır ve hasta ya da yaralıların durumları konusunda daha net bilgilerin elde edilmesini sağlar. Bilgisayar modeli yardımıyla tanı koyma ve protez yerleştirme gibi işlemler çok daha kolay ve sağlıklı yapılır.
- Karmaşık DNA yapılarının modellerine sabit bir noktadan bakıldığında anlaşılması zordur. Bilgisayar ekranında etkileşimli olarak döndürüldüğünde daha rahat anlaşılabilir.

- Biyoloji uygulamalarında moleköl düzenlerinin bilgileri, veri tabanı bilgisayara girildiğinde ekranda grafiğı oluşturulup değışik noktalar referans alınarak döndürölüp hareketlendirilebilir.
- Otomobil ve uçak kazaları gibi karmaşık olay dizilerinin görselleştirilmesi. Mahkemelerde, fizik kanunlarından yararlanarak kazaların grafiksel olarak canlandırılması, kazaların nedenleri ve nasıl gerçekleşdikleri konusundaki şüpheleri ortadan kaldırarak gerçeklere ulaşılmasını sağlar.
- Değışik şirketlerin aylık satışlarına ilişkin verilerin üç boyutlu sütun grafiklerine dönüştürölmesi ile şirketlerin satış eğilimleri kolay bir şekilde görölür ve şirketler arasında kolaylıkla karşılaştırmalar yapılır. Görsel bir grafik, sayısal verilerin incelenmesi ile elde edilemeyecek bilgilerin bir bakışta görölmesini sağlar.
- Arkeolojide geçmişe ışık tutmak üzere antik bir yapının tamamının görselleştirilmesi. Gerektiğinde sanal olarak oluşturulan bu ortamda hareket edilerek insanların yaşayışlarına ilişkin bazı ipuçları bulunabilir. Yapının her bir bölümünün ne amaçla yapıldığı veya nasıl kullanıldığı konusunda sonuçlara ulaşılabilir. Yapıların bilgisayar modelleri teknolojik gelişmeye paralel olarak maketlerine göre daha kısa sürede yapılabilmektedir ve daha avantajlıdır. Örnek olarak Mısır'daki antik Karnak kentinin küçültölmiş bir modeli 24 ayda tamamlanabilmişken, bilgisayar modeli 16 ayda bitirilmiştir.
- Nesli tükenmiş eski çağlara ait canlıların üç boyutlu modellerinin oluşturulması ile yaşayışları, yapıları, hareketleri vs. konularda bilgi edinilebilir. Fiziksel kanunların grafiksel modeller üzerinde uygulanması yoluyla mantıksız durumlar belirlenerek yıllar boyunca doğru olduğı varsayılan bilgilerin yanlış olduğı bulunabilir. Örnek olarak paleontologlar varsayımlarına dayanarak belirledikleri kas,

eklem ve iskelet yerleşim durumlarına göre sanal bir dinozoru yürütmeye çalışmış ancak fizik kanunlarına göre başaramamışlardır. Kas ve eklem dağılımının yanlış yapıldığının anlaşılması üzerine denemelerle doğrusu bulunmuştur.

- Hava durumu sistemlerinde yüksek ve alçak basınç bölgeleri üç boyutlu grafiklerle gösterilebilir. Fırtınalardaki hava akımlarının sayısal modelleri oluşturulabilir. Uçakların gövdesi ve kanatlarındaki basınç ve akımlar görselleştirilebilir.

3.2.2 CAD/CAM

Bilgisayar Destekli Tasarım / Üretim (CAD/CAM) kısaca, tasarımda ve üretimde bilgisayarlardan yararlanılması ile gerçekleştirilen tasarım ve üretimdir. Tasarımda ve üretimde bilgisayarların kullanılması bu alanlara yeni bir boyut getirmiştir. Bilgisayar destekli tasarımın ve üretimin kullanıldığı alanlardan bazıları şu şekilde sıralanabilir :

- Binaların, kampüs, endüstri kompleksi gibi bina gruplarının tasarımında ve üretiminde bilgisayar grafiklerinden yararlanılmaktadır. Bazı mimari projeler üç boyutlu olarak tasarlanıp geliştirilmektedirler. Tasarlanan yapıların durağan gerçekçi görüntüleri ve hareketli kamera veya bakış noktasına göre oluşturulan hareketli görüntüleri müşterilerin, yatırımcılarının ve planlama bölümlerinin kullanımına sunulmaktadır. Bir ev tasarlandıktan sonra yapımına geçilmeden önce bilgisayar animasyonu yardımıyla çevresinde ve içinde dolaşılabilen ve herhangi bir noktadan, günün herhangi bir saatindeki güneş ışığı yansımalarına kadar görüntülenebilmektedir.
- Şehir planlamalarında da bilgisayar destekli tasarım ve üretimden yararlanılmaktadır. Şehirlerde metro hatlarının kurulması,

stadyumların, alışveriş merkezlerinin inşa edilmesi gibi konularda hata yapılmaması için bilgisayar destekli tasarımdan yararlanılır. Aksi halde yıkıp tekrar yapmak mümkün olmayabilir. Bilgisayar üzerinde oluşturulan bir şehir modeli üzerinde caddelerde dolaşılabilmekte, günün herhangi bir saatindeki her türlü veri göz önünde bulundurularak şehrin herhangi bir bölümündeki durumlar görülebilmektedir. Gerekğinde bazı koşullar değiştirilerek model üzerinde yapılan işlemler sonucunda en avantajlı durumlar bulunabilmekte ve kararlar daha doğru verilebilmektedir. Örnek olarak şehrin caddelerinin kenarlarına ağaç dikilmesinin iyi olup olmayacağı model üzerinde görülebilir. On sene boyunca ağaçların büyüme durumlarına göre caddenin görüntüsü görülebilir ve oluşabilecek kötü durumlar fark edilirse başka bir tür ağaç dikilmesine veya hiç ağaç dikilmemesine karar verilebilir. Şehir modeli üzerinde deneme sürüşleri yapılarak cadde ve sokaklardaki düzenleme gereksinimleri ve hatalar belirlenebilir. Şehrin estetik görüntüsü güzelleştirilebilir.

- Kara, deniz, hava ve uzay taşıtlarının ve parçalarının tasarımında, üretiminde ve hareketli görüntülerinin oluşturulmasında, önceden performanslarının test edilip gerekli değişikliklerin yapılarak geliştirilmesinde bilgisayar grafikleri ile oluşturulan modelleri büyük önem taşımaktadır. Örnek olarak otomobil tasarımcıları, değişen ışık ve gölge durumlarına göre otomobilin görüntüsünü kolaylıkla oluşturabilecek karmaşık hesaplamaları yapabilecek yazılımları kullanırlar. Tasarımcılar bu yazılımları motorun yerleştirileceği yeri belirlemekte de kullanabilmektedir.
- Bilgisayarların, bilgisayar parçalarının, elektronik, tekstil, gıda ve diğer alanlardaki ürünlerin tasarımında ve üretiminde bilgisayar grafiklerinden yararlanılması önemli yararlar sağlamaktadır. Üç

boyutlu modellemenin ürün tasarım şirketleri açısından büyük önem taşımasının nedeni müşterilerin sürekli değişkenlik gösteren isteklerini karşılamak üzere, renk, biçim ve ayrıntıları düzeltip değişikliklerin model üzerinde yapılmasının kolay olmasındandır.

3.2.3 Eğitim ve Öğretim

Fiziksel, kimyasal, matematiksel, sosyal, ekonomik ve finansal sistemlerin bilgisayarda oluşturulmuş modelleri eğitim ve öğretim açısından da çok yararlı olabilmektedir. Özellikle kara, deniz, hava ve uzay araçlarına benzetilerek hazırlanan özel sistemler yani simülatörler, insanların tehlikeye atılmadan bu taşıtların kullanımını öğrenmelerini çok ucuza sağlaması açısından oldukça yararlıdırlar. Ağır iş makinelerinin operatörlerinin ve trafik denetim personelinin eğitiminde ve benzeri konularda geliştirilmiş simülatörler de vardır. Askeri alanda sanal hareket ortamı oluşturularak gerçekleştirilen tatbikat simülasyonları maliyetleri ve riskleri en aza indirmektedir.

Öğrencilerin okudukları konuları canlı olarak bilgisayarda görmeleri öğrenmelerini kolaylaştıracaktır. Örnek olarak fizik derslerinde cisimlerin hareketleri, hızları ve ivmelerine ilişkin animasyonlar öğrencilerin öğrenmesine yardımcı olacaktır. Kimya derslerinde kimyasal tepkimeler; biyoloji derslerinde hücre bölünmeleri, canlıların hareketleri; matematikte vektörler; coğrafyada dünyanın hareketleri; tarihte göçler bilgisayar animasyonu yardımı ile anlatılabilecek konulara sadece birkaç örnektir. Örnek olarak bir motorun yapısı ve çalışması anlatılırken gerçek bir motor temininin pahalı ve güç olmasına karşılık, bilgisayar modelinde motorun dışı saydamlaştırılarak iç yapısı görüntülenebilir, sadece belirli parçaları gösterilebilir veya rahatlıkla kesiti alınabilir. Eğitimde bilgisayar ortamının tüm avantajlarından yararlanılması sağlanabilir.

Yabancı dil öğreniminde sesli animasyonlardan ve etkileşimli CD'lerden yararlanılmaktadır.

3.2.4 Eğlence

Bilgisayar grafikleri eğlence dünyasının ayrılmaz bir parçası haline gelmiştir. Film endüstrisinde, televizyon şovlarında, müzik videolarında ve bilgisayar oyunlarında hareketli görüntülerin oluşturulmasında bilgisayar grafikleri kullanılmaktadır. Çizgi film ve bilim-kurgu filmlerinin yapımında (özellikle uzay gemileri, yaratıklar, robotlar, sanal gezegenler, şehirler) veya filmlerdeki efektlerin hazırlanmasında bilgisayarlı animasyonlar son on yılda etkili bir şekilde kullanılmıştır. Jurassic Park, Total Recall, Robocop, Terminator 2, Mask, Toy Story bunların ilk örnekleridir. Titanic, KarıncaZ, Star Wars gibi filmler de yakın tarihte bilgisayarlı animasyon kullanılarak hazırlanan önemli filmlere örnek olarak verilebilir. Görsel efektler (VFX) yine de izleyicinin aldatmacanın farkına varmaması için kısa tutulmaktadır.

Tümüyle bilgisayarda yapılmış ilk çizgi film olan “Oyuncak Hikayesi” (Toy Story), 1995'te Amerika'da gösterime girmiştir. O tarihe kadar bütün filmlerinde çizgi ustalarını kullanan Disney'in bilgisayarla yaptırdığı bu ilk çizgi film bilgisayarda bu şekilde hazırlanmış çizgi filmlerin öncüsü olmuştur.

Üç boyutlu grafiklerin en eğlenceli kullanım alanlarından birisi de oyunlardır. Her tür oyunda bilgisayar grafikleri ve animasyonları önemli yer tutmaktadır. Hareketli bilgisayar oyunlarında DOOM türü motorların ortaya çıkması önemli bir adım olmuştur. Oyuncuların özgürce dolaşabildiği ortamlar oluşturulmuş ve 360 derecede her yöne bakılabilmesi sağlanmıştır.

3.2.5 Reklamcılık

Reklam logolarında ve diğer tasarımlarda üç boyutlu metin ve grafikler yoğun olarak kullanılmaktadır. Televizyonlarda özellikle ürün reklamlarında üç boyutlu grafik ve animasyonların önemi oldukça fazladır. Film, dizi ve belgesellerin özellikle jeneriklerde bilgisayar grafiklerinden yararlanılmaktadır. Örnek olarak reklamlarda kola emen bir çocuğun şişenin içine girmesi, bir arabanın giderken koşan bir kaplana dönüşmesi, otomobillerin dans etmesi gibi animasyon öğelerine yer veren reklamlar hepimizin belleklerinde yer etmiştir.

3.2.6 Sanat

Bilgisayar grafikleri güzel sanatlarda da sanatçılar tarafından kullanılmaktadır. İki boyutlu resimlerin bilgisayar yardımıyla çizilmesinin pek çok üstünlükleri vardır. Örnek olarak bir evin tüm duvarları bir anda 16 milyondan çok daha fazla renk içeren paletten alınan bir renge boyanabilir. Bu rengi standart sanat malzemesi ile elde etmek zor olabilir. İlgili tüm alanları duyarlı bir şekilde boyamak uzun sürer ve yeni bir renk vermek olanaksızlaşabilir. Bilgisayar, sanatçılar için yeni bir araç olmakla kalmamış, sanata yeni boyutlar da getirmiştir. Örnek olarak başka bir araç ile kolaylıkla elde edilemeyecek olan üç boyutlu resimler bilgisayar yardımı ile kolaylıkla çizilebilir, animasyon yardımı ile hareket ettirilir, döndürülür, ölçeklendirilir ya da üzerinde başka bir işlem uygulanabilir. İki farklı resim arasındaki geçişin tüm aşamaları bilgisayar aracılığı ile oluşturulabilir. Ayrıca matematiksel formüllerin üç boyutlu bilgisayar görüntüleri de sanatta kullanılmaya başlamıştır. Üç boyutlu boyama, yüzey kaplama gibi yöntemleri uygulayan yazılımların ortaya çıkması da bilgisayar grafiklerinin sanata yenilikler getirmesini sağlamıştır.

3.2.7 Sanal Gerçeklik ve Güçlendirilmiş Gerçeklik

Sanal gerçeklik (virtual reality), bilgisayarda oluşturulan üç boyutlu sanal dünya ve bu dünyadaki gerçeklik hissidir. Kullanıcının özel başlık ve fiber optik eldiven gibi donanımları giyerek bu dünyaya girmesi, içinde hareket etmesi, nesnelerle etkileşimi, sanal olarak oluşturulan bu dünya gerçekte varmış duygusu uyandırır. Güçlendirilmiş gerçeklik teknolojisi ise bir anlamda sanal gerçekliğin tersine döndürülmüş halidir. İnsanların içinde yaşadıkları fiziksel dünyada bilgisayar destekli ortamların oluşturulmasını sağlar. Sanal gerçeklik ve güçlendirilmiş gerçeklik teknolojilerinin kullanıldığı alanlar çok geniştir. Beyin ameliyatlarının uzaktan denetiminden, sinema filmlerine ve bilgisayar oyunlarına kadar birçok alanda sanal ve güçlendirilmiş gerçeklikten yararlanılmaktadır.

3.2.8 Web

İnternette web sayfalarındaki (home page) basit düğmelerden, üç boyutlu animasyonlara kadar bilgisayar grafiklerinden yararlanılması sayfaların daha dikkat çekici hale gelmesini sağlamaktadır. Web sayfalarının tasarımında her geçen gün daha fazla kişi daha kaliteli grafikler ve animasyonlar hazırlamak için Photoshop, Paint Shop Pro gibi grafik uygulama yazılımlarını kullanmaktadırlar. Bu şekilde bilgisayar grafiklerinin kullanım alanlarına bir yenisi daha eklenmiştir. Ayrıca elektronik ticaret konusunda da üç boyutlu grafiklerin etkisi büyüktür. Elektronik ticaret sitelerinde üç boyut genelde şu üç formdan birisi şeklinde kullanılmaktadır : Ürün tasarımcıları, ürünlerinin üç boyutlu modellerini internete yerleştirmektedirler (örnek olarak www.styleclick.com'da elbise, ayakkabı ve aksesuarlara herhangi bir açıdan bakılabilmektedir); Casio, Yamaha, Mercedes, Nokia, Lego gibi birçok büyük ve tanınmış şirket, ürünlerinin üç boyutlu modellerini

internet üzerinden sunmaktadırlar. Sayfalarını ilgi çekici hale getirmek isteyenler üç boyutlu çizgi film karakterleri ve “banner”lar kullanmaktadırlar; Tasarımcılar bir şehrin, inşa edilecek bir iş merkezinin modelini önceden üç boyutlu olarak internette gezilebilecek şekilde sunmaktadırlar. 3B oyun motorlarının emlak yazılımlarına uyarlanması ile ev arayanlar internet üzerinden evleri gezebileceklerdir.

İnternet üzerine üç boyutlu içerik ekleme konusunda son günlerde üzerinde yoğun olarak çalışmalar yapılmaktadır. Günümüzde 30’den fazla şirket internet üzerine üç boyutlu içerik eklemeyi sağlayan yazılımlar geliştirmişlerdir. Bu “Web3D” şirketleri ağırlıklı olarak elektronik ticaret ve eğlence alanına yönelik olarak çalışmalar yapmaktadırlar. Birçok firma, bu yazılımları kullanarak ürünlerinin üç boyutlu modellerini internet üzerine yerleştirmektedirler. Web3D şirketleri Netscape ve Internet Explorer gibi popüler web tarayıcılarına da eklenebilen görüntüleyicilerini ücretsiz olarak vermektedirler. Bu sayede müşteriler, rahatlıkla ticari şirketlerin ürünlerini, internet üzerinden erişerek üç boyutlu olarak inceleyebilmektedirler. Web3D şirketlerinin beklentisi bu alanda standartların geliştirilmesidir. Bu şirketlerden bazıları şu şekilde belirtilebilir :

(Ağırlıklı olarak e-ticaret alanındakiler)

- ViewPoint (www.metastream.com)
- Cycore (www.cycore.com)
- RichFX (www.richfx.com)

(Ağırlıklı olarak eğlence alanındakiler)

- Pulse Entertainment (www.pulse3d.com)
- Brilliant Digital Entertainment (www.brilliantdigital.com)
- Eyematic (www.eyematic.com)

3.3 Bilgisayar Grafiklerinin Geleceđi

İkibinli yılların başlarında bilgisayar grafikleri alanında řu gelişmelerin yaşanması beklenebilir :

- Üç boyutlu bilgisayar grafikleri bilgisayar yazılımlarında ve gerçek yaşamda ağırlığını daha fazla hissettirecektir. Kelime işlemcilerden internetteki web sayfalarına kadar birçok yerde üç boyutlu bilgisayar grafikleri ortamın doğal bir parçası haline gelecektir. Sonuçta üç boyutlu bilgisayar grafiklerinin kullanım alanları genişleyecek ve daha etkin kullanımı gerçekleşecektir.
- Üç boyutlu bilgisayar grafikleri içeren uygulama veya yazılımları geliştirmek isteyen kişilerin çok iyi matematik altyapısına sahip olması veya uzun programlar yazması gerekmeyecektir. Deneyimsiz programcılardan profesyonellere kadar her düzeydeki insan üç boyutlu sanal dünyalar oluşturup bunlar üzerinde işlemler yapabilecektir. Bunun için sadece kişisel bilgisayara sahip olmak yetecektir. Sonuçta daha çok kişi üç boyutlu grafikleri anlayacak ve kullanacaktır.
- Üç boyutlu nesnelere ilişkin çok geniş, üç boyutlu bilgi kütüphaneleri oluşturulacak ve bunların yeniden kullanımı ile çeşitlilik sağlanacaktır.
- Yeni grafik donanım ve çevre aygıtları bulunacaktır. Örnek olarak üç boyutlu nesneleri bilgisayara yükleyebilmek için hazırlanan üç boyutlu tarayıcılar geliştirilecektir.
- Grafik yazılım ve donanımlarındaki gelişmeler üç boyutlu animasyonlardaki kaliteyi yani gerçekçiliđi artırmaya devam edecektir.

- Bilgisayar grafiklerindeki gelişmeler kullanıldığı alanlarda da yeni ufuklar açmaya devam edecektir ve daha kaliteli ürünlerin ortaya çıkmasını sağlayacaktır.
- Bilgisayar grafikleri ile ilgili derslerde alt düzey programlamanın yerini yüksek düzeyli API'lerin alması ile dersler teori ve birkaç basit örnekle sınırlı kalmayacak ve daha ileri konular ders programlarına katılarak gerçek yaşamda kullanılabilecek projeler yapılabilir.
- Bilgisayar grafiklerinin yapay zeka ve robotbilim gibi alanlarla etkileşimi sonucunda örnek olarak ameliyat yapan robotlar üretilebilecektir.



4. GEOMETRİK PROBLEMLERİN ÇÖZÜMÜNDE KULLANILAN ALGORİTMİK YÖNTEMLER

4.1 Genel Bilgiler

Bu bölümde kısaca algoritma analizi ve veri yapılarına değinilecek, bilişimsel geometri alanında ortaya çıkan temel bazı problemler tanımlanacak ve çözümlerinde kullanılan pratik ve basit çözüm yolları sunulacaktır.

Bilişimsel geometri, problemlerin çözümünde belirli algoritmik paradigmaları kullanır. Bu paradigmalardan en önemlileri aşağıda belirtilmiştir : (Laszlo, 1996)

- Birer Birer Ekleme (Incremental Insertion) Yöntemleri
- Birer Birer Seçme (Incremental Selection) Yöntemleri
- Düzlem Tarama Algoritmaları (Plane-Sweep Algorithms)
- Böl ve Çöz Algoritmaları (Divide and Conquer Algorithms)
- Uzaysal Bölümleme (Spatial Subdivision) Yöntemleri

Bir problem değişik paradigmalardaki algoritmalar kullanılarak farklı yollarla çözülebilir. Bu paradigmalara uyan algoritmalar ve farklı veya benzer problemlerin çözümündeki kullanım şekilleri bu bölümde incelenmiştir.

4.2 Problem, Veri, Algoritma, Veri Yapıları

Bir **problem**, kabul görmüş bir dizi komut kullanan bir yöntemle çözülebilir. Bir bilişimsel geometri problemi, bir problem deyimi ile belirtilir. Bu, hem tüm geçerli girdileri belirtir, hem de geçerli girdilerin bir fonksiyonu olarak çıktıyı gösterir. Örnek olarak, SINIR-KESİŞİMİ olarak isimlendirdiğimiz problemi düşünelim : (Laszlo, 1996)

Problem Deyimi, “düzlemdeki iki çokgenin sınırlarının kesişip kesişmediğinin bulunması”dır. Geçerli bir girdi, belirlenmiş biçimdeki iki çokgenden oluşur. Verilen iki geçerli girdi çokgeni için, sınırları kesişiyorsa çıktı “evet” tir, değilse “hayır” dır (Laszlo, 1996).

Bir **algoritma**, bir problemin çözümünde kullanılan komutlar dizisidir. Bir problemi çözmek için geliştirilmiş kesin bir yöntemdir. Algoritmalar soyuttur. Verilen bir algoritma, pek çok şekilde gösterilebilir; farklı programlama dillerinde (Pascal, C, Java gibi) bir bilgisayar programı olarak gerçekleştirilebilir ve farklı bilgisayarlarda çalıştırılabilir. Her bilgisayar programı bir algoritmayı canlandırır ve program iyi düzenlendi ise altında yatan algoritma, programdan anlaşılabilir. Böyle bir algoritma örneği olarak şu algoritma SINIR-KESİŞİMİ problemini çözer : Verilen iki çokgen için, (Laszlo, 1996)

- İlk çokgenin her kenarını, ikinci çokgenin her kenarı ile karşılaştır.
- En az bir çift kenar kesişiyorsa, sınırların kesişmekte olduğunu belirt;
- diğer durumda ise, sınırların kesişmediğini belirt.

Dikkat edilecek olursa iki kenarın kesişip kesişmediğini belirlemek için, diğer bir algoritma gereklidir.

Bir algoritmik paradigma, bir algoritmanın deseninin tasarımıdır. Aynı algoritmaya dayanan farklı programlar olabildiği gibi, aynı paradigmaya dayanan farklı algoritmalar da olabilir. Paradigmaları bilmek algoritmaların anlaşılmasında, açıklanmasında ve yeni algoritmaların tasarımında yararlı olacaktır. Daha önce benzeri ile karşılaşılan bir algoritmanın geliştirilmesi gerçekten daha kolaydır. Taslak olarak belirtilen SINIR_KESİŞİMİ problemini çözen algoritma birer birer ekleme paradigması içinde yer alır. Her aşamada ilk çokgenin

bir kenarı seçilerek ikinci çokgenin her kenarı ile karşılaştırılmaktadır (Laszlo, 1996).

Veriler, algoritmalar tarafından işlenen en temel elemanlardır (sayısal bilgiler, metinsel bilgiler, resimler, sesler ve girdi/çıkı olarak veya ara hesaplamalarda kullanılan diğer bilgiler). Bir algoritmanın etkin, anlaşılır ve doğru olabilmesi için algoritmanın işleyeceği verilerin düzenlenmesi gerekir.

Algoritmalar, **veri yapıları** denilen ve veriyi düzenleyen nesneleri içerirler. Algoritmalar ve veri yapıları birlikte düşünülürler : Veri yapıları, kullanıldıkları algoritmaların verilerinin düzenlenme biçimini belirler ve yapıtaşlarını oluşturur. Veri yapılarının iyi tasarlanması etkin algoritmalarının oluşturulmasını kolaylaştıracaktır. **SINIR_KESİŞİMİ** algoritması, çokgenleri ve kenarlarını temsil eden veri yapılarını içerir. Tamsayılar (integer), gerçel sayılar (float,double), karakterler (character) gibi temel veri tipleri ve diziler (array), yapılar (structure) gibi bileşik veri tipleri dışında listeler (lists), yığıt (stack), kuyruklar (queues), ikili arama ağaçları (binary search tree) gibi birçok temel veri yapıları problemlerin çözümünde etkin olarak kullanılmaktadır. Bunlar yeterince basit ve problemlerin çözülebilmesini sağlayabilecek kadar güçlüdür. Bunlar dışında bu veri yapılarının geliştirilmiş şekilleri ve daha karmaşık veri yapıları da kullanılabilmektedir. Her bir veri yapısı problemin ve önerilen çözüm yolunun doğasına uygun olarak belirlenir ve her birinin kendine göre avantaj ve dezavantajları vardır.

4.3 Algoritmaların Karşılaştırılması ve Algoritma Analizi

Bir programın zaman karmaşıklığı (time complexity) programın işletim süresidir. Bir programın yer karmaşıklığı (space complexity) program işletildiği sürece gerekli olan yer yani bellek miktarıdır. Bir problemin çözümünde kullanılabilecek olan algoritmalarından en etkin

olanı seçilmelidir. En kısa sürede çözüme ulaşan veya en az işlem yapan algoritma tercih edilmelidir. Burada, bilgisayarın yaptığı iş önemlidir. Uçuş kontrolü ve robotların görmesindeki gibi gerçek zamanlı sistemlerde bozulmanın meydana gelmemesi için işlemlerin göz açıp kapayıncaya kadar yapılması gereklidir. Metin editörleri gibi daha toleranslı sistemlerin kullanıcının komutlarına birkaç saniye içinde yanıt verebilmesi yeterlidir. Bordro programlarının çalışmasının ve tıbbi görüntülerinin analizlerinin birkaç saat veya birkaç gün içinde tamamlanması beklenir. Bazen işletim zamanının daha hızlı bilgisayarların geliştirilmesinden daha az önemli olduğu düşünülür. Bu doğru değildir. Günümüzde bilgisayarlara aktarılan problemler hiç olmadığı kadar zorlaşmıştır ve zorluk dereceleri çok daha hızlı çözümler üretilmesini gerektirecek kadar bilgisayar teknolojisinin ilerisindedir. Yer, zamana göre daha az önemlidir. Zaman genelde en kritik kaynak olmakla birlikte eldeki kaynak durumuna göre bazı durumlarda da en az bellek harcayan algoritmanın tercih edilmesi gerekebilir. Ayrıca, programcının yaptığı iş açısından veya algoritmaların anlaşılabilirlikleri bakımından da algoritmalar karşılaştırılabilir. Daha kısa sürede biten bir algoritma yazmak için daha çok kod yazmak veya daha çok bellek kullanmak gerekebilir (trade-off).

Birçok problemde çözüme ulaştıran birçok rakip algoritma ve veri yapıları vardır. Rakip algoritmaları yaptıkları iş açısından karşılaştırmak için her algoritmaya uygulanabilecek somut ölçütler tanımlanmalıdır. Aynı işi yapan algoritmalarından daha az işlemde sonuca ulaşan (hızlı olanın) belirlenmesi yani daha genel olarak algoritma analizi teorik bilgisayar bilimlerinin önemli bir alanıdır. Algoritma analizi, verilen bir problemin daha etkin bir çözümünün olup olmadığının belirlenmesinde ve en iyi çözümün seçilmesinde kullanılır.

Yazılımcılar, iki farklı algoritmanın yaptıkları işi nasıl ölçüp karşılaştırırlar? İlk çözüm, algoritmaları bir programlama dilinde kodlayıp her iki programı da çalıştırarak işletim sürelerini karşılaştırmaktır. Ancak, işletim süresi kısa olanın daha iyi algoritma olduğu söylenebilir mi? Bu yöntemde işletim süreleri belirli bir bilgisayara özeldir. Dolayısıyla işletim süresi de bu bilgisayara bağlıdır. Daha genel bir ölçüm yapabilmek için olası tüm bilgisayarlar üzerinde algoritmanın çalıştırılması gerekir.

İkinci çözüm, işletilen komut ve deyimlerin sayısını bulmaktır. Fakat bu ölçüm, kullanılan programlama diline ve programcılarının stiline göre değişim gösterir. Bunun yerine kritik algoritmik geçişlerin sayısı hesaplanabilir. Her tekrar için sabit bir iş yapılıyor ve sabit bir süre geçiyorsa, bu ölçü anlamlı hale gelir.

Buradan, algoritmanın temelinde yatan bir işlemi ayırarak, bu işlemin kaç kere tekrarlandığını bulma düşüncesi doğmuştur. Örnek olarak bir tamsayı dizisindeki tüm elemanların toplamını hesaplama işleminde gerekli olan iş miktarını ölçmek için tamsayı toplama işlemlerinin sayısı bulunabilir. 100 elemanlı bir dizideki elemanların toplamını bulmak için 99 toplama işlemi yapmak gerekir. Buradan hareketle n elemanlı bir listedeki elemanların toplamını bulmak için $n-1$ toplama işlemi yapmak gerekir şeklinde genelleştirme yapabiliriz. Böylece algoritmaları karşılaştırırken belirli bir dizi boyutu ile sınırlı kalınmaz.

İki gerçel matrisin çarpımında kullanılan algoritmaların karşılaştırılması istendiğinde, matris çarpımı için gereken gerçel sayı çarpma ve toplama işlemlerinin karışımı bir ölçü olacaktır. Bu örnekten ilginç bir sonuca ulaşılır : Bazı işlemlerin ağırlığı diğerlerine göre fazladır. Birçok bilgisayarda bilgisayar zamanı cinsinden gerçel sayı çarpımı gerçel sayı toplamından çok daha uzun sürer. Dolayısıyla tüm

matris çarpımı düşünüldüğünde toplama işlemlerinin etkinlik üzerindeki etkisi az olacağından ihmal edilebilir ve sadece çarpma işlemlerinin sayısı dikkate alınabilir. Algoritma analizinde genelde algoritmada egemen olan bir işlem bulunur ve bu işlem diğerlerini gürültü (noise) düzeyine indirger.

4.3.1 İşletim Zamanı (Running Time)

İşletim zamanını girdi boyutunun bir fonksiyonu olarak ele almak tüm geçerli girdileri tek değere indirir. Bu da değişik algoritmaları karşılaştırmayı kolaylaştırır. En yaygın karmaşıklık ölçütleri en kötü durum işletim süresi (Worst-Case Running Time) ve ortalama durum işletim süresidir (Average-Case Running Time) (Laszlo, 1996).

En kötü durum işletim süresi : Bu işletim süresi, her girdi boyutundaki herhangi bir girdi için en uzun işletim süresini tanımlar. Örnek olarak bir programın en kötü olasılıkla ne kadar süreceğinin tahmin edilmesi istenen bir durumdur. Örneğin n elemanlı bir listede sıralı arama en kötü olasılıkla (aranan en son eleman ise ya da bulunamazsa) n karşılaştırma gerektirecektir. Yani en kötü durum işletim zamanı $T(n)=n$ 'dir. Tüm problemlerde sadece en kötü girdi dikkate alındığı durumlarda bu işletim zamanının değerini hesaplamak göreceli olarak kolaydır (Laszlo, 1996).

Ortalama durum işletim süresi : Bu işletim süresi her girdi boyutundaki tüm girdilerin ortalamasıdır. Örneğin n elemanın her birinin aranma olasılığının eşit olduğu varsayıldığında ve liste dışından bir elemanın aranmayacağı varsayıldığında ortalama işletim süresi $(n+1)/2$ 'dir. İkinci varsayım kaldırıldığında ortalama işletim süresi $[(n+1)/2, n]$ aralığındadır (aranan elemanların listede olma eğilimine bağlı olarak). Ortalama durum analizi basit varsayımlar yapıldığında bile

zordur ve varsayımlar da gerçek performansın iyi tahminlenememesine neden olabilir (Laszlo, 1996).

4.3.2 Asimptotik Analiz

Algoritmaların karşılaştırılmasında asimptotik etkinlikleri (girdi boyutu sonsuza yaklaşırken işletim süresinin artışı) de dikkate alınabilir. Asimptotik gösterimin 4 önemli elemanı olan 4 önemli gösterim vardır : O-notation, o-notation, Ω -notation ve θ -notation. Burada sadece O gösterimi üzerinde durulacaktır. O gösterimi, fonksiyonların artış oranının üst sınırını belirler. $O(f(n))$, $f(n)$ fonksiyonundan daha hızlı artmayan fonksiyonlar kümesini gösterir.

n elemanlı bir listedeki elemanların toplamını bulmak için $n-1$ toplama işlemi yapmak gerekir şeklinde bir genelleştirme yapılabilmektedir. Burada yapılan işi, girdi boyutunun bir fonksiyonu olarak ele almış olmaktadır. Bu fonksiyon yaklaşımını matematiksel gösterim kullanarak Big-O gösterimi veya büyüklük derecesi (order of magnitude) ile ifade edebiliriz. Büyüklük derecesini problemin boyutuna bağlı olarak fonksiyonda en hızlı artış gösteren terim belirler. Örnek olarak : $f(n) = n^4 + 100n^2 + 10n + 50 = O(n^4)$ fonksiyonunda n 'in derecesi 4'tür yani n 'in büyük değerleri için fonksiyonu en fazla n^4 etkiler. n 'in büyük değerleri için n^4 , $100n^2$ 'den, $10n$ 'den ve 50 'den çok büyük olacağından daha düşük dereceli terimler dikkate alınmayabilir. Bu, diğer terimlerin işlem süresini etkilemedikleri anlamına gelmez; bu yaklaşım yapıldığında n 'in çok büyük değerlerinde önem taşımadıkları anlamına gelir. Buradaki n , problemin boyutudur ve yığıt, liste, kuyruk, ağaç gibi veri yapılarında eleman sayılarıdır (n elemanlı bir dizi gibi).

Yaygın olarak kullanılan bazı artış fonksiyonları $O(1)$ "sabit zaman", $O(\log_2 n)$ "logaritmik zaman", $O(n)$ "doğrusal zaman", $O(n \log_2 n)$, $O(n^2)$ "ikinci dereceli zaman", $O(n^3)$ "kübik zaman" ve $O(2^n)$

“üstel zaman” ‘dır. Bu fonksiyonlarla ilgili ayrıntılı bilgi Algoritma, Veri Yapıları ve Algoritma Analizi kitaplarından alınabilir (Langsam ve ark., 1996) (Main ve Savitch, 1995) (Sahni, 1998) (Weiss,1996) (Weiss,1999).

4.4 Birer Birer Ekleme (Incremental Insertion) Yöntemi

Birer birer ekleme adı verilen algoritmik tasarım yaklaşımı, o ana kadar eklenen tüm elemanları içeren etkin çözümün üzerine, probleme bir eleman daha eklenmesi temeline dayanır. Her eklenme sırasında, bir sonraki girdi elemanı gözden geçirilerek işlenir ve geçerli çözüm, yeni elemanı da içine alacak şekilde güncellenir. Tüm girdiler işlendiğinde, problemin tümü de çözülmüş olur (Laszlo, 1996).

Bu yöntem, gizemli bir hikaye kitabının okunması olayına benzetilebilir. Okuyucu kitap boyunca, cinayeti kimin, neden ve nasıl işlediğine ilişkin varsayımlarını oluşturacaktır. Her yeni ipucu, oluşturduğu son varsayımı destekleyecek veya yeniden gözden geçirilmesine neden olabilecektir. Bazı durumlarda elde edilen ipuçları, son varsayımı geçersiz kılıp yeni bir varsayım yapmayı da gerektirebilecektir. Kitabın sonuna gelinip de tüm ipuçlarının ortaya çıkması ile, yeterince zeki olan bir okuyucu cinayeti çözmüş olacaktır (Laszlo, 1996).

Bazı durumlarda algoritma, etkin çözümü değil, etkin durumu belirtiyor olabilir. Çünkü o ana kadar görülen girdiler, tutarlı bir çözümü gösterebilecek derecede yeterli olmayabilir. Bu durum ile özellikle çokgenleri içeren problemlerin çözümünde karşılaşılır : Çokgen sınırları, bir anda bir köşe noktası işlenecek şekilde inceleniyorsa, tüm girdi işleninceye kadar elde basit bir çokgen olduğu bile söylenemez. Gizemli hikaye kitabı benzetmesine dönülecek olursa bu olay, okuyucunun cinayet işlenmesinden önce tahminlemelerde bulunmasına benzetilebilir.

Ortada çözülecek bir problem olmamasına rağmen ilk ipuçları ve sezgiler, sonradan kullanılmak üzere düzenlenir (Laszlo 1996).

Bir dizi içindeki en küçük tamsayıyı bulmak için kullanılan en yaygın yaklaşım –o ana kadar görülen en küçük tamsayının izini tutarak, dizi içinde sondan başa doğru ilerlemek– birer birer ekleme yaklaşımına örnek olarak gösterilebilir. Ekleme, o anki en küçük tamsayıyı tutan değişkene, şartlı atama yapılarak gerçekleştirilir. Her aşamada bu değişken, o ana kadar işlenmiş tüm sayıları dikkate alacak şekilde problemin çözümünü içerir. Birer birer ekleme, genelde bu kadar basit değildir (Laszlo, 1996).

Ekleme sıralama “insertion sort” adı verilen ve az elemanlı listelerin sıralanmasında çok yararlı olan sıralama yöntemi, birer birer ekleme yöntemine dayanmaktadır. Geometrik problemlerin çözümü ile ilgili olarak da bu yöntemi kullanan bir dizi ilginç algoritma vardır:

- Sonlu noktalar kümesi içinde bir yıldız-şekilli çokgenin bulunması
- Noktalar kümesinden dışbükey kabuğun (convex hull) bulunması
- Verilen bir noktanın bir çokgen içinde olup olmadığının bulunması
- Çokgenlerin üçgenlere ayrılması

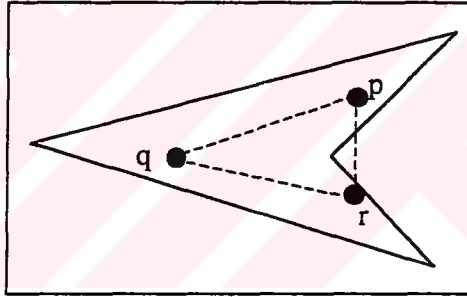
Geometrik problemlerin çözümünde birer birer ekleme yöntemlerinin nasıl kullanıldığı konusuna ışık tutmak üzere bazı algoritmalar aşağıda kısaca açıklanmıştır.

4.4.1 Yıldız-Şekilli Çokgenlerin Bulunması

Düzlemdeki sonlu noktalar kümesinin elemanlarının birbirine bağlanması ile kenarlar ve bir çokgen oluşturulabilir. Buna, noktalar

kümesinin çokgenleştirilmesi adı verilir. Bu alt bölümde yıldız-şekilli çokgenler oluşturulması için kullanılan bir yöntem anlatılacaktır. Bu yöntem, yıldız-şekilli çokgen oluşturmak için noktaları birbirine bağlar (Laszlo, 1996).

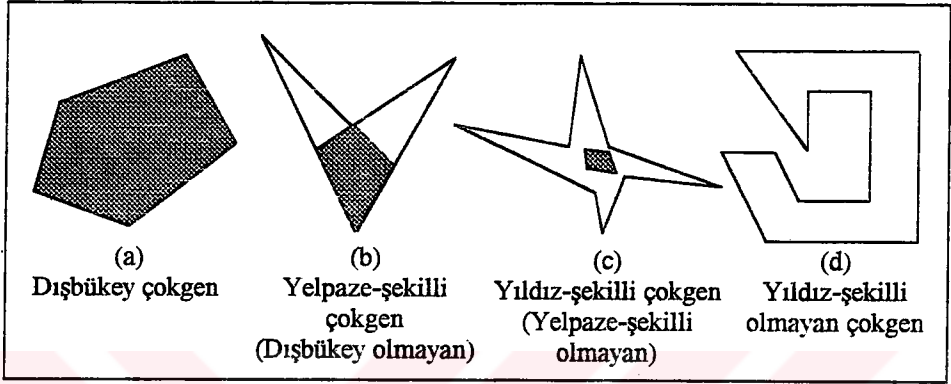
Aynı çokgen içinde p ve q noktaları olsun. $\overline{pq}$ doğru parçası, aynı çokgen içinde ise p , q 'yu görür deriz. Çokgenin sınırları ışık geçirmez duvarlar olarak ve içi de hava gibi saydam bir ortam olarak varsayıldığında bir nokta diğerini, ancak arada duvar yoksa görebilir. Görme simetriktir (p , q 'yu görürse, q da p 'yi görür), fakat geçişli değildir (p , q 'yu görürse, q da r 'yi görürse, p de r 'yi görür denilemez). Bu konuya ilişkin bir örnek Şekil 4.1'de görülmektedir.



Şekil 4.1 : Noktaların birbirini görmesi. p ve q noktaları birbirini görürken, q ve r noktaları birbirini görürken, p ve r noktaları birbirini görememektedir.

Çokgen içindeki her noktayı gören noktalar kümesi, çokgenin özü (kernel) olarak adlandırılır. Özü boş olmayan bir çokgen, yıldız-şekilli çokgen olarak adlandırılır (Şekil 4.2). Boş olmayan ve özü, bir veya daha fazla köşe noktası içeren çokgen ise, yelpaze-şekilli (fan) çokgen olarak adlandırılır. Her dışbükey çokgen (convex polygon) özü köşe noktası içerdiği için aynı zamanda yelpaze-şekilli çokgenler sınıfına girer. Zaten dışbükey çokgenlerin her köşe noktası, çokgenin her noktasını

görmektedir. Yelpaze-şekilli her çokgen, özü boş olmadığı için aynı zamanda yıldız-şekillidir (Laszlo, 1996).



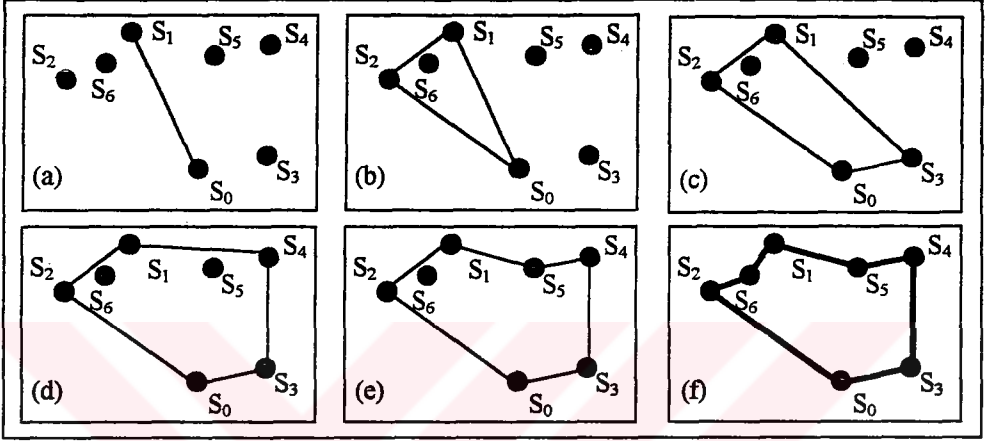
Şekil 4.2 : Özleri koyu renkli olarak belirtilmiş değişik çokgenler

Düzlemde, $s_0, s_1, \dots, s_{n-1}$ noktalarından oluşan S noktalar kümesi verilmiş olsun. Buna göre, S noktalar kümesinden, bir yıldız çokgen oluşturma problemini inceleyelim. Bu tür çokgenlerden birden fazla var olabileceğini anlamak zor değildir. Burada s_0 (ilk nokta) noktasının öz olduğu bir çokgen bulunacaktır (Laszlo, 1996).

Algoritma, S noktalar kümesinden anlık çokgeni oluştururken, yinelemeli olarak işlemektedir. Başlangıçta, anlık çokgen bir noktadan (s_0) oluşmaktadır. 1'den $n-1$ 'e kadar olan her yinelemede (i), bir sonraki nokta (s_i), anlık çokgene eklenmektedir. Yinelemeler tamamlandığında anlık çokgen, aradığımız yıldız şekilli çokgen olacaktır (Laszlo, 1996).

Anlık çokgene her yeni s_i noktasını eklemek için, s_0 noktasından başlanarak, saatin tersi yönünde dolaşılır. Dolaşma çokgen sınırları üzerinde, saatin tersi yönünde bir sonraki s_i bulunana kadar devam eder; s_i , bu köşeden daha önce eklenir. Dolaşım s_0 'a dönecek şekilde bir daireye tamamlanırsa s_i , s_0 'dan önce eklenir. Burada s_0 , nöbetçi (sentinel)

olarak görev yapar ve daha fazla dolaşımı engellemeyi garantiler. Şekil 4.3 küçük bir problem üzerinde çalışan bu algoritmanın fotoğraflarını göstermektedir.

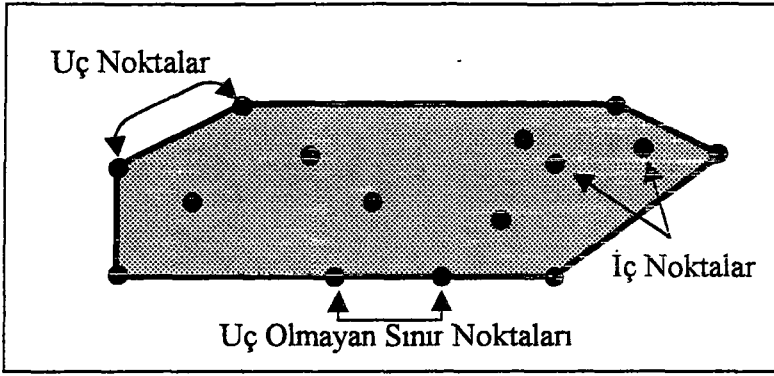


Şekil 4.3 : Bir noktalar kümesinden yıldız-şekilli bir çokgen oluşturulması

4.4.2 Dışbükey Kabukların (Convex Hulls) Bulunmasında Eklemeli Kabuk Yöntemi

S , bir düzlemdeki sonlu noktalar kümesi olsun. S kümesinin dışbükey kabuğu $CH(S)$ olarak gösterilir ve bu kabuk S 'yi içeren tüm dışbükey çokgenlerin kesişimidir. Bir başka deyişle dışbükey kabuk, S kümesinin tüm noktalarını kapsayan en küçük alanlı dışbükey çokgendir. S kümesinin noktaları tarafından oluşturulabilecek tüm üçgenlerin bileşimi olarak da ifade edilebilir (Laszlo, 1996).

Bu konuyu gözümüzde canlandırabilmek için bir tahta plaka üzerinde S kümesinin her noktasından çıkan çiviler olduğu düşünülebilir. Tüm çiviler etrafına lastik şerit gerilip bırakıldığında çivilerin etrafında dışbükey kabuk oluşturacak şekilde gerili kalır (Şekil 4.4).



Şekil 4.4 : Sonlu noktalar kümesi ve dışbükey kabuğu

Dışbükey kabuklar geometrik uygulamalarda geniş bir kullanım alanı bulmaktadır. Örüntü tanımda bilinmeyen bir şekil dışbükey kabuğu ile tanımlanabilir; hareket planlama alanındaki çalışmalarda hareketli bir robotun engellerle dolu bir bölgeye girdiğinde engellerin dışbükey kabuğunu bulması gidebileceği bir yol bulmasını kolaylaştırır (Laszlo, 1996).

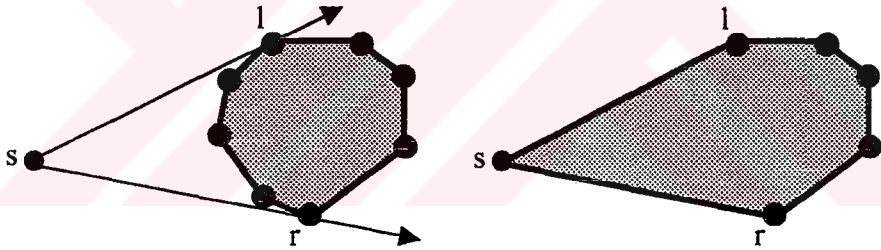
Dışbükey kabuğun sınırlarındaki noktalara *sınır noktaları* adı verilir. Dışbükey kabuğun köşe noktaları olan sınır noktalarına *uç noktalar* adı verilir. Dışbükey kabuğun köşelerini oluşturmayıp kenarlarında olan noktalara *uç olmayan sınır noktaları* veya kenar noktaları adı verilir. *İç noktalar* ise dışbükey kabuk alanı içinde olan noktalardır (Laszlo, 1996).

Birer birer ekleme yöntemine göre S sonlu noktalar kümesinin dışbükey kabuğunun bulunması o ana kadar eklenmiş noktalara göre oluşturulmuş anlık dışbükey kabuk korunurken sisteme yeni bir nokta eklenmesine dayanır. Anlık dışbükey kabuğa etkin dışbükey kabuk da denilmektedir. Başlangıçta etkin kabuk bir tek nokta içermekteyken tüm

noktalar eklendiğinde tüm noktaları gözönüne alan gerçek dışbükey kabuk $CH(S)$ bulunmuş olur (Laszlo, 1996).

Dışbükey kabuğa yeni bir s noktası eklendiğinde iki durumdan birisi oluşur :

- s noktası etkin kabuğun içindedir (sınırında veya içinde). Bu yeni nokta etkin dışbükey kabuğun yenilenerek değiştirilmesini gerektirecek bir durum oluşturmaz.
- s noktası etkin kabuğun dışındadır. Bu durum etkin kabuğun yeni nokta gözönüne alınarak değiştirilmesini gerektirir (Şekil 4.5). s noktasından etkin kabuğa teğet iki tane destek çizgisi çizilir. s noktası ve teğet noktalarındaki köşeler (l,r) arasında kalan ardışık noktaların oluşturduğu zincir çıkartılarak s noktası etkin kabuğa eklenir.



Şekil 4.5 : Eklmeli kabuk yönteminde etkin kabuğa s noktasının eklenmesi

4.5 Birer Birer Seçme (Incremental Selection) Yöntemleri

Birer birer seçme yöntemleri problemleri azar azar çözerek ilerler. Bu yöntemler girdinin geliş sırasına göre değil kendi belirlediği sıraya göre girdiyi işler. Mantığı, girdiyi tarayarak bir sonra işlenecek en iyi elemanı bulmaktır (Laszlo, 1996).

Bu yöntemin bazı uygulamalarında elemanların işlenme sırası yöntem ilerledikçe belirlenir. Bazı durumlarda girdiler önceden sıralanır. Diğer uygulamalarda sıralama önceden yapılmaz, sonra gelecek elemanın seçimi o ana kadar gelmiş olan veriler içinden yapılır (Laszlo, 1996).

Birer birer seçme yöntemlerinin en bilineni seçimli sıralamadır (selection sort). Ayrıca birer birer seçme yöntemleri kapsamındaki diğer uygulamalardan bazıları şunlardır : (Laszlo, 1996)

- Sonlu noktalar kümesinin dışbükey kabuğunun (convex hull) oluşturulması : Hediye sarma (gift wrapping) ve Graham Taraması (Graham scan)
- Bir düzlemde verilen iki dışbükey çokgenin kesişimi
- Delaunay Triangulation (Bir düzlemde verilen sonlu noktalar kümesinin özel bir şekilde üçgenlere dönüştürülmesi)

4.5.1 Seçimli Sıralama

Seçimli sıralama, küme boşalana kadar kümedeki en küçük elemanı çıkararak işleyen bir sıralama algoritmasıdır. Bir dizi elemanın sıralanmasında şu şekilde çalışır : (Laszlo, 1996)

Dizinin en küçük elemanını bulur ve bu elemanla dizinin ilk konumundaki elemanı karşılıklı olarak yer değiştirir. Sonra kalan elemanların (dizinin ilk konumundan sonraki) en küçüğünü bulur ve dizinin ikinci konumundaki elemanla karşılıklı yerlerini değiştirir. Bu şekilde devam edildiğinde dizi sıralanmış olur (Laszlo, 1996).

Seçimli sıralama bir çevrim-dışı (off-line) yöntemdir. Bunun anlamı tüm girdilerin baştan belli ve hazır olmasıdır. Baştan tüm girdi hazır olmasaydı ve en küçük eleman belirlenip karşılıklı yer değiştirme işlemi yapıldıktan sonra daha küçük bir eleman gelseydi bu yöntem

sıralamayı doğru bir şekilde yapamazdı. Birer birer seçme yöntemlerinin tümü çevrim-dışıdır. Çünkü tüm elemanlara ulaşılamazsa doğru elemanın seçilmesi garantilenemez.

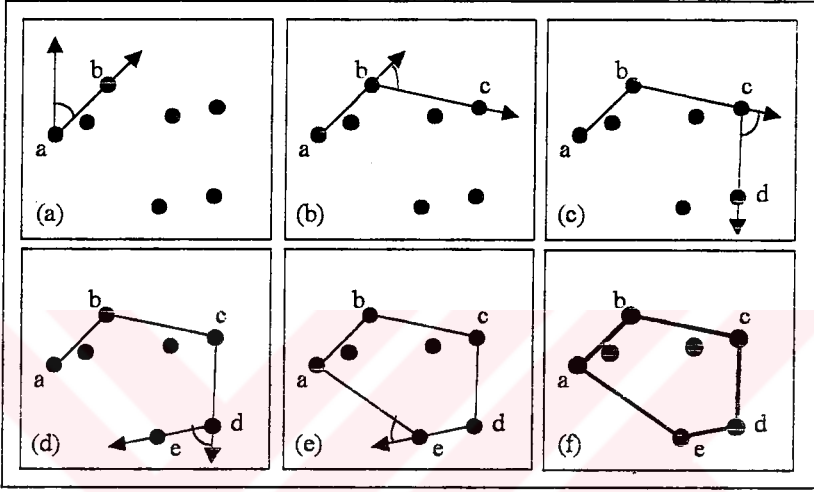
Ekleme sıralama (insertion sort) tüm birer birer ekleme yöntemleri gibi çevrimiçi (on-line) bir sıralama yöntemidir. Çevrimiçi bir yöntem girdiye bakmaz. Bunun anlamı girdinin zamanın akışına göre gelebilmesi ve baştan tüm girdinin belirli olmasının gerekmeiştir (Laszlo, 1996).

Çevrimiçi programlar gerçek zamanlı uygulamalarda çok yararlıdır. Kelime işlemciler ve uçuş simülatörleri çevrimiçidir. Çünkü bu programlara verilen girdi kullanıcı tarafından gerçek zamanlı olarak üretilir. Kullanıcının kararları da önceden belirlenemez (Laszlo, 1996).

4.5.2 Dışbükey Kabukların Bulunmasında Hediye Sarma Yöntemi

- Dışbükey kabuğun sınırı üzerinde olduğu bilinen bir nokta $a \in S$ seçilir (Örnek olarak en soldaki köşe seçilirse dışbükey kabuğun köşelerinden biri olduğu garantilenir)
- a noktası etrafında düşey bir ışın S içindeki diğer bir b noktasına rastlayıncaya kadar saat yönünde döndürülür. İlk bulunan nokta b noktasıdır. ab doğru parçası dışbükey kabuğun ilk kenarıdır.
- Diğer kenarı bulmak için sabit b noktası etrafında ab doğru parçasından uzatılan ışın saat yönünde döndürülür. İlk bulunan nokta c noktasıdır. bc doğru parçası dışbükey kabuğun sonraki kenarıdır.
- a noktasına dönünceye kadar bir önceki işlem her adımda, bulunan yeni noktadan devam edilerek tekrarlanır.

Bu yöntemle gerçek hayattaki hediyeleri paketlenme işlemine benzediğinden dolayı hediye sarma yöntemi denilmektedir (Şekil 4.6) (Laszlo, 1996).

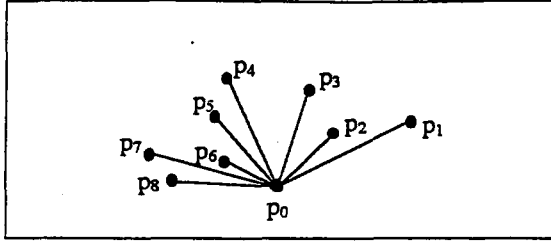


Şekil 4.6 : Bir düzlemdeki noktalar kümesinin dışbükey kabuğunun hediye sarma yöntemi ile bulunması.

4.5.3 Dışbükey Kabukların Bulunmasında Graham Taraması

R.L.Graham tarafından bulunmuş bir dışbükey kabuk bulma yöntemidir. Graham taraması, S sonlu noktalar kümesinin dışbükey kabuğunu iki aşamada bulur (Laszlo, 1996) :

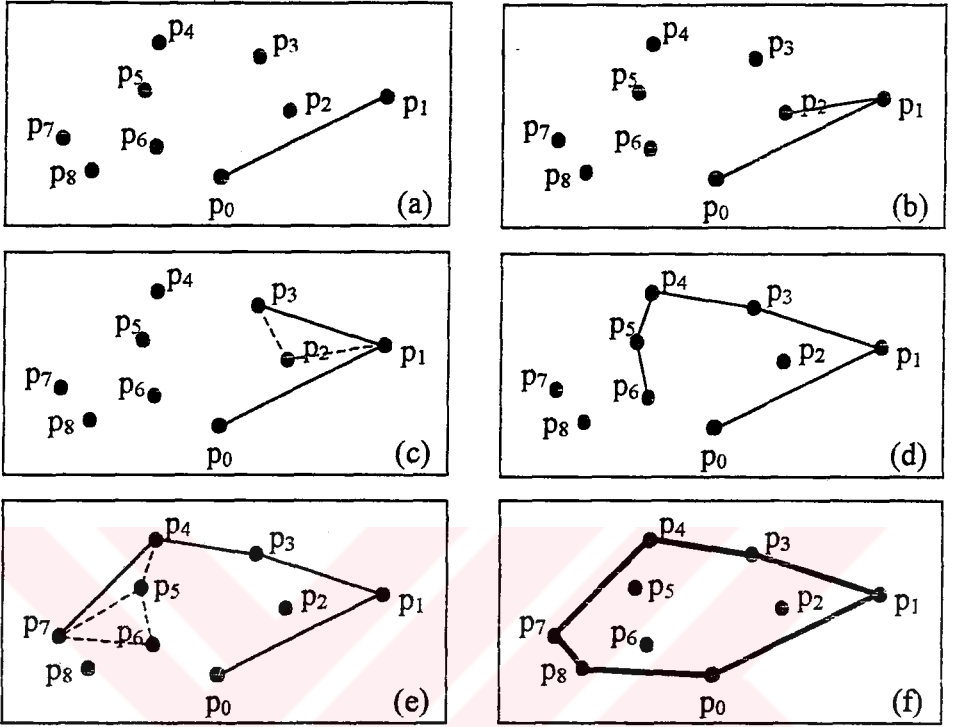
Ön sıralama aşamasında algoritma uç bir $p_0 \in S$ noktası seçer ve kalan S noktalarını p_0 etrafında açısıl olarak sıralar (Şekil 4.7). (Laszlo, 1996)



Şekil 4.7 : Graham taramasında noktaların p_0 'a göre açısal koordinatlarına bakılarak etiketlendirilmesi

Kabuk bulma aşamasında algoritma, sıralanmış noktaları yinelemeli olarak işler. Dışbükey kabuk ortaya çıkıncaya kadar oluşan ara kabuklar dizisini oluşturarak ilerler. Önceden sıralama kabuk bulma aşamasını kolaylaştırır : Çünkü noktaların sıralanmış olması, kabuk bulma aşamasında her noktanın etkin ara kabuğa doğrudan eklenebilmesini sağlar. Bu sayede etkin ara kabuktan çıkartılacak olan köşeler de kolaylıkla bulunabilir (Laszlo, 1996).

Bir örnek üzerinde durulacak olursa, Graham taraması ilk olarak S noktalar kümesinde $p_0 \in S$ uç noktasını bulur (Şekil 4.8). En küçük y koordinatına sahip olan nokta, S noktalar kümesi içindeki p_0 noktası olarak belirlenebilir. Kalan noktalar p_0 etrafında kutupsal açılarına göre sıralanır. Bu noktalar sırayla $p_1, p_2, \dots, p_{n-1}$ olarak etiketlendirilir. Aynı kutupsal açıya sahip iki noktadan ilki yakında olmalıdır. Kabuk bulma aşamasında, eklenen noktalardan sonra eklenmiş noktalardan geriye doğru gidilerek açılara bakılır ve gerekiyorsa ilgili köşeler kabuktan çıkartılır. Örnek olarak p_7 eklendiğinde (Şekil 4.8 e) $p_5p_6p_7$ açısı içbükeydir. p_6 kabuktan çıkartılır. $p_4p_5p_7$ aynı şekilde olduğundan p_5 kabuktan çıkartılır. $p_3p_4p_7$ ise dışbükey olduğundan p_4 noktasından sonra p_7 gelecektir (Laszlo, 1996).



Şekil 4.8 : Graham Taramasının bir örnek üzerindeki adımları. c'den d'ye geçişte üç adım doğrudan gerçekleştirilmiştir.

4.6 Düzlem Tarama Algoritmaları

Düzlem tarama algoritmaları, düzlemde geometrik nesneler içeren problemlerin çözümünde kullanılan etkin bir yaklaşımdır. Bu tür algoritmalarda sanal bir düşey tarama çizgisi nesneler boyunca soldan sağa doğru hareket eder. Tarama hattı ilerledikçe solunda kalan alan için problem parçası çözülmüş olmaktadır. Tarama hattı ilerledikçe bazı önemli noktalarda geçişler olur. Çokgen içeren problemlerde çokgenin köşelerine ulaşılması bu geçiş noktalarına bir örnek olarak verilebilir. Düzlem tarama algoritmalarının birçok uygulaması vardır. Çok bilinen birkaç uygulama aşağıda verilmiştir : (Laszlo, 1996)

- Bir düzlemde verilen doğru parçaları arasında kesişen doğru parçası çiftlerinin bulunması
- Düzlemdeki noktalar kümesinin dışbükey kabuğunun bulunması.
- Kenarları eksenlere paralel dikdörtgenlerden oluşan kümenin birleşiminin sınırlarının bulunması.
- Verilen bir çokgenin monoton alt çokgenlerine ayrıştırılması

4.6.1 Dışbükey Kabukların Bulunmasında Eklemeli Kabuk Yöntemi

Eklemeli kabuk algoritması, bir düzlemdeki n tane noktanın oluşturduğu dışbükey kabuğu, etkin kabuğa noktaları her defasında birer birer ekleyerek bulmaktadır. Algoritma analizi bu algoritmada iki işlemin ağır bastığını göstermektedir: (1) Eklenen her p noktasının etkin kabuğun içinde olup olmadığının bulunması, (2) p noktası etkin kabuğun dışında kaldığında yakın zincirdeki bir noktanın bulunması. Her iki işlem de etkin kabuğun boyutu ile orantılı miktarda zaman harcanmasına neden olurlar. En kötü durumda etkin kabuk her yinelemeye bir köşe genişleyeceğinden algoritmanın zaman karmaşıklığı $O(n^2)$ 'dir. Düzlem tarama yöntemi kullanılarak bu algoritmanın zaman karmaşıklığı $O(n \log n)$ 'e indirilebilmektedir (Laszlo, 1996).

Burada temel düşünce, noktaları soldan sağa doğru önceden sıralamak ve etkin kabuğa her bir elemanı bu sırada eklemektir. Bu yöntem, eklenecek her p noktasının etkin kabuk içinde olup olmadığının sıranın gerekliliğini ortadan kaldırır. Çünkü p noktası, önceden eklenen tüm noktaların sağındadır (bir önce eklenenin en azından üstündedir). Ardından p noktası etkin kabuğa ekleneceği zaman etkin kabuğun yakın zincirinin bir köşesi bulunmalıdır. Önceden sıralama yapıldığı için bu nokta da bellidir : Eklenen bir önceki nokta, bu tür bir noktadır (Laszlo, 1996).

Algoritmadaki ekleme kısmında n tane nokta olduğundan bu kısmın zaman karmaşıklığı $O(n)$ olmaktadır. Burada baskın olan işletim süresi, sıralama kısmına aittir. Birleştirmeli sıralama gibi bir sıralama algoritması sıralama işlemini $O(n \log n)$ gibi bir sürede tamamlamaktadır. Sonuçta algoritma $O(n \log n)$ işletim süresine sahiptir (Laszlo, 1996).

4.7 Böl ve Çöz Algoritmaları

Böl ve Çöz yaklaşımı, bir problemi birden fazla alt probleme böler, bu problemleri özyineli olarak çözer ve çözümlerini orijinal probleme çözüm olacak şekilde birleştirir. Alt problemler, orijinal problemin benzerleri olup boyutça ondan küçüktürler. Onların boyutları toplamı da, orijinal problemin boyutuna eşittir. Bir problem yeterince küçük ve kolay çözülebilir hale gelirse, daha fazla bölünmeksizin doğrudan çözülür ve bu, işlemi sonlandıran durumdur. En yaygın şekli ile böl ve çöz yöntemi, bir problemi her biri orijinalin yarısı olan iki alt problem haline getirmektedir (Laszlo, 1996).

Bir dizi tamsayı içerisindeki en küçük değeri bulan böl ve çöz algoritmasını düşünelim. Diziyi, sol ve sağ alt diziler olarak bölmekteki amaç, her birinde yinelemeli olarak en küçük değeri bulmak ve sonra bu iki değerden daha küçük olanını döndürmektir. Algoritma böldüğü dizileri de iki parçaya bölerek ilerler. Alt dizi boyutu bire indiğinde sonlandırma koşulu gerçekleşir ve değer geriye doğru döndürülür (Laszlo, 1996).

Bu tür algoritmaların bilinen bazı uygulamaları aşağıda belirtilmiştir :

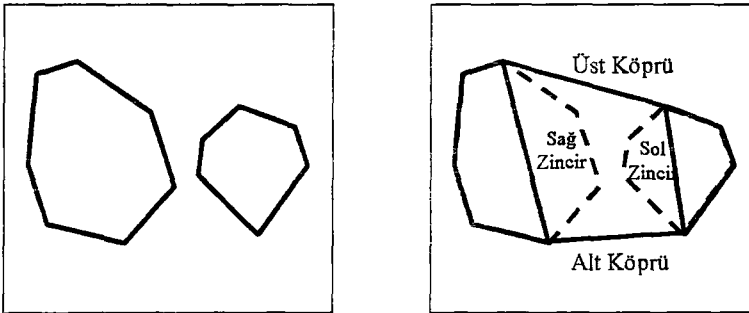
- Birleştirmeli Sıralama (Merge Sort)
- Bir düzlemdeki sonlu noktalar kümesinin dışbükey kabuğunun bulunması : Birleştirmeli kabuk (merge hull)
- Voronoi çokgenlerinin oluşturulması

- Bir düzlemdeki noktalar kümesi içindeki en yakın nokta çiftlerinin bulunması

4.7.1 Dışbükey Kabukların Bulunmasında Birleştirmeli Kabuk (Merge Hull) Yöntemi

Bu böl ve çöz algoritmasının temelinde, S noktalar kümesini sanal bir düşey çizgi ile eşit boyutlu S_L ve S_R kümelerine ayırmak ve özyineli (recursive) olarak her birinin dışbükey kabuğunu bulmak $CH(S_L)$, $CH(S_R)$ ve birleştirerek S kümesinin dışbükey kabuğunu $CH(S)$ oluşturmak düşüncesi vardır (Şekil 4.9). Bu özyineli algoritmada temel durum $|S|=1$ durumudur. (1)gen'in dışbükey kabuğu kendisidir (Laszlo, 1996).

İki dışbükey kabuğun $O(|S|)$ sürede birleştirilebilmesinden dolayı algoritma asimptotik olarak etkindir. Birleştirme için sol parçanın dışbükey kabuğunun sağ zinciri ve sağ parçanın dışbükey kabuğunun sol zinciri bulunarak yerine üst köprü ve alt köprü konulur. Noktalar sanal düşey çizginin her iki tarafında ayrı bölgelerde bulunduklarından dışbükey kabukları da ayrıdır ve bu tür köprüler kurulabilmektedir (Laszlo, 1996).



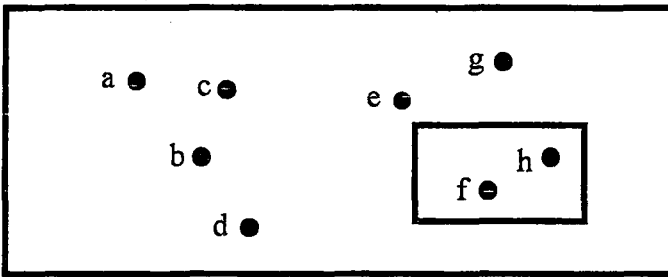
Şekil 4.9 : Farklı dışbükey kabukların birleştirilmesi

4.8 Uzaysal Bölümleme Algoritmaları

Uzaysal bölümleme algoritmaları uzaysal bir alanı küçük parçalarına ayrıştırır. A alanını veya alan içindeki geometrik nesneleri içeren problemin daha küçük ve daha basit alt problemlerine indirgenerek çözülmesini sağlar. Örnek olarak, üçgenlerine ayrılmış bir çokgenin alanı, üçgenlerin alanları toplanarak bulunabilir (Laszlo, 1996).

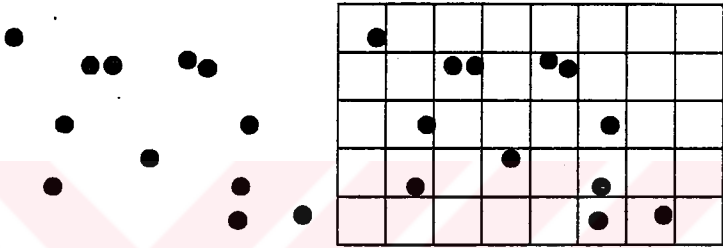
Uzaysal bölümleme büyük ölçüde şekle ve yapıya bağlıdır. Hiyerarşik bölümleme, alanın özyineli olarak küçük parçalarına ayrıştırılması ile oluşturulur. Bunun bir örneği, sayılar içeren ikili arama ağacıdır (Laszlo, 1996).

Verilen noktalar kümesinin belirli bir bölge içinde kalanlarının belirlenmesi problemi, aralık arama (range searching) problemi olarak adlandırılır. Örnek olarak belirli enlemler ve boylamlar arasındaki şehirlerin bulunması problemi verilebilir. Verilen bir noktadan veya yıldızdan verilen bir uzaklıktan daha yakın yıldızların isimlerinin bulunması da uzay araştırmalarında istenen bir bölge sorgulamasıdır (region query) (Şekil 4.10). (Laszlo, 1996)



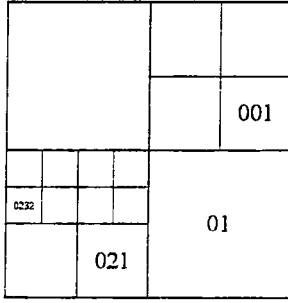
Şekil 4.10 : Bölge sorgulaması

Hiyerarşik olmayan bölümlleme yöntemlerinden birisi ızgara (grid) yöntemidir (Şekil 4.11). Bir kare alanı parçalarına ayırmanın en kolay yolu, eşit aralıklı düşey ve yatay çizgiler çekerek bu alanı hücrelerine (cell) ayırıştırırmaktır. Aralık arama probleminde $m \times m$ 'lik bir ızgara için, her bir elemanı liste olan $m \times m$ 'lik iki boyutlu bir dizi tutulabilir (Laszlo, 1996).

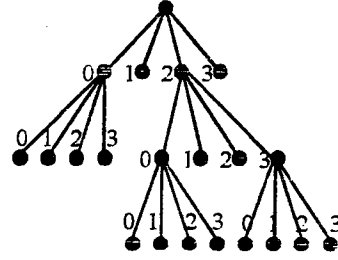


Şekil 4.11 : Izgara (Grid) Yöntemi

Hiyerarşik bölümlleme yöntemlerinden birisi, quadtree'dir (4-way tree) (Şekil 4.12). Izgara yöntemi, S noktaları düzgün dağılmadığında yani bazı bölgelerde yoğun bazı bölgelerde daha az yoğun olduğunda etkin değildir. "Quadtree" bölümlmesi bu problemi çözer. Quadtree bölümlmesi, bir A alanını 4 parçaya ayırmaya dayanır (kuzeydoğu, güneydoğu, güneybatı, kuzeybatı). Özyineli olduğundan her parça kendi parçalarına ayrıştırılabilmektedir.



(a)

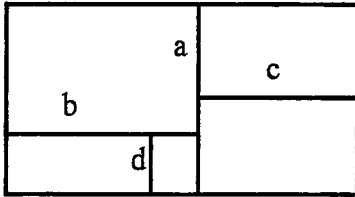


(b)

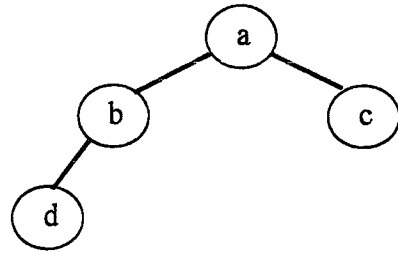
Şekil 4.12 : (a) Quadtree bölümlmesi ve (b) Quadtree gösterimi

“Quadtree” bölümlmesi, quadtree (4 yönlü ağaç) ile temsil edilir. Kök düğüm Alan A’yı göstermektedir. Kökten çıkan çocuk düğümlerin her biri çeyrek parçalık alana karşılık gelmektedir (Laszlo, 1996).

İki boyutlu ağaç (2-d tree) veya iki boyutlu ikili arama ağacı (Şekil 4.13), hiyerarşik bölümlme yöntemlerinden bir diğeridir. Bir düzlemi özyineli olarak bölen ikili ağaçtır. Bölme çizgileri yatay ve/veya düşey olabilir. Değişik boyutlarda dikdörtgenler oluşturulabilir.



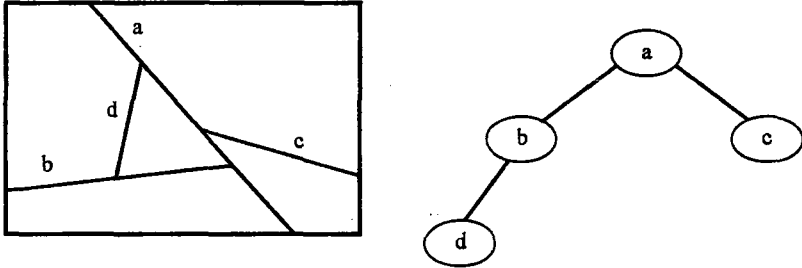
(a)



(b)

Şekil 4.13 : (a) Bir çokgenin kesim çizgileri ile ayrıştırılması ve (b) iki boyutlu ağacı

İkili uzay bölümlene ağacı (Binary Space Partition Tree) yani BSP ağacı bir düzlemi özyineli olarak çizgilerle bölümlerine ayırır (Şekil 4.14). İki boyutlu ağaçlara benzer ama bölümlene çizgileri herhangi bir yönde ve konumda olabilir (Laszlo, 1996).



Şekil 4.14 : (a) İkili uzay bölümlene ve (b) temsil eden BSP ağacı

4.9 Dışbükey Kabuk Bulma Algoritmalarının Karşılaştırılması

Dışbükey kabukların bulunmasını sağlayan algoritmaların zaman karmaşıklıklarına göre karşılaştırılması Çizelge 4.1'de verilmiştir (n : nokta sayısı, h : dışbükey kabuk üzerindeki köşe sayısı olmak üzere)

Çizelge 4.1 : Dışbükey kabuk bulma algoritmalarının zaman karmaşıklıkları

Algoritma	Zaman Karmaşıklığı	Sınıfı	Açıklama
Ekleme Kabuk	$O(n^2)$	Birer birer ekleme	
Hediye Sarma	$O(n.h)$	Birer birer seçme	$h \leq n$
Graham Taraması	$O(n \cdot \log n)$	Birer birer seçme	
Düzlem Tarama	$O(n \cdot \log n)$	Düzlem tarama	
Birleştirmeli Kabuk	$O(n \cdot \log n)$	Böl ve Çöz	

5. BİLGİSAYAR GRAFİKLERİNDE GERÇEKÇİLİK

Bilgisayar grafiklerinde yapılmakta olan çalışmaların önemli bir kısmı gerçekçiliği ve niteliği artırma üzerinedir. Bu bölümde, aydınlatma modelleri ve yüzey kaplama yöntemleri incelenmiştir.

5.1 Aydınlatma Modelleri ve Yüzey Kaplama Yöntemleri

Ortamların gerçekçi görüntüleri, nesnelerin perspektif izdüşümleri üretilerek ve görünen yüzeylere doğal aydınlatma etkileri uygulanarak elde edilir. Bir aydınlatma modeli, ışıklandırma modeli (lighting model) veya gölgelendirme modeli (shading model) olarak da adlandırılır ve nesnelerin yüzeylerindeki noktaların görülen ışık yoğunluklarını hesaplamak için kullanılır. Yüzey kaplama algoritması, sahnedeki tüm yüzeylerin izdüşüm noktalarının ışık yoğunluğunu belirlemek için aydınlatma modelinin yoğunluk hesaplamalarını kullanır. Yüzey kaplama, görünen her yüzey noktasına aydınlatma modelinin uygulanması ile yapılabildiği gibi aydınlatma modeli hesaplamalarının küçük bir bölümünün yapılmasının ardından yüzeyler boyunca ara değerlerin hesaplanması (interpolasyon) ile de tamamlanabilir. Tarama hattı (scan-line) ve görüntü uzayı (image-space) algoritmaları genelde interpolasyon düzenlerini kullanırken ışın izleme (ray-tracing) algoritmaları, her görüntü yüzey noktası konumunda aydınlatma modelinden yararlanır. Yüzey kaplama prosedürleri, yüzey gölgelendirme yöntemleri olarak da adlandırılır. Karışıklığın önlenmesi açısından tek yüzey noktasındaki yoğunluğu hesaplamak için kullanılan modele “aydınlatma modeli” veya “ışıklandırma modeli” denmesi uygun olacaktır. Yüzey kaplamayı da sahnedeki tüm yüzey izdüşüm noktalarının yoğunluklarını elde etmekte kullanılan bir aydınlatma modeli yöntemi olarak görmek yararlı olacaktır.

Bilgisayar grafiklerinde gerçekçiliği sağlamak için iki bileşene dikkat edilmesi gerekir : Nesnelerin grafiksel olarak doğru temsili ve ortamdaki aydınlatma etkilerinin (ışık yansımaları, saydamlık, yüzey dokuları yani texture ve gölgeleri ...) fiziksel tanımlamalarının iyi yapılması.

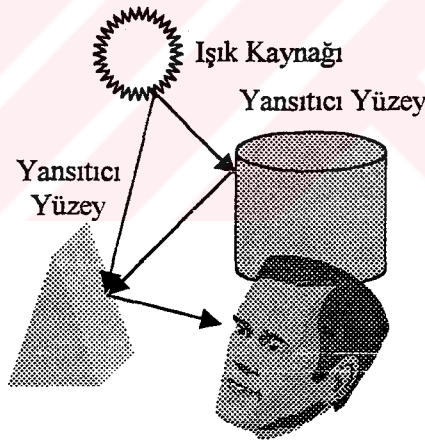
Bir nesneye bakıldığında görülen renkleri ve aydınlatma etkilerinin modellenmesi hem fizik hem de psikoloji kanunlarını içeren karmaşık bir işlemdir. Temelde aydınlatma etkileri, nesne yüzeylerinin elektromanyetik enerji etkileşimlerini dikkate alan modellerle tanımlanırlar. Işık göze ulaştığında ortamda gerçekte neyin görüldüğünü belirleyen algılama işlemleri başlatılır. Fiziksel aydınlatma modelleri, nesnenin türü, nesnenin ışık kaynaklarına ve diğer nesnelere göre yeri ve ortam için ayarlanan ışık kaynağı şartları gibi bir dizi faktörü içerirler. Nesneler, saydam olmayan modellerden oluşabildiği gibi, az veya çok saydam da olabilirler. Ayrıca parlak veya donuk yüzeyleri olabildiği gibi değişik yüzey dokularına da sahip olabilirler. Değişik tür, şekil, renk ve konumlardaki ışık kaynakları, ortamdaki aydınlatma etkilerini sağlamak için kullanılabilirler. Yüzeylerin optik özellikleri ile ilgili olarak verilen parametreleri, ortamdaki yüzeylerin göreceli konumlarını, ışık kaynaklarının renkleri ve konumlarını, görüntü düzleminin yeri ve yönünü dikkate alan aydınlatma modelleri, belirtilen görüntü yönünde tanımlanmış belirli bir yüzey noktasından izdüşümü alınan yoğunluk değerini hesaplarlar (Hearn ve Baker, 1997).

Bilgisayar grafiklerindeki aydınlatma modelleri, yüzey ışık yoğunluklarını tanımlayan fizik kanunlarından türetilir. Yoğunluk hesaplamalarını en aza indirmek için birçok grafik paketi basitleştirilmiş fotometrik hesaplamalara dayalı deneysel modeller kullanır. Işıma (radiosity) algoritması gibi daha gerçekçi modeller ışık yoğunluklarını ortamdaki yüzeyler ve ışık kaynakları arasındaki ışıma enerjisinin

yayılımını dikkate alarak hesaplarlar. İzleyen bölümlerde, ilk olarak grafik paketlerinde yaygın olarak kullanılan temel aydınlatma modellerine değinilecektir. Sonra da, yüzey yoğunluklarının hesaplanmasında kullanılan daha doğru fakat daha zaman alıcı yöntemler ve ortamdaki görünür yüzeylerin uygun bir şekilde gölgelendirilmesini sağlayan bazı yüzey kaplama algoritmaları anlatılacaktır.

5.2 Işık Kaynakları

Saydam olmayan ve ışık saçmayan bir nesneye bakıldığında nesnenin yüzeylerinden yansıyan ışık görülür. Toplam yansıyan ışık, tüm ışık kaynaklarının ve ortamdaki diğer yansıtıcı yüzeylerden gelen yansımaların katılımlarının toplamına eşittir (Şekil 5.1).



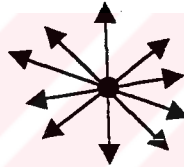
Şekil 5.1 : Işık yansımaları

Sonuçta, bir ışık kaynağından doğrudan ışık almayan bir yüzeyin yakınındaki nesneler de aydınlatılırsa görülebilir. Bazen ışık kaynakları, ışık yayan kaynaklar olarak, oda duvarları gibi yansıtıcı yüzeyler de ışığı

yansıtıcı kaynaklar olarak adlandırılır. Işık kaynağı deyimi genelde güneş ve lamba gibi ışıma enerjisini yayan nesneler için kullanılacaktır.

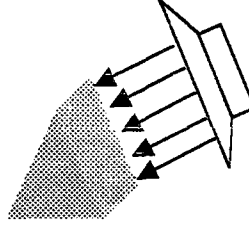
Işık yayan bir nesne genelde hem bir ışık kaynağı hem de ışık yansıtıcısı olabilir. Örnek olarak içinde lamba olan plastik bir küre hem ışık yayar hem de kürenin yüzeyinden ışık yansıtır. Küreden yayılan ışık, çevresindeki diğer nesneleri aydınlatabilir.

En basit ışık yayan model *nokta kaynağıdır* (point source). Böyle bir kaynağın ışınları Şekil 5.2'de gösterildiği gibi kaynağın bulunduğu yerde merkezden çıkan yolları izler (Hearn ve Baker, 1997).



Şekil 5.2 : Nokta ışık kaynağından çıkan ışın yolları

Işık kaynağı modeli, ortamdaki nesnelerin büyüklükleri ile karşılaştırıldığında küçük boyutları olan kaynaklar için uygun bir yaklaşımdır. Güneş gibi ortamdaki yeterince uzak kaynaklar, nokta kaynaklar şeklinde doğru olarak modellenenebilir. Şekil 5.3'teki uzun floresan ışığı gibi yakındaki bir kaynağı, dağıtık ışık kaynağı (distributed light source) olarak modellemek daha uygun olur. Bu durumda aydınlatma etkilerine nokta kaynak ile gerçekçi olarak yaklaşım yapılamaz. Çünkü kaynağın alanı ortamdaki yüzeyler ile karşılaştırıldığında küçük değildir. Dağıtık kaynak için doğru bir model, kaynağın yüzeyi üzerindeki noktaların aydınlatma etkilerinin toplamını dikkate almalıdır (Hearn ve Baker, 1997).



Şekil 5.3 : Dağıtık ışık kaynağı ile aydınlatılmış bir nesne

Işık saydam olmayan bir yüzeye geldiğinde, bir miktarı yansıtılırken bir miktarı emilir. Işığın yüzey tarafından yansıtılma miktarı, maddenin türüne bağlıdır. Parlak nesneler gelen ışığın çoğunu yansıtırken donuk yüzeyler gelen ışığın çoğunu emer. Aynı şekilde aydınlatılmış saydam bir yüzey, gelen ışığın birazını yansıtırken birazını da maddeden geçirecektir.

5.3 Temel Aydınlatma Modelleri

Verilen yüzey noktalarındaki ışık yoğunluklarını hesaplamada kullanılan basitleştirilmiş yöntemler vardır ve yeterince iyi sonuçlara ulaşılmasını sağlarlar. Aydınlatma hesaplamaları yüzeylerin optik özelliklerine, ortamın aydınlatma şartlarına ve ışık kaynaklarının özelliklerine dayalı olarak yapılır. Parlaklık, matlık, yoğunluk ve saydamlık gibi optik parametreler yüzey özelliklerini ayarlama da kullanılır. Bunlar gelen ışığın emilme ve yansıma miktarını kontrol eder. Nokta kaynak şeklindeki tüm ışık kaynakları, koordinat konumu ve yoğunluk değeri ile belirtilir.

En basit aydınlatma modeli, nesneyi kendine ait bir renk yoğunluğu kullanarak görüntülemek şeklinde gerçekleştirilir. Bu, gerçekçi olmayan yani yansıtıcı olmayan ve sadece kendini aydınlatan nesnelerden oluşan dünyalar için geçerli bir modeldir. Bir çokyüzlünün çokgenleri gibi her özel parça tek renklidir. Bu basit modelin aydınlatma denklemi şu şekilde belirtilebilir (I : sonuç yoğunluk ve k_i : nesnenin yoğunluğu)

$$I = k_i \quad (5.1)$$

Nesnenin bir veya daha çok noktası için aydınlatma denkleminin uygulanması *aydınlatma* olarak adlandırılır.

5.3.1 Genel Aydınlik (Ambient Light)

Bir ışık kaynağını doğrudan görmeyen bir yüzey, yakınındaki nesneler aydınlatıldığında da görülecektir. Bir aydınlatma modelinde ortamın genel aydınlatma düzeyi belirtilmelidir. Tüm yüzeylerden gelen ışık yansımalarının toplamını, modellemenin en basit yolu sabit bir aydınlatma belirtmektir. Ortamın aydınlığı (ambient light, background light) tüm ışık yansımalarının bir toplamı olarak genel bir aydınlanma olarak algılanmalıdır. Bu aydınlığın uzaysal veya yönlü özellikleri yoktur. Ortamdan her nesneye gelen ışık miktarı tüm yüzeyler için tüm yönlerde sabittir. Ortamın aydınlık düzeyi I_a parametresi ile belirtilir ve her yüzey bu sabit değerle aydınlatılır.

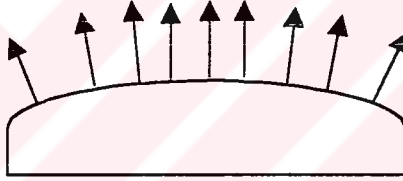
Her nesnenin kendi aydınlığı yerine modelde, yayılan yönsüz bir ışık kaynağı olduğu düşünüldüğünde karşımıza genel aydınlık çıkar. Genel aydınlık denklemi (I : sonuç yoğunluk, I_a : genel aydınlatma yoğunluğu ve k_a : 0 ile 1 arasında değişen nesnenin aydınlatma katsayısı olmak üzere),

$$I = I_a k_a \quad (5.2)$$

şeklinde belirtilebilir.

5.3.2 Yayılan Yansıma (Diffuse Reflection)

Genel aydınlık, yayılan aydınlatma etkilerinin toplamını bulmakta kullanılan bir yaklaşımdır. Yayılan yansımalar bakış yönünden bağımsız olarak ortamdaki tüm nesneler için sabittir. Bu tür yayılma etkisi, *yayılan yansıma* (diffuse reflection) olarak adlandırılır. Düzgün olmayan yüzeyler birinci derecede yayılan yansımaya neden olduklarından yüzeyleri her bakış noktasından aynı parlaklıkta görülür. Şekil 5.4'te, yüzeyden yansıyan yayılan ışık görülmektedir.



Şekil 5.4 : Bir yüzeyden yayılan yansımalar.

Bir nesneden yayılan yansıma nesnenin rengidir. Beyaz bir ışık kaynağı ile aydınlatılmış mavi bir nesne, beyaz ışığın mavi bileşenini yansıtır ve diğer bileşenlerinin tümünü emer. Mavi bir nesneye kırmızı ışık altında bakıldığında gelen ışığın tümünün emilmesinden dolayı siyah olarak görülür.

Gelen ışığın bir bölümü yayılarak yansıtılır ve bu miktar yayılan yansıma katsayısıdır ve her bir yüzey için k_d parametresi ile gösterilir. k_d 'nin değeri yüzeyin yansıtıcılık özelliklerine göre 0 ile 1 arasında belirlenir. Çok yansıtıcı yüzeylerde k_d değeri 1'e yakındır ve bu durumlarda yüzey, gelen ışık miktarının tümüne yakın bir kısmını yansıtacağından yoğunluğu gelen ışığın düzeyine yakındır. Gelen ışığın

çoğunu emen bir yüzeyi simüle etmek için k_d değeri 0'a yakın tutulmalıdır. k_d , yüzey renginin bir fonksiyonu olmasına rağmen kolaylaştırmak açısından sabit kabul edilebilir. Sadece ortamdaki ışığı alan bir yüzeyin herhangi bir noktasındaki aydınlık :

$$I_{ambdiff} = k_d I_a \quad (5.3)$$

ile hesaplanır.

Sadece genel aydınlık yeterince gerçekçi olmayacağından yüzey kaplamalarda nadiren tek başına kullanılır. Ortama en azından bir ışık kaynağı eklenir ve genelde bu ışık kaynağı bakış noktasına yerleştirilir.

Bir nokta kaynağın neden olduğu yayılan yansımalar da benzer şekilde modellenir. Bakış noktasından bağımsız olarak yüzeyden yayılan yansımaların tüm yönlerde eşit miktarlarda olduğu varsayımı yapılan yüzeyler ideal yansıtıcılar (ideal diffuse reflectors) olarak adlandırılır. Yüzeyin herhangi bir noktasından yayılan enerji Lambert kanununa (Lambert cosine law) göre düzenlendiğinden bu yüzeylere Lambert yansıtıcıları (Lambertian Reflectors) adı da verilir. Tebeşir gibi donuk (mat) yüzeyler Lambert yansıması ile modellenirler. Bu yüzeyler tüm bakış açılarından eşit parlaklıkta görülürler. Çünkü ışığı tüm yönlerden eşit miktarlarda yansıtırlar.

Bu kanuna göre küçük bir dA alanından herhangi bir ϕ_N yönünde yayılan ışık enerjisi, yüzey normaline göre yaptığı θ_N açısının kosinüsüne bağlıdır.

İdeal bir yansıtıcıda tüm yönlerde eşit miktarlarda ışık yayılmasına rağmen yüzeyin aydınlığı, yüzeyin ışık kaynağına göre yönüne bağlı olarak hesaplanır. Gelen ışığın yönüne dik olan yüzeyler eğik olan yüzeylere göre daha parlak görünür (Şekil 5.5). Örneğin beyaz bir kağıt, pencere yanında tutulup yavaşça döndürüldüğünde bu etki görülür.

Yüzey normali ile gelen ışık arasındaki açı arttıkça, gelen ışığın daha az bir miktarı yüzey üzerine düşer.

Gelen ışığın yönü ile yüzey normali arasındaki açıya gelme açısı (θ) denilmektedir. Aydınlanma miktarı (izdüşüm yüzey parçasını kesen ışık ışınlarının sayısı) $\cos\theta$ ile orantılıdır. Gelen ışığa dik olan yüzey tüm ışık ışınlarını alır. Eğim arttıkça yüzeyin aydınlığı azalır.



Şekil 5.5 : Gelen ışığın yönüne dik ve eğik yüzeyler. Gelen ışığın yönüne dik olarak yerleştirilmiş olan yüzey eğik olarak yerleştirilmiş aynı büyüklükteki bir yüzeye göre daha fazla ışık alır.

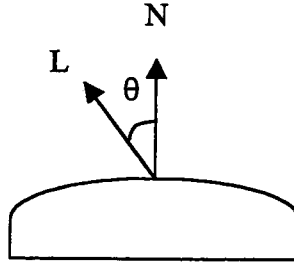
$I_l (I_p)$: noktasal ışık kaynağının yoğunluğu ve

k_d : maddenin yayılan yansıma katsayısı (0 ile 1 arasında)

olmak üzere, bir yüzey üzerindeki bir noktanın yayılan yansıma denklemi aşağıdaki gibidir :

$$I_{l,diff} = k_d I_l \cos \theta \quad (5.4)$$

Verilen bir yüzey için parlaklık, ışık kaynağına giden yön (L) ile yüzey normali (N) arasındaki açıya bağlıdır (Şekil 5.6).



Şekil 5.6 : Yayılan yansıma

Bir yüzey, geliş açısı 0° ile 90° arasında değişen nokta ışık kaynağı ile aydınlatılır (Bu durumda $\cos\theta$, 0 ile 1 aralığındadır. θ açısı negatif olduğunda ışık kaynağı yüzeyin arkasında demektir).

Formül hesaplama kolaylığı açısından $\cos\theta = (N.L)$ ifadesinden yararlanarak şu şekilde de ifade edilebilir :

$$I_{l,diff} = k_d I_l (N.L) \quad (5.5)$$

N : yüzeyin birim normal vektörü

L : yüzeyden ışık kaynağına birim yön vektörü

Işık kaynağı nesnelerden yeterince uzağa konulduğunda aynı yüzey normaline sahip tüm yüzeyler aynı açıyı yaptıklarından bu ışık kaynakları *yönlü ışık kaynağı* olarak adlandırılır ve L sabittir.

Yukarıda verilen formüle genel aydınlığın etkisi de katıldığında, ortamın yansıtma katsayısı k_a olmak üzere, toplam yayılan yansıma denklemi :

$$I_{diff} = k_a I_a + k_d I_l (N.L) \quad (5.6)$$

gibi ifade edilebilir.

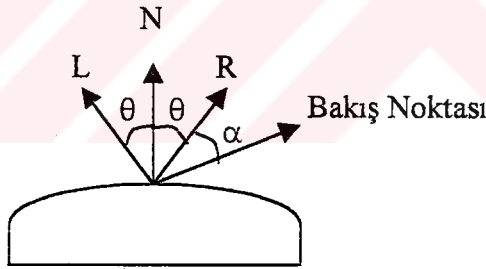
Ortamda birden fazla ışık kaynağı olduğunda ise bu formül,

$$I_{diff} = k_a I_a + \sum_{i=1}^n k_d I_{li} (N.L_i) \quad (5.7)$$

şeklini alır.

5.3.3 Yönlü Yansıma (Specular Reflection)

Yayılan yansıma dışında ışık kaynakları, elma ve parlak metal gibi yüzeylere belirli yönlerden bakıldığında, bu tür yüzeylerde ışıklı bölümler veya parlak noktalar oluşturabilir. Buna *yönlü yansıma* adı verilir. Bu etki donuk yüzeylere nazaran parlak yüzeylerde daha belirgin olarak görülür. Nesnelerin geri kalan kısmı yayılan yansıma ile aydınlatılabilir. Yönlü yansımada yüzeyler, farklı yönlerden bakıldıklarında farklı parlaklıkta görülürler (Şekil 5.7).



Şekil 5.7 : Yönlü yansıma

İdeal ayna gibi ideal parlak yüzeylerde L yönünde gelen ışık sadece R yönünde yansıtılır. Phong Bui-Tuong, elma gibi ideal olmayan yansıtıcılar için Phong aydınlatma modeli diye bilinen modeli geliştirmiştir (Foley ve ark., 1997).

5.4 Basit Aydınlatma : Çokgen kaplama yöntemleri

Aydınlatma modeli bilindikten sonra, nesnenin yüzeyinin kaplanması için, görünen her noktası için yüzey normalinin hesaplanması ve ardından aydınlatma modelinin uygulanması gerektiği açıktır. Ancak bu doğrudan yöntem çok uzun süreceğinden çokgenler üzerinde daha etkin kaplama modelleri tanımlanmıştır. Sabit grafik nesnelerini kaplama yöntemleri (aydınlatma modelinin bir uygulaması) çokyüzlülere, eğrisel yüzeylere ve çokgensel yüzey yaklaşımı ile oluşturulan nesnelere de uygulanabilir. Bu amaçla kullanılan iki yöntem vardır : Her çokgen sabit bir yoğunlukla kaplanabilir veya yüzeylerin her bir noktasının yoğunluk değeri interpolasyon kullanılarak elde edilebilir. Bu yöntemler hızlı ve basit aydınlatmada kullanılırlar.

5.4.1 Sabit Yoğunluklu Kaplama

Çokgen kaplama modellerinden en basiti sabit yoğunluklu kaplamadır. Sabit yoğunluklu kaplamada (constant intensity shading = flat shading) her çokgen için sabit yoğunluk hesaplanır. Bu model genelde eğrisel yüzeylerin genel görünümünün hızlı bir şekilde oluşturulmasında kullanılır. Bu yaklaşım bazı varsayımlar yapıldığında doğrudur :

- Nesne eğrisel yüzey yaklaşımı ile oluşturulmamıştır. Çokyüzlü şeklindedir.
- Nesneyi aydınlatan tüm ışık kaynakları, yüzey boyunca (N.L) sabit olacak şekilde yüzeyden yeterince uzaktadır veya sonsuzdadır.
- Bakış noktası da, (V.R) değeri yüzey boyunca sabit olacak şekilde yüzeyden yeterince uzaktadır veya sonsuzdadır.

Tüm bu şartlar yeterince doğru olmasa bile sabit gölgelendirme kullanılarak küçük çokgen yüzeyler üzerinde yeterince iyi yaklaşık

sonuçlar elde edilir. Avantajı ise çokgenin tek renkli bir çokgen deyimini ile ekrana çizilebilmesidir.

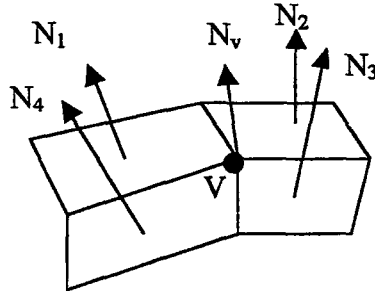
5.4.2 Gouraud Gölgelemesi

Gouraud tarafından geliştirilen **yoğunluk** interpolasyon düzeni Gouraud gölgelemesi olarak anılır. Bu yöntem, yoğunluk değerlerini yüzey boyunca doğrusal olarak hesaplayarak yüzeyi kaplar. Her çokgenin yoğunluk değerleri, ortak kenarlara sahip bitişik çokgenlerin değerleri kullanılarak hesaplanır. Bu, sabit yoğunluk kaplamada görülen ve sürekliliği bozan yoğunluk atlamaları problemini ortadan kaldırır. Gouraud gölgelemesinde her çokgenin yüzeyi, izleyen adımların uygulanması ile kaplanır:

- Her çokgen köşesi için ortalama birim normal vektörü belirlenir (köşeye komşu çokgensel yüzeylerin normal vektörleri ortalaması).
- Her köşenin yoğunluğu aydınlatma modeli uygulanarak bulunur.
- Çokgen yüzeyi boyunca köşe yoğunlukları doğrusal olarak interpolasyondan geçirilir.

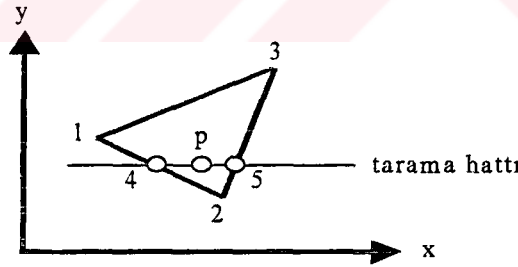
Herhangi bir V köşesini paylaşan tüm çokgen yüzeylerinin normallerinin ortalaması alınarak belirlenen normal vektörü (Şekil 5.8):

$$N_v = \frac{\sum_{k=1}^n N_k}{\left| \sum_{k=1}^n N_k \right|} \quad (5.8)$$



Şekil 5.8 : Gouraud gölgelendirmesinde normal vektörü hesaplaması

formülü ile bulunur. Gouraud gölgelendirmesinde sonraki adım, interpolasyon kullanılarak yoğunluk değerlerinin bulunmasıdır (Şekil 5.9). 4 noktasının yoğunluk değeri, 1 ve 2 noktalarındaki yoğunluk değerleri kullanılarak doğrusal aradeğerleme yöntemi ile hesaplanır. Aynı şekilde 5 noktasının yoğunluk değeri 2 ve 3 noktalarının yoğunluk değerleri kullanılarak; p'nin yoğunluk değeri de hesaplanan 4 ve 5 noktalarının yoğunluk değerleri kullanılarak hesaplanır.



Şekil 5.9 : Gouraud gölgelendirmesinde aradeğerleme

Kenar üzerindeki herhangi bir I_4 noktası, kenar uç noktalarının yoğunluk değerleri yardımı ile aşağıdaki interpolasyon formülü kullanılarak hesaplanır.

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2 \quad (5.9)$$

Benzer şekilde I_5 noktası, 2 ve 3 köşe noktalarının yoğunluk değerlerinden yararlanılarak hesaplanır.

Tarama hattı üzerindeki herhangi bir $p(x_p, y_p)$ noktasının yoğunluk değeri de 4 ve 5 uç noktalarının yoğunluk değerleri yardımıyla interpolasyon kullanılarak aşağıdaki gibi hesaplanır :

$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5 \quad (5.10)$$

Yüzeyler renkli kaplanmak istenirse, bu işlemler her yoğunluk bileşeni için ayrı ayrı uygulanır. Gouraud gölgelendirmesi, sabit gölgelendirme modelindeki yoğunluk atlamaları problemini ortadan kaldırır. Fakat bazı anormal şekillerde “mach şeridi” adı verilen parlak ya da koyu yoğunluk çizgileri oluşur. Bu istenmeyen etkiler, yüzeyin çok sayıda çokgen yüzeylere parçalanması ile veya daha fazla hesaplama gerektiren diğer yöntemlerin (Phong gölgelendirmesi gibi) kullanılması ile ortadan kaldırılabilir.

5.4.3 Phong Gölgelelendirmesi

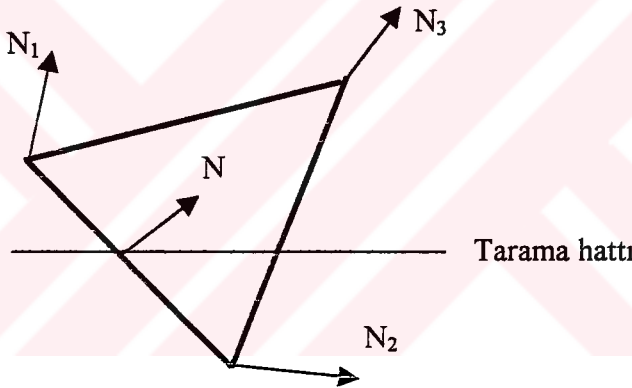
Phong Bui Tuong tarafından geliştirilmiştir. Normal vektörü interpolasyonu kullanılır. Mach şeridi etkisini ortadan kaldırarak daha gerçekçi görüntüler oluşturur.

Phong gölgelendirmesinde her çokgenin yüzeyi aşağıda belirtilen adımların uygulanması ile kaplanır:

- Her çokgen köşesi için ortalama birim normal vektörü belirlenir.

- Çokgen yüzeyi boyunca köşe normalleri doğrusal olarak interpolasyondan geçirilir.
- Her tarama hattı boyunca yüzey noktalarının izdüşüm piksel yoğunlukları hesaplanır (her noktada ayrı yüzey normali olduğundan aydınlatma modeli her noktaya uygulanır).

Phong gölgelendirmesi daha çok hesaplama gerektirir. Örnek olarak iki köşe noktası arasındaki çokgen kenarı boyunca yüzey normallerinin ara değerlerinin elde edilmesi (interpolasyon) Şekil 5.8'de gösterilmektedir.



Şekil 5.10 : Çokgen kenarı boyunca yüzey normali aradeğerlemesi

$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2 \quad (5.11)$$

Tarama hattı boyunca bulunan noktaların yoğunlukları doğrudan interpolasyon yolu ile veya ara noktaların yüzey normaleri hesaplanarak doğrudan aydınlatma modeli uygulanarak hesaplanabilir.

5.5 Genel Aydınlatma Algoritmaları

Genel aydınlatma algoritmaları, ışığın tüm yüzeyler arasında geçişi sırasında oluşan etkileri de dikkate alarak daha gerçekçi görüntüler oluşturmayı sağlar. En önemli iki genel aydınlatma algoritması, özyineli ışın izleme (recursive ray tracing) ve ışımadır (radiosity). Basit aydınlatma algoritmalarına göre daha uzun sürerler ve kaplama işlemlerinin adımları da farklı olmaktadır (Foley ve ark., 1997).

Işın izleme, parlak yüzeylerdeki yansımaları elde etmek için kullanılan bir yöntemdir. Ekrandaki noktaların (piksel) her birinden gönderilen ışınlar nesneden nesneye yansırken, elde edilen yoğunluk katılım değerleri toplanır. Bu sayede ekrandaki her bir noktanın yoğunluk ve renk değerleri hesaplanır (Hearn ve Baker, 1997).

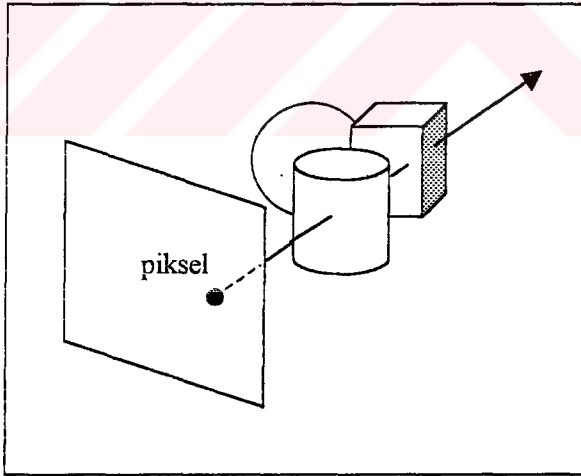
Işıma yöntemleri de, ortamdaki değişik yüzey parçaları arasında yayılan ışıma enerji aktarımlarını, yani yönsüz aydınlanmayı modellemek için kullanılır. Işıma değeri, yüzeyden çıkan enerji oranıdır. Aşamalı Arındırma (Progressive Refinement) yöntemi, her defasında bir yüzeyden enerji aktarımı yaparak, ışıma hesaplamalarını hızlandırır (Hearn ve Baker, 1997).

Çok gerçekçi görüntüler, ışın izleme ve ışıma yöntemlerinin her ikisinin birlikte kullanılması ile elde edilir (Hearn ve Baker, 1997). Işın izleme 'bakış noktası bağımlı' bir teknik ve ışıma da 'bakış noktası bağımsız' bir teknik olduğundan birbirlerinin tamamlayıcısıdır (Ashdown, 1994).

Ayrıca nesnelere yüzey ayrıntılarını verme, desen eşleme, yüzey şekillerini verme gibi daha gerçekçi etkiler de uygulanabilmektedir.

5.5.1 Işın Dağıtma (Ray-Casting) Yöntemi

Ortamdaki görüntü düzlemi üzerindeki bir noktanın (piksel) bulunduğu konumdan görülen çizgi düşünülecek olursa (bkz. Şekil 5.11), hangi nesnelerin (eğer varsa) bu görüntü noktasından çıkarılan doğru ile kesiştiği belirlenebilir. Tüm ışın-yüzey kesişimlerinin hesaplanmasından sonra, kesişim noktası görüntü noktasına en yakın olan görünen yüzey belirlenir. Bu görünürlük belirleme düzeni, ışın dağıtma yöntemlerini kullanır. Işın dağıtma, görünürlük belirleme aracı olarak, ışık ışınlarının yollarını izleyen optik geometri yöntemlerine dayanır. Ortamda sonsuz sayıda ışık ışını olduğundan ve sadece görüntü noktası konumundan geçenlerle ilgilenildiğinden dolayı, ışık ışınlarının yolları, ortamdaki görüntü noktalarından geriye doğru izlenebilir. Işın dağıtma yöntemi, küre gibi eğrisel yüzeyli ortamlar için etkin bir görünürlük belirleme yöntemidir.



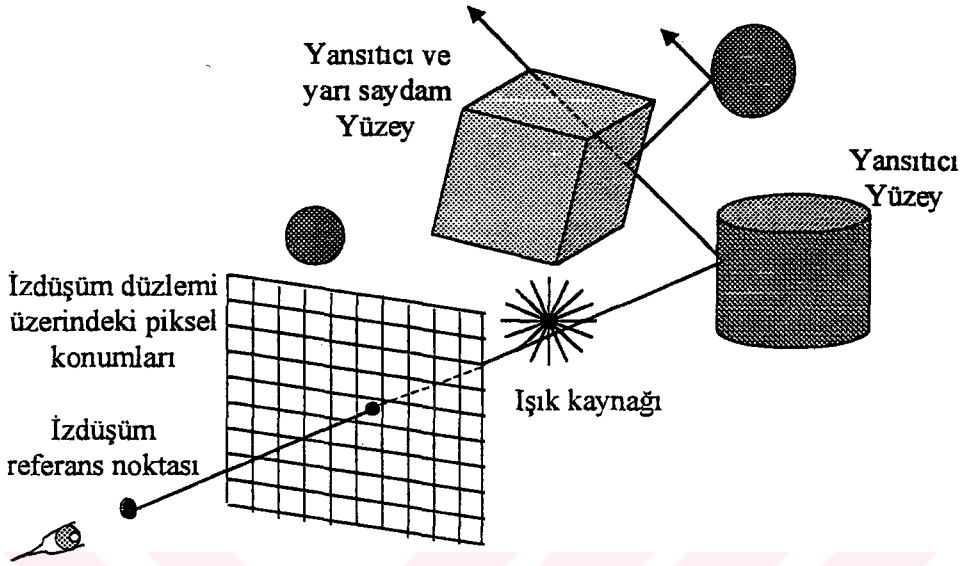
Şekil 5.11 : Işın dağıtma yöntemi. Ortamdaki görüntü düzlemi noktasından çıkan doğru boyunca sahneye gönderilen ışın.

Işın dağıtma yöntemi, derinlik arabelleği (depth buffer) yönteminin değişik bir şekli olarak düşünülebilir. Derinlik arabelleği algoritmasında yüzeyler işlenir ve yüzeydeki tüm izdüşüm noktalarının derinlik değerleri hesaplanır. Hesaplanan yüzey derinlikleri, her görüntü düzlemi noktası için görünen yüzeylerin belirlenmesi amacıyla önceden kaydedilmiş derinlik değerleri ile karşılaştırılır. Işın dağıtma yönteminde de bir anda bir görüntü düzlemi noktası için işlem yapılır ve bu görüntü düzlemi noktasının izdüşüm yolu üzerinde olan tüm yüzeylerin derinlikleri hesaplanır.

Işın dağıtma, ortamdaki tüm nesnelerin oluşturduğu tüm yansıma ve kırılma katılımlarını da dikkate alacak şekilde birçok ışın yollarını izleyen ışın izleme (ray-tracing) algoritmalarının özel bir durumudur. Işın dağıtmada, her görüntü düzlemi noktasından çıkan ışın en yakın nesneye kadar takip edilir. Küre gibi bilinen nesneler için etkin ışın-yüzey kesişim hesaplamaları geliştirilmiştir.

5.5.2 Özyineli Işın İzleme (Recursive Ray Tracing)

Işın izleme (ray tracing), üç boyutlu sanal bir ortamın iki boyutlu resmini oluşturma işlemidir. Bu ortamda veya modelde yüzeyler ve ışık kaynakları bulunabilmektedir. Bakış (izdüşüm referans) noktası, modelin ekranda görüntülenecek olan kısmını belirler. Bakış noktasının önünde bir görüntü düzlemi seçilir. Modelin bakış noktasından görünen bölümü, görüntü (izdüşüm) düzleminde resmi biçimlendirir (Şekil 5.12). (Reinhard ve Jansen, 1995) (Reinhard, 1996) (Reinhard ve Jansen, 1997) (Reinhard ve ark., 1999). Işık ışınları ışık kaynaklarından her yöne yayılabilmektedir. Bu ışınların bir bölümü yüzeylere çarpmakta ve sonra kısmen emilmekte, yansıtılmakta veya kırılmaktadır. Bazı ışık ışınları yansıtıldıktan veya kırıldıktan sonra yollarına devam etmektedirler. Bakış noktasından geçme veya geçmeme olasılıkları vardır.



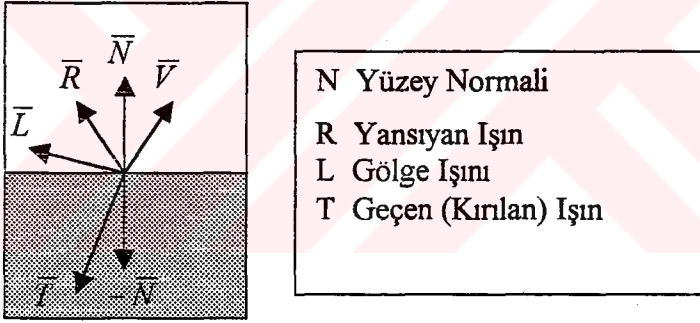
Şekil 5.12 : Işın İzleme

Bir resmi oluşturmak için işlem tersine çevrilir ve ışınlar bakış noktasından ortama doğru izlenir. Bu ışınlar *birincil ışınlar* adı verilmektedir. Böyle bir ışın bir yüzeyle kesiştiğinde yüzeyin gölgelendirmesi belirlenir ve görüntü düzlemindeki piksele atanır. Bu yöntem, ışın izlemenin en temel şeklidir. Resmin kalitesini artırmak veya işlem sayısını azaltmak için, aydınlatma etkilerini daha çok dikkate alacak şekilde iyileştirmeler yapılabilmektedir.

Temel ışın izleme algoritması, gözden çıkan ışın ile bir nesnenin kesişimine en yakın pikselin rengini belirlemekle sınırlıdır. Işın izleme algoritmaları, gölgeleri, yansımaları ve kırılmaları da dikkate almalıdır. Gölgeli bölgeleri bulabilmek için kesişim noktalarından tüm ışık kaynaklarına da ışınlar gönderilir. Işınlar arada herhangi bir nesneye rastladıklarında nesne bu noktada gölgelidir ve algoritma bu ışık kaynağının katılımını dikkate almaz. Bilgisayar grafiklerinde ışın izlemeye ilişkin bir makale Arthur Appel tarafından 1968 yılında yayınlanmıştır. Whitted (1980) ve Kay (1979) temel ışın izleme

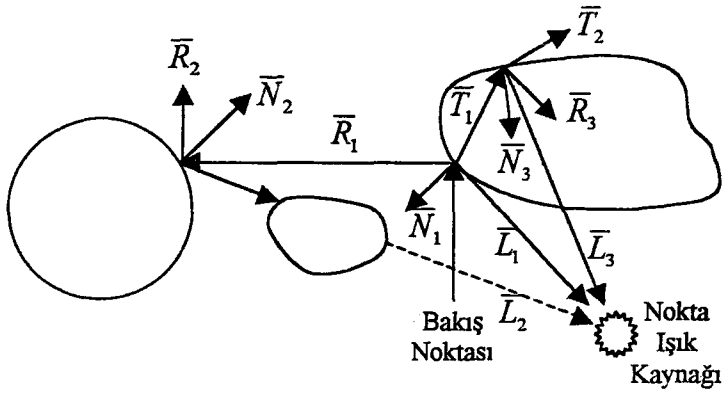
algoritmalarını, yönlü yansımaları ve kırılganlığı da dikkate alacak şekilde genişletmişlerdir. Whitted'in algoritması, ikincil ışınları da dikkate almaktadır. Bu konuda ayrıntılı bilgi Foley'de (1997) yer almaktadır. İkincil ışınlar (Şekil 5.13) :

- *Gölgeleştirme ışınları* (aydınlatma ışınları) : Bu ışınlar, ışığı, ışık kaynağından yüzeye doğrudan taşırlar.
- *Yansıyan ışınlar* : Bu ışınlar, yansıma yönünden gelen ışığın katılımını hesaplamada kullanılırlar.
- *Kırılan veya geçen ışınlar* : Bu ışınlar, ışığı nesne boyunca taşırlar. Bir nesnenin geçirgenliği varsa, ışınlar bu ortama geçerken *Snell kanununa* göre kırılırlar.

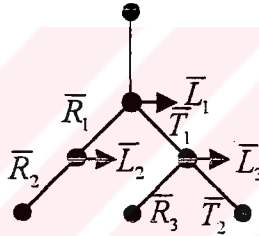


Şekil 5.13 : Kesişim noktasından çıkan yansıma, kırılma ve gölge ışınları

Yansıyan ve geçen ışınlar birincil ışınların oluşturulmasına benzer şekilde oluşturulduklarından algoritmaya özyineleme getirirler (Şekil 5.14). Bitiş şartı olarak, özyinelemenin en yüksek düzeyini belirtmek üzere genelde bir eşik değeri verilir. Özyineli algoritmada kullanılan veri yapısı olan ışın ağacı ikili bir ağaç şeklindedir.



Bakış Noktası



Şekil 5.14 : Kesişim noktalarında diğer ışıklardan özyineli oluşturulan ışınlar ve kullanılan ışın ağacı

Işın ağaçlarının tutulması ve işlenmesi için gerekli olan yer ve zamanı en etkin hale getirmek için ilk teknikler 1989 yılında geliştirilmiştir.

Basit bir özyineli ışın izleme algoritması aşağıda verilmektedir (Foley ve ark., 1997) :

```

    Görüntü düzlemindeki pencerenin ve izdüşüm merkezinin seçilmesi
    for (resimdeki her tarama hattı için) {
        for (tarama hattındaki her piksel için) {
            izdüşüm merkezinden çıkarak pikselden geçen ışını belirle
            piksel = RT_TRACE(ray,1);
        }
    }

```

*Işın nesnelerle kesiştir ve en yakındaki kesişimin rengini belirle
depth, ışın ağacındaki anlık derinliktir.*

```
RT_color RT_trace(RT_ray ray, int depth)
{
    ıının bir nesne ile en yakın kesişimini belirle
    if(kesişen nesne varsa) {
        kesişimin yüzey normalini hesapla
        return RT_shade(en yakın nesne, ıın, kesişim, normal, derinlik);
    } else
        return zemin_rengi_değeri;
} /* RT_trace */
```

Nesne üzerindeki noktanın rengini belirle (gölgelendirme, yansıma ve kırılma ışınlarını dikkate alarak)

```
RT_color RT_shade(
    RT_object object; /* Kesişen nesne */
    RT_ray ray; /* Gelen ışın */
    RT_point point; /* Kesişim noktası */
    RT_normal normal; /* Noktadaki yüzeyin normali */
    int depth; /* Işın ağacının derinliği */)
{
    RT_color color; /* Işının rengi */
    RT_ray rRay, tRay, sRay; /* Yansıyan ışın, kırılan ışın ve gölge ışını */
    RT_color rColor, tColor; /* Yansıyan ve kırılan ışın renkleri */
    color = genel_aydınlık_değeri;
    for(her ışık) {
        sRay = noktadan ışığa giden ışın;
        if(normalin ve ışığa giden ışının noktasal çarpımı pozitifse) {
            Geçirgen olan veya olmayan yüzeyler tarafından tutulan ışık miktarını
            hesapla ve renge katmadan önce yayılan ve yönlü terimleri ölçeklendirmek
            için kullan; }
    }
    if(depth < maxDepth)
        if(nesne yansıtıcı ise) {
            rRay = noktadan yansıma yönüne ışın;
            rColor = RT_trace(rRay, depth+1);
            rColor'ı yönlü katsayı ile ölçeklendir ve renge ekle;
        }
    if(nesne geçirgense) {
        tRay = noktadan kırılma yönüne ışın;
        tColor = RT_trace(tRay, depth+1);
        tColor'ı geçirgenlik katsayısı ile ölçeklendir ve renge kat;
    }
    return color;
} /* RT_shade */
```

5.5.3 Işıma (Radiosity)

Işıma probleminde, ışıma değerlerini (B) bulan ilk algoritmalar, doğrudan çözüm tekniklerini kullanmakta idi. Fakat, zaman içerisinde, daha etkin olacağı anlaşılmış olan yinelemeli (iterative) yöntemler geliştirilmiştir.

İlk ışıma uygulamalarında kullanılan tekniklerden birisi Gauss-Seidel yineleme tekniğidir. Bu teknik, “gathering” olarak da adlandırılır. Çünkü, bir yüzey parçasının yeni ışıma değerini hesaplamak, tüm diğer parçaların katılımının toplamını hesaplamaya denktir (Greiner ve ark., 1993).

Işıma probleminin çözümünde geliştirilen tekniklerden birisi de Aşamalı Arındırma tekniğidir. Bu tür bir algoritma, ışıma değerini, bir yüzeyden tüm ortama dağıtır. Ortamdaki tüm parçaları güncellediğinden dolayı, algoritmanın başlarındaki görsel yakınsama daha hızlıdır (Greiner ve ark., 1993).

Ortamdaki düzlemsel yüzey parçaları, birbirlerini etkileme durumlarına göre, sistemin alt kümeleri olarak gruplandırılırlar. Önce, her grup içerisindeki parçaların aralarındaki etkileşim değerleri hesaplanır. Sonra, grupların birbirleri ile etkileşimleri bulunarak aydınlatma gerçekleştirilir. Böylece, sistem parçalara ayrılarak daha basit alt kümelere bölünmekte ve aydınlatma daha basit matematiksel ifadelerle indirgenerek hesaplanabilmektedir. Bu sırada, ortamdaki aydınlatmaya etkisi az olacak parçaların da ihmal edilmesi ile, çok sayıda gereksiz işlemin önüne geçilerek hız artırılır. Bloklayarak Arındırma (Blockwise Refinement) olarak anılan bu yöntemde, başlangıç aşamasında kabul edilebilir bir tahminleme yapılabilmesi için, o ana kadar ele alınmayan kalan parçaların değerleri de yaklaşık olarak hesaplamaya katılır (Greiner ve ark., 1993).

Işıma algoritmalarında, vurucu algoritmalar (shooting algorithms) işletildikleri ilk anlarda iyi sonuçlar verirken, artırmaya (gathering) dayalı yöntemler, uzun vadede daha iyi sonuçlara ulaşılmasını sağlarlar (Greiner ve ark., 1993).



6. BİLGİSAYAR GRAFİKLERİNDE GEOMETRİK PROBLEMLER

Bilgisayar grafikleri, 1950'lerde ortaya çıkışından bu yana piyasa baskısı ve bilgisayar teknolojisindeki ilerlemeler nedeniyle çok hızlı gelişmiştir. Buna karşın bilişimsel geometrinin tarihi çok eskilere dayanmaktadır. Bilişimsel geometrinin temellerini Öklit'in (Euclid) yapıları oluşturmaktadır.

Bilgisayar grafiklerinde sahnelerin modellenmesi ve kaplanması (rendering), geometrik problemler içeren iki önemli alandır. Modelleme, sahnelerin tanımlarının oluşturulmasında kullanılır ve doğası değişkenlik gösterir. Örneğin kare, daire, küp, silindir, küre gibi temel şekillerden veya bulut, ağaç gibi doğal nesnelerden, yazılardan, sayı yığınlarından veya çok değişik bileşenlerden oluşabilir. Kaplama ise nesne tanımını resme veya animasyona dönüştürmekte kullanılır.

Bilgisayar grafiklerindeki temel geometrik problemlerden ikisi kırpma (clipping) ve gizli yüzeylerin elenmesidir (hidden-surface removal) :

- *Kırpma* : Geometrik bir nesnenin görüntülenecek pencere dışında kalan kısımlarının belirlenerek ortadan kaldırılması.
- *Gizli Yüzeylerin Elenmesi* : Bir bakış noktasına göre arka yüzeylerin veya diğer nesnelerin arkasında kalan nesnelerin, yüzeylerin veya yüzey parçalarının belirlenerek elenmesi.

Bilişimsel geometri, geometrik problemlerin çözümünde kullanılan algoritmaları içerir. Karmaşık geometrik problemlerden önce basit geometrik nesneleri içeren (düzlemde nokta, doğru, çokgen ve daire; uzayda nokta, doğru, üçgen) ve formül şeklinde ifade edilmesi kolay problemlerin incelenmesi, konuların anlaşılabilirliğini artırmaktadır. Tez metninde Bölüm 4'te bilgisayar grafiklerindeki basit geometrik

problemlerin çözümünde kullanılan algoritmik yöntemler de incelenmiştir. Bu tür problemlerin bazıları :

- Çokgenleri kendini oluşturan üçgenlere ayırtmak (triangulation)
- Sonlu noktalar kümesi içinde gizli şekilleri bulmak (çokgenler, dışbükey kabuklar)
- Değişik geometrik nesnelerin kesişiminden oluşan nesneleri bulmak.
- Düzlemde belirli şartları taşıyan geometrik nesneleri aramak.

Bilgisayar grafikleri ile bilişimsel geometri arasındaki bağlantı, her ikisinin de geometrik nesneler içermesi ile sınırlı değildir. Bazı yöntemler tümüyle bir alana veya diğer alana ait olmakla birlikte birçok yöntem her ikisi tarafından da kullanılır. Ayrıca bilişimsel geometrideki belirli yöntemler, bilgisayar grafiklerinde ortaya çıkan problemlerin çözümünde etkin bir şekilde uygulanırlar. “Görünmeyen yüzeylerin elenmesi” buna bir örnektir. Bilgisayar grafiklerinin merkezindeki bir problem olmakla birlikte son yıllarda bilişimsel geometri araştırmacılarının da üzerinde yoğun olarak çalıştıkları ve etkin çözümler ürettikleri bir problemidir.

Bilgisayar grafiklerinde temel hesaplama işlemleri, kaplama yani fiziksel nesnelerin gerçekçi resimlerini oluşturma sırasında yapılmaktadır. Bu konu tez çalışmasının 5. bölümünde ayrıntılı olarak anlatılmıştır. Bu işlemler, aydınlatmanın simülasyonunu yaparak gerçek resmin görüntüsünü tahminleme şeklinde gerçekleştirilmektedir. İşlem sırasında, bilişimsel geometride olduğu gibi nesnelerin şekilleri ve aralarındaki uzaysal ilişkiler dikkate alınır. Bilgisayar grafiklerinde ayrıca yansıma, kırılma, geçirme ve ışın yayma gibi, madde ve ışık özelliklerine bağlı özelliklerin kullanılmasıyla boyama yapılacaktır. Yani grafik algoritmaları radyometriyi de modellemelidir. Diğer taraftan radyometri, ışık kaynaklarının sayısal gösterimleri ve yaydıkları

(yoğunluk, dalga boyu ve faz gibi özelliklerine bağlı olarak) elektromanyetik radyasyon ile ilgilenen alandır. Yani grafik algoritmaları psikofiziği de içermelidir.

Kaplama işlemi iki alt bölüme ayrılarak gerçekleştirilir :

- Her bir yüzeyin görünürlük durumunun belirlenmesi
- Görünen yüzeylerin renklendirilmesi. Başka bir deyişle gözlemci tarafından görünecek radyometrik değerlerin belirlenmesi

6.1 Bilgisayar Grafiklerinde Geometrik İşlemler

Bilgisayar grafiklerinde üç temel işlem aşaması vardır :

- *Bilgi toplama (acquisition)* : Simülasyon verisinin toplanması
- *Gösterim (representation)* : Verinin, niteliklerin ve enerji kaynaklarının temsili
- *Simülasyon (simulation)* : Görünüm veya davranışın tahminlenmesi şeklinde gerçekleştirilen benzeşim. Temel bir simülasyon işlemi kaplamadır. Kaplama, görünürlük ve gölgelendirme olmak üzere iki bileşenden oluşmaktadır.

6.1.1 Modelleme, Bilgi Toplama ve Basitleştirme

Grafik algoritmaları girdiye ihtiyaç duymaktadır. Gerçekçi görüntüler de büyük geometrik ve radyometrik modellerin oluşturulmasını gerektirmektedir. Modeli oluşturma problemi, bilgisayar grafiklerinde temel geometrik problemlerdendir. Modeller;

- Bir tasarımcı (insan) tarafından oluşturulabilir (CAD paketi kullanılarak),
- Prosedürel olarak üretilebilir (özyineli kurallar uygulanarak),

- Görüntü verisinin makine destekli olarak işlenmesi ile yapılandırılabilir (birçok fotoğraf yardımı ile topografik arazinin üç boyutlu haritasının oluşturulması)
- Makine algılama yöntemleri ile oluşturulabilir.

Büyük ölçekli geometrik modellerin üretiminde tümüyle otomatik yöntemler (insan desteği olmayan) daha başlangıç aşamasındadırlar. Büyük veri setlerinde, oluşturulan modellerin depolanmasında ve görüntülenmesinde kullanılan algoritmaların basitleştirilmesi ve etkinleştirilmesi, üzerinde çalışılan konulardandır.

6.1.2 Gösterim

Bir grafik işlem aşaması olarak modellerin geometri, ışık ve kuvvet bileşenlerinin bilgisayarda temsil edilmesi gerekmektedir.

Geometrik verilerin temsilinde içsel ve dışsal veri yapıları (quadtree, octree, BSP tree, k-d tree, B-rep, CSG) kullanılmaktadır. Bunun dışında fiziksel kuvvetler ve ışıma enerjisi de modellenmelidir. Bilgisayar grafikleri araştırmacıları, her bir yüzey için ışıma enerjisi etkileşimini ve fiziksel kuvvetlerin etkilerini, sayısal tanımlarını kullanarak temsil ederler. Değişik türlerde nesneler vardır :

- a) İnsan yapısı nesneler : makine parçaları, mobilyalar, binalar gibi.
- b) Organik nesneler : bitkiler ve hayvanlar gibi.
- c) Doğada bulunan nesneler : Moleküller, yeryüzü şekilleri ve galaksiler gibi.

6.1.3 Simülasyon

Bilgisayar grafiklerinde davranışları tahminleme yolu ile çok çeşitli simülasyon işlemleri gerçekleştirilmektedir. Örneğin atılan iki zarın simülasyonu ve çarpışma durumunun belirlenmesi, sanal nesneler

üzerinde çalışan bir araştırmacı için mekanik cihazların sürtünme kuvvetlerinin simülasyonu gibi. Fiziksel tabanlı bu tür simülasyonlar birçok grafik uygulamasında kullanılır.

6.2 Görünen Yüzeyleri Belirleme Yöntemleri

Bu bölümde kaplama işleminin adımlarından biri olan görünen yüzeyleri belirlemeyle ilgili olarak geliştirilen bazı yöntemler üzerinde durulacaktır. Bir bakış noktasından (kameranın veya gözün yeri) belirli bir yöne doğru bakıldığında nesnelerin bakış noktasına göre arka yüzeyleri görülmeyecektir. Aynı zamanda diğer nesnelerin arkasında kalan nesneler, yüzeyler ve yüzey parçaları da görünmeyecektir. Bu yüzeylerin belirlenmesi, gizli yüzeylerin elenmesini veya görünen yüzeylerin belirlenmesini gerektirir. Bu gereklilik, gerçekçi grafik görüntülerinin oluşturulmasında önemli bir konudur. Görünen yüzeylerin belirlenmesi amacıyla, değişik türlerde uygulamalar için etkin yöntemler geliştirilmiştir. Bu yöntemlerin yer ve zaman karmaşıklıkları yani bellek gereksinimleri ve işletim süreleri değişkenlik göstermektedir. Seçilecek yöntem sahnenin karmaşıklığına, görüntülenecek yüzeylerin türüne, eldeki donanıma ve görüntünün hareketli olup olmaması gibi birtakım faktörlere bağlıdır.

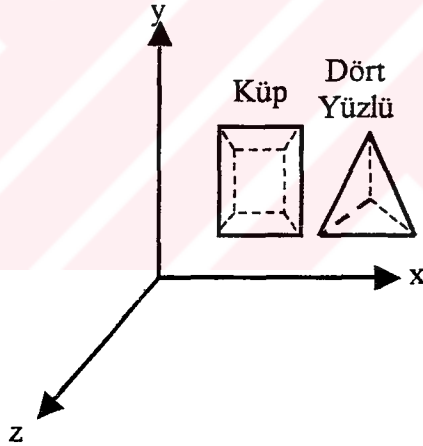
Görünen yüzeyleri belirlemek için kullanılan başlıca iki yaklaşım vardır :

- **Nesne Uzayı Yöntemleri (object-space methods)** : Nesne uzayı algoritmaları, nesnelerin görünürliğini, yüzeyleri diğer nesne ve yüzeylerle karşılaştırarak tanımlandıkları uzayda belirler.
- **Görüntü Uzayı Yöntemleri (image-space methods)** : Görüntü uzayı algoritmaları, nesnelerin ve yüzeylerin görünürliğini piksel piksel izdüşüm düzleminde belirler.

Görünen yüzeylerin belirlenmesine ilişkin algoritmaların çoğu görüntü uzayı yöntemlerini kullanır.

6.2.1 Arka Yüzeylerin Belirlenmesi

Üç boyutlu nesnelerin arkalarında kalan yüzeyler görünmeyecektir. Arka yüzeylerin belirlenmesi kolaydır. Yüzey normalinin z bileşeni negatif olan (sağ el koordinat sistemine göre) arka yüzeydir. Genel yüzey denkleminde $(Ax+By+Cz+D=0)$, herhangi bir yüzeyin normali bulunabilmektedir : $N=(A,B,C)$. $C \leq 0$ ise arka yüzeydir. Şekil 6.1’de küp ve dört yüzlünün sadece ön yüzeyleri görünmektedir. Küpün ön yüzeyinin normali $N=(0,0,1)$ olarak belirlenmiştir.



Şekil 6.1 : Arka yüzeylerin belirlenmesi.

6.2.2 Arkada Kalan Nesne Parçalarını Belirleme Yöntemleri

Diğer nesne parçalarının arkasında kalan yüzeyleri belirlemede kullanılan yöntemlerin en önemlileri; derinlik arabelleği (depth-buffer, z -

buffer) yöntemi, A-arabelleği (A-buffer) yöntemi, tarama hattı (scan line) yöntemi, derinlik sıralaması (depth sort) yöntemi, BSP-ağacı (BSP tree) yöntemi, Alan bölümlleme (area subdivision) yöntemi ve sekizli ağaç (octree) yöntemleridir. Bu yöntemler için daha ayrıntılı bilgiler temel bilgisayar grafikleri kitaplarında yer almaktadır (Hearn ve Baker, 1997) (Foley ve ark., 1996). Burada sadece, çok yaygın olan ve tezin uygulama kısmında da kullanılan derinlik sıralaması yöntemi üzerinde durulacaktır.

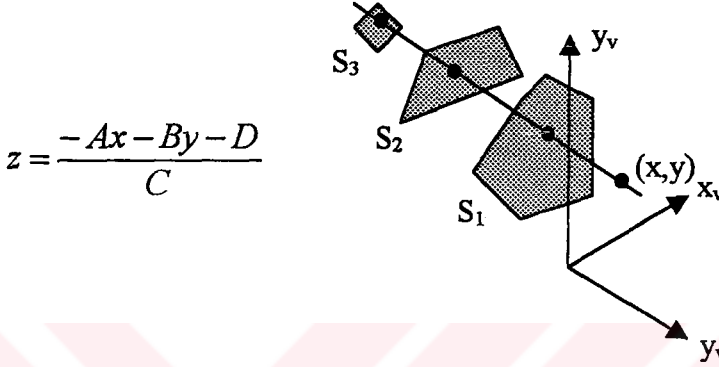
Derinlik Sıralaması Yöntemi

Derinlik sıralaması yöntemi hem görüntü uzayında hem de nesne uzayında işlemlerin yapılmasını gerektirmektedir. Yöntemin mantığı kısaca şu temel fonksiyonlardan oluşmaktadır :

Sahneyi oluşturan yüzeyler derinliklerine göre azalan sırada sıralanırlar. Nesneler, en derinden başlanan tarama işleminden geçirilerek görüntülenir. Sıralama işlemleri hem görüntü hem de nesne uzayında yapılmaktadır.

Görünmeyen yüzeylerin belirlenmesinde kullanılan bu yöntem ressamın algoritması (painter's algorithm) olarak da adlandırılır. Yönteme bu ismin verilmesinin nedeni yağlıboya ressamlarının zemin renklerini çizdikten sonra, nesneleri en geriden başlayarak öne doğru resme eklemesine benzemesinden dolayıdır. Her katman yani çizilen her yeni nesne, üzerine geldiği nesneleri veya katmanı kaplar. Derinlik sıralaması yönteminde de en uzaktaki yüzeyin yoğunluk değerleri yenilenen arabelleğe yerleştirilir. Daha sonra sırayla öndeki nesnelerin yüzeylerine doğru gelinerek yoğunluk değerleri yazılır. Diğer nesnelerin üzerine gelen yüzeyler arkadaki nesnelerin yoğunluk değerlerini ortadan kaldırır.

Yüzey denklemi : $Ax + By + Cz + D = 0$ ise, sahnedeki her bir yüzey için (x,y) yüzey noktasındaki derinlik değerleri, düzlem denkleminde Şekil 6.2'deki gibi hesaplanır.



Şekil 6.2 : Yüzey noktalarının derinlik değerlerinin hesaplanması. (x,y) görüntü düzlemi noktasında S_1 , görüntü düzlemine en yakın yani en az derinlikteki yüzeydir. En son çizileceğinden dolayı diğer nesnelerin üzerinde yer alır.

Derinlik sıralaması yönteminde derinlikleri çıkan nesneler için yapılması gerekli olan ek işlemler değişik grafik kitaplarında yer almaktadır (Hearn ve Baker, 1997) (Foley ve ark., 1996).

6.3 Bilgisayar Grafiklerinde Çözüm Bekleyen Problemler

Bilgisayar grafiklerinde bilişimsel geometri yardımı ile çözülebilecek geometrik problemlerden bazıları aşağıda belirtilmiştir :

- Bilgisayar grafiklerinde yapılmış olan çalışmaların çoğu statik ortamlar için düşünülmüştür. Belirli bir bakış açısından bakıldığında gerçekleştirilmesi gereken aydınlanmada olduğu gibi nesneler hareket etmiyor veya şekil değiştirmiyordu. Çözüm bekleyen problemler günlük yaşamda birçok durumda ve bilim alanında karşılaşılan

hareketli nesnelerin yani dinamik ortamların modellenmesi ile ilgilidir. Verilerin toplanması, modellenmesi ve simülasyonu için kullanılan geometrik algoritmaların dinamik ortamlar için ne şekilde değiştirileceği, üzerinde çalışılmakta olan problemlerdendir. Her bir probleme özel çözümlerle, tüm problemler için genel çözümler ayrı ayrı düşünülmektedir. Zamanla değişen modeller ve çarpışma modelleri dinamik ortamlara örnektir. Üzerlerinde etkileşimli kontrol bulunması gereken uydular gibi nesneler için bilişimsel geometri kapsamında uygun algoritmaların ve veri yapılarının geliştirilmesi örnek olarak verilebilir. Bu konuda algoritmaların paralelleştirilmesi de önemli yer tutmaktadır.

- Bilgisayar grafiklerindeki ters işlemler de (örneğin; belirli bir görünümü elde edecek şekilde ışık kaynaklarının ayarlanması), üzerinde çalışılmakta olan konulardandır.
- Bilişimsel geometri genelde verilerin birincil bellekte yani RAM üzerinde olduğunu ve maliyetsiz veya birim maliyetle erişildiğini varsayar. Oysa uygulamada böyle değildir ve modellerde büyük veritabanları vardır. Birincil ve ikincil bellek ortamlarında verilerin gidip gelmesi bilişimsel geometri için ilginç bir çalışma alanıdır. Özellikle sanal gerçeklik ortamlarında dışsal bellek algoritmaları üzerinde çalışılmaktadır. Sanal ortamlardaki hareketler genelde düzgün olduğundan, gerçek zamanlı olma kısıtlaması hareketlerin tahminlenmesini gerektirir.

7. GRAFİK YAZILIMLARI GELİŞTİRMEDE İKİ VE ÜÇ BOYUTLU UYGULAMA GELİŞTİRME ARAYÜZLERİ

7.1 Giriş

Üç boyutlu bilgisayar grafikleri, yazılım ve donanımlardaki gelişmelere paralel olarak günlük yaşamda kendisini her geçen gün daha fazla hissettirmektedir. Özellikle internet üzerine üç boyutlu grafiksel içerik ekleme, son günlerde üzerinde çalışılan konulardandır. Internet2 ve NGI (Next Generation Internet) gibi ileri ağ teknolojilerini oluşturmaya yönelik projelerden elde edilecek başarılarla, daha yüksek bant genişliği, servis kalitesi sağlanacak ve çok kullanıcı, etkileşimli, gerçek zamanlı uygulamaların beklediği ortam oluşacaktır. Sonuçta üç boyutlu grafiksel içerikler oluşturmayı ve üzerinde işlemler yapmayı sağlayan yazılımların ve yazılım geliştirme ortamlarının önemi daha da artacaktır.

Bu bölümde üç boyutlu grafik yazılımları geliştirmede iki ve üç boyutlu uygulama geliştirme arayüzlerinin kullanımları anlatılmış ve karşılaştırılmaları yapılmıştır.

7.2 Grafik Yazılımları

Grafik yazılımları ikiye ayrılır : Uygulama yazılımları ve uygulama geliştirme arayüzleri (API). Uygulama yazılımları arasında, çizim, boyama, modelleme, animasyon, ışın izleme, CAD programları ve diğer grafik paket programları vardır. Uygulama geliştirme arayüzleri ise, bilgisayarda grafikler oluşturmaya yönelik olarak kullanılan sınıfları, veri yapılarını, değişkenleri, metotları, komutları ve diğer yapıları içeren yazılımlardır. Uygulama geliştirme arayüzlerine de Core, PHIGS, OpenGL ve Java 3D örnek olarak verilebilir. Grafik yazılımları çok farklı donanım ve yazılım ortamlarında çalışabilmektedir.

7.2.1 Grafik Uygulama Yazılımları

Donanımlardaki gelişmelerle birlikte internet üzerinden de yüklenebilen shareware/freeware binlerce grafik yazılımı üretilmiştir. Kişisel bilgisayar ortamları dışında grafik iş istasyonlarında kullanılan çok daha ileri yazılımlar da bulunmaktadır.

Grafik uygulama yazılımları günümüzde oldukça ucuzlamıştır. Bu yazılımlar yardımıyla çok karmaşık grafik ve animasyonlar kolaylıkla elde edilebilmektedir. Bu yazılımlar basit bir eğitim verilen ve matematik veya grafik konusunda çok bilgisi olmayan sıradan bilgisayar kullanıcıları tarafından da rahatlıkla kullanılabilir. Üç boyutlu grafik uygulama yazılımları, bu tür yazılım üreten şirketlerin çoğalması ile giderek artmakta ve yaygınlaşmaktadır. Bu tür programlar arasında AutoCAD, LightWave 3D, Corel Draw, PhotoShop, Macromedia Flash, Crystal 3D Impact, PaintShopPro, PovRay sayılabilir.

7.2.2 Grafik Uygulama Yazılımlarında Kullanılan Programlama Dilleri

Piyasadaki başarılı uygulama yazılımlarının çoğunun geçmişi 1970'lere ve 1980'lere dayanır. Bunlardan birçoğu, Fortran, Pascal, C gibi prosedürel programlama dilleri kullanılarak yazılmışlardır ve tasarımları da bir tür yukarıdan aşağıya ayrıştırma yaklaşımına göre yapılmıştır. Sonuçta, geliştirilmesi ve bakımı zor olan ürünler ortaya çıkmıştır. Yazılım parçaları, "yeniden kullanılabilirlik" özelliğini yeterince taşımadığı için, yeni bir duruma göre programların uyarlanması, programı yeniden yazma ya da program kodunu kopyalama şeklinde olmuştur. Sonuçta grafik yazılımlarının geliştirilme süresi uzamış, kod tekrarı olmuş ve birçok şirket birçok fırsatlar kaçırmıştır.

Nesneye yönelik yaklaşımın ortaya çıkması ile, değişikliklere kolayca uyarlanabilen, genişletilebilen uygulamaların geliştirilmesi mümkün olmuştur. Nesneye yönelik yaklaşımın en önemli üstünlükleri şu şekilde belirtilebilir :

- Sistemin anlaşılması kolaydır, sistemle gerçek arasındaki fark azdır.
- Model üzerinde yapılacak olan değişiklikler yerel olarak yapılabilir.
- Yeniden kullanılabilir kod yazılabilir.

Bilgisayar grafiklerindeki uygulamaları geliştirme konusunda Pascal programlama dilinden sonra, etkinliği, hızı, taşınabilirliği ve diğer üstün özellikleri ile C popüler hale gelmiş ve tercih edilir olmuştur. Nesneye yönelik yaklaşımla birlikte, bilgisayar grafikleri içeren uygulamaların yazılmasında da nesneye yönelik programlama dilleri tercih edilmeye başlanmıştır. Bu sırada C++ grafik programlamada önem kazanmıştır.

Geleneksel C ve C++ programlama dilleri karakter tipinde girdi/çıkı uyarlıydı. C++'ın, platform bağımlı sınıf kütüphaneleri tarafından desteklenen bazı sürümleri grafik üretebilmekteydi. Fakat bu kütüphanelerin kullanımları, uygulamaları taşınabilir olmaktan çıkarmıyordu. Resmi olarak 1995 yılında ortaya çıkan Java'nın grafik yeteneklerini kullanmak üzere geliştirilen programlar ise taşınabilir özelliğe sahiptir. Ayrıca Java, grafiksel giriş/çıkış işlemlerini de desteklemektedir. Geliştirilen yoğun olarak grafiklerle yüklü Java appletleri web üzerinden dünyanın herhangi bir yerindeki yerel java ortamlarından erişilerek çalıştırabilirler.

7.2.3 Uygulama Geliştirme Arayüzleri

İki ve üç boyutlu uygulama geliştirme arayüzleri, yazılım geliştiricilerin, yazılımlarına iki ve üç boyutlu kaliteli grafik ve

animasyonlar ekleyerek kullanabilmelerini ve geliştirilen uygulama programlarının taşınabilirliğini sağlar. Bu arayüzler kullanılarak grafik uygulama yazılımları geliştirilebileceği gibi sıradan grafik programcılar grafiksel öğeler içeren kendi programlarını ve uygulamalarını da yazabilirler. İki boyutlu uygulama geliştirme arayüzleri uzun yıllardır kişisel bilgisayar ortamlarında yaygın olarak kullanılmaktadır. Ancak üç boyutlu grafik uygulama geliştirme arayüzlerinin kullanımı donanım ve yazılımsal gelişmelerle birlikte kişisel bilgisayar ortamlarında yakın zamanda yaygınlaşmaya başlamıştır. Bu tür uygulama geliştirme arayüzlerinin gelişmesi ile derin matematik ve grafik altyapısına sahip olmayan programcılar bile gereksinim duydukları grafikleri, yazılımlarına ekleyebileceklerdir.

Örneğin bir bilim adamının veya öğretmenin, gezegenlerin hareketlerini ve gezegenlere ilişkin bilgileri içeren bir program yapması ve internet üzerine koyması birçok kişinin ve öğrencinin bu sayfaya erişip güneş sistemi hakkında daha ayrıntılı ve görsel bilgi edinmesini sağlayacaktır. Bu tür amatör amaçlarla ilgili çok gelişmiş bir yazılımın geliştirilmesi beklenmeyecek ve çok büyük maliyetlerle alınması gerekmeyecektir. Kullanıcılar da yapılan esnek bir program yardımı ile örnek olarak programın parametre değerlerini değiştirerek farklı durumlara göre deneyebilecek ve gözlemleyebilecektir. Üç boyutlu uygulama geliştirme arayüzleri desen eşleme işlemini rahatlıkla gerçekleştirebildiğinden, gezegen yüzeylerinin resimleri gezegen modellerinin üzerine kaplanarak daha gerçekçi görüntüler oluşacaktır.

7.3 İki Boyutlu Uygulama Geliştirme Arayüzleri

Grafik yazılımları geliştirmede iki boyutlu grafik uygulama geliştirme arayüzleri uzun yıllardır kullanılmaktadır. İki boyutlu grafikler içeren yazılımların geliştirilmesinde kullanılmak üzere yeteri derecede

işlev ve özellik içermektedirler. Ayrıca üç boyutlu grafikler içeren yazılımlarının geliştirilmesinde de kullanılmaktadırlar.

Kişisel bilgisayar ortamlarında DOS işletim sistemi ortamında Pascal, C, C++ programları ile kullanılmış olan BGI (Borland Graphics Interface) ve Windows ortamındaki Windows API'leri birçok grafik programcısının bildiği arayüzlerdir. Günümüzde Java awt ve Java 2D API de yaygın olarak kullanılmaya başlanmıştır. Java programlama dilinin ağ merkezli olmasının getirdiği avantajlardan da yararlanılmaktadır. Geliştirilip internet üzerine yerleştirilen grafiksel içerikli bir applet, dünyanın farklı bölgelerindeki kullanıcılar tarafından erişilerek yerel makinelerden işletilebilmektedir.

Java 2D, Java 2 ortamı ile birlikte gelen bir API yani uygulama geliştirme arayüzüdür. Java 2'den sonraki Java sürümlerinde de bu iki boyutlu uygulama geliştirme arayüzü korunmaktadır. Java 2D, kaliteli ve ileri düzeyli iki boyutlu grafikler oluşturmayı kolaylaştıran bir dizi sınıfı ve özelliği içermektedir. Bu özellikler arasında, temel şekillere ilişkin olanlar dışında geometrik dönüşümler, görüntü işleme, sağa veya sola doğru yazabilme imkanı gibi birçokları sayılabilmektedir (Knudsen, 1999).

Java 2D, java.awt ve java.awt.image paketleri içine yeni sınıflar eklediği gibi, java.awt.color, java.awt.font, java.awt.geom, java.awt.print, java.awt.image.renderable, com.sun.image.codec.jpeg gibi tümüyle yeni paketleri de beraberinde getirmiştir (Knudsen, 1999).

Java 2D, bilgisayar grafikleri oluşturmak için tasarlanmıştır. Java 2 ortamından önceki JDK1.0 ve JDK1.1 sürümlerinde Java 2D yoktur. JDK1.0 ve JDK1.1 sürümleriyle gelen awt (abstract windowing toolkit) pencerelerle ilgili işlemleri, resimleri, çizimleri ve yazı tiplerini içermekteydi fakat birçok özellikten yoksundu. Sun mühendisleri bu kötü

durumu ortadan kaldırıp kaliteli grafikler oluşturmayı kolaylaştıracak grafik araçlarını da içeren JFC'yi (Java Foundation Classes) geliştirmişlerdir. Java 2D, JFC'nin bir parçasıdır. JFC temel bir API olduğundan her Java 2 uygulamasında vardır. Bu aynı zamanda jdk'nın eski sürümlerinde kullanılamaması anlamına da gelmektedir. Java 2D'yi anlayabilmek için awt'yi iyi anlamak gerekir. awt iki parçadan oluşmaktadır : Kullanıcı arayüzü parçası ve çizim parçası. Jdk1.1'den Jdk1.2'ye geçerken kullanıcı arayüzü parçası yerine "Swing", çizim parçası yerine de Java 2D yani 2B API geçmiştir. 2D API'nin geliştirilmesi sırasında Sun firması, Adobe, Ductus, Kodak, Taligent gibi birçok kuruluşla işbirliği yaparak nitelikli bir ürün ortaya çıkarmıştır (Knudsen, 1999).

Günümüzde iki boyutlu grafik uygulama geliştirme arayüzlerinin oldukça genişlemiş olan özelliklerinden bazıları aşağıdaki gibi belirtilebilir :

- **Temel Şekiller** : Temel iki boyutlu şekilleri çizdirmeyi sağlayan komutlar (nokta, çizgi, dikdörtgen, elips, yay, eğri, çokgen vb.) vardır. Diğer geometrik şekiller, temel şekillerin bileşiminden oluşur.
- **Renk** : Değişik amaçlarla hazır renklerin kullanımını, renklerin elde edilmesini ve renk özelliklerinin ayarlanmasını sağlayan renk komutları kullanılır.
- **Resim** : BMP, GIF, JPEG dahil birçok resim formatındaki resmi yüklemeyi ve kaydetmeyi sağlayan komutlar yardımıyla, herhangi bir grafik programı ile veya tarayıcıdan alınarak oluşturulmuş bir resim, grafik programlarında kullanılabilir hale gelmektedir.

- **Çizgi ve Dolgu Biçemi** : Şekilleri biçemlemeyi sağlayan komutlar ve hazır özellikler, nesnelere çeşitlilik katmaktadır.
- **Çizgi biçemi** : Şekillerin sınırları ve çizgiler kesiksiz veya noktalı çizgi türleri ile çizilebildiği gibi herhangi bir kalınlıkta ve renkte de çizilebilir.
- **Dolgu biçemi** : Şekillerin içi sabit bir renkle, desenle, renk basamakları veya herhangi bir resim ile doldurulabilir.
- **Metin** : Yazılar, yazıtipleri ve yazılar üzerinde biçemlemeyi sağlayan komutlar, iki boyutlu grafiklerle, grafiksel yazıların beraber kullanılabilmesini sağlar.
- **Görüntü İşleme** : İki boyutlu gelişmiş arayüzler, resimler üzerinde hazır görüntü işleme komutlarının uygulanabilmesini sağlar.
- **Dönüşüm** : Şekiller, resimler, yazılar küçültülüp, büyütülebilmekte, taşınabilmekte ve döndürülebilmektedir.
- **Kırpma** : Nesnenin bir pencere içinde kalan alanları dışındaki kısımlarını ortadan kaldırma işlemi kolaylıkla yapılabilir.
- **Kıraksızlaştırma “Antialiasing”** : Çizimlerdeki basamaklı kenarları düzgünleştirme tekniğidir.
- **Saydamlık** : Nesnelere geçirgenlik verilmesi üst üste gelen resim parçalarında daha gerçekçi görüntülerin oluşturulmasını sağlar.
- **Yazıcı** : Ekrandaki resmin ve şekillerin yazıcı çıktısının alınmasını kolaylaştıran komut ve özellikler önemlidir.

Bu uygulama geliştirme arayüzleri, kullanıldıkları programlama dilinin tüm özelliklerine grafik özellikleri de katmayı sağlamışlardır. Ayrıca donanım ve yazılım sisteminin ve dilin grafik kullanıcı arayüzü özellikleri ile etkin pencereler ve pencere elemanları oluşturulabilmekte, fare ve klavye gibi cihazlarla etkin bir etkileşim gerçekleştirilebilmektedir.

Bu bölümde üç boyutlu nesnelerin iki boyutlu uygulama geliştirme arayüzleri ile bilgisayarda oluşturulup görüntülenmesinde kullanılan yöntemler üzerinde durulacaktır.

7.3.1 Üç Boyutlu Nesnelerin Bilgisayarda Temsili

Günümüzde bilgisayar donanım ve yazılımlarının gelişmesi ve bilgisayar kullanıcılarının sayısının artması ile üç boyutlu bilgisayar grafikleri de daha çok kullanılır olmuştur. Her geçen gün e-ticaret alanından eğitim alanına, sanal gerçeklikten bilimsel görselleştirmeye kadar birçok alanda üç boyutlu bilgisayar grafiklerinin kullanımı giderek yaygınlaşmaktadır. Kişisel bilgisayar ortamında bu tür programların yanında özellikle bilgisayar oyunları da üç boyut teknolojisinin ve donanımlarının geliştirilmesinde itici güç haline gelmiştir.

Grafiksel sahnelerde binalar, insanlar, taşıtlar, eşyalar, ağaçlar, çiçekler, hayvanlar, bulutlar, taşlar, tuğlalar, tahtalar, plastikler, kağıtlar, mermerler, çelikler, camlar, kumaşlar, lastikler, sular ve diğer sıvılar gibi değişik birçok nesne bulunabilir (Hearn ve Baker, 1997). Farklı madde özelliklerine sahip nesneleri bilgisayarda tanımlamak için birden çok gösterim şekli vardır. Daha gerçekçi görüntüler oluşturmak için bu nesne özelliklerini doğru bir şekilde modellemek gerekir. Çokgenler ve ikinci dereceden yüzeyler, çokyüzlülerin ve elipsoitlerin oluşturulmasında kullanılırlar ve kesin tanımları vardır. “Spline” yüzeyleri ve yapılandırma teknikleri uçakların kanat ve motorlarının ve eğrisel yüzeyler içeren diğer

mühendislik yapılarının tasarımında kullanılırlar. Fraktal yapıları ve madde sistemleri, bulut ve ot gibi doğal nesnelerin doğru bir şekilde temsil edilebilmesini sağlarlar. Etkiyen kuvvet sistemlerini kullanan fiziksel tabanlı modelleme yöntemleri, bir kumaş parçasının veya bir pelte küresinin katı olmayan hareketini tanımlar. "Octree" kodlamaları, nesnelerin tıbbi görüntülerde olduğu gibi içsel görüntülerinin gösteriminde kullanılır. Bunlar dışında birçok üç boyutlu görüntüleme tekniği vardır.

Katı nesnelerin gösterim düzenleri iki çeşittir: sınır gösterimleri (boundary representations) ve uzay bölümlleme gösterimleri. Sınır gösterimi, nesneyi, nesnenin içini dışından ayırmayı sağlayan yüzeyler dizisi şeklinde tanımlar. Çokgen yüzeyleri ve "Spline" parçalarını kullanır. Uzay bölümlleme gösterimleri, "Octree" gösteriminde olduğu gibi nesneyi küçük katı parçalarına ayırarak tanımlar. Üç boyutlu değişik nesnelerin tanımlanmasında önemli bazı gösterim yöntemleri şunlardır :

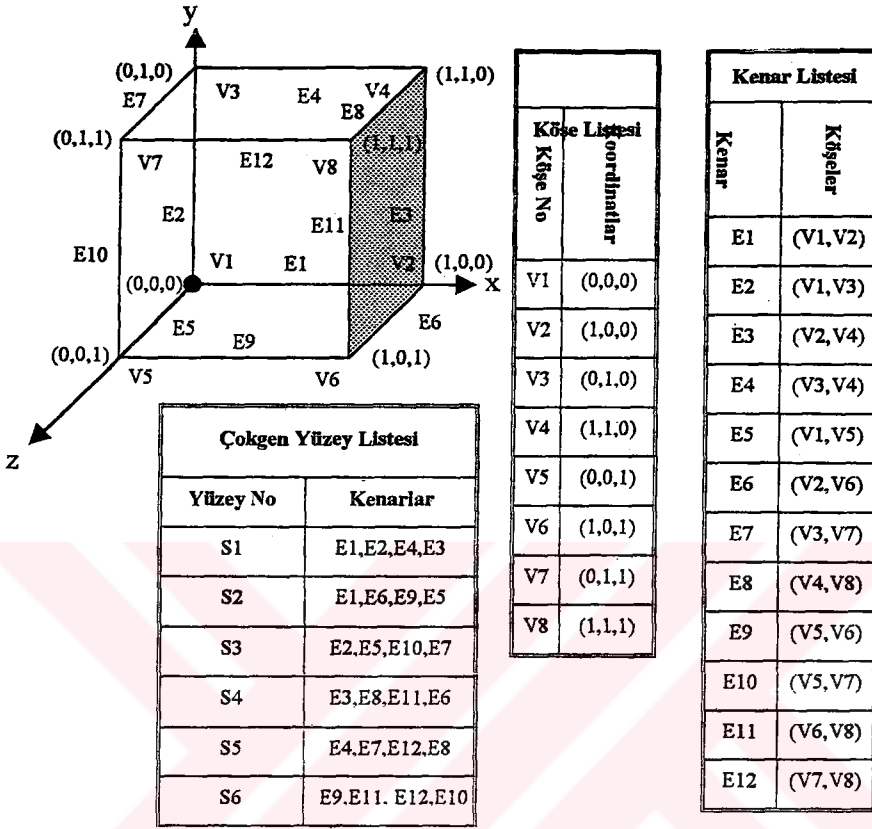
- Çokgen yüzey gösterimleri
- Eğriler ve eğrisel yüzey gösterimleri (Bezier, B-Spline, Beta-Spline, Rational Spline, NURBs)
- "Quadric" (küre, elipsoid, torus...) ve "Superquadrics" (süperelips, süperelipsoid) yüzey gösterimleri
- Organik (blobby) yüzey gösterimleri
- Tarama (sweep) gösterimleri
- Katı geometri yapılandırma gösterimleri (CSG -Constructive Solid Geometry-) (birleşim, kesişim, fark)

Üç boyutlu grafik programları bu tür gösterim şekillerinin bir veya daha fazlasını kullanmaktadırlar. Üç boyutlu görüntülemenin anlatımından önce, izleyen bölümde, üç boyutlu nesnelerin bilgisayarda

tanımlanmasına ilişkin yöntemlerden “çokgen yüzeyli nesnelerin sınır gösterimi” üzerinde durulmuştur. Diğer tür nesnelere ve nesnelerin tanımlanmasına ilişkin yöntemler de ayrıca araştırma konuları olmakla birlikte tezin konusu dışında olduklarından isimleri verilerek geçilmiştir.

7.3.2 Çokgen Yüzeyli Nesnelerin Bilgisayarda Tanımlanması

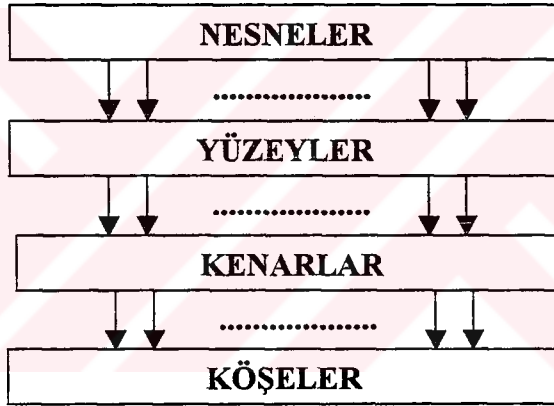
Yüzeyleri çokgen olan üç boyutlu nesneler bilgisayarda köşe, kenar ve yüzey bilgileri ile tanımlanırlar. Bu bilgileri tutmak için köşe listesi, kenar listesi, çokgen yüzey listesi oluşturulur. Bu listelerin oluşturulmasında statik veya dinamik veri yapıları kullanılabilir. Köşe listesi, köşelerle birlikte köşelerin üç boyutlu koordinatlarını içerir. Kenar listesi, kenarları oluşturan köşeleri tutar. Çokgen yüzey listesi ise yüzeyleri oluşturan kenarları tutar. Bir küp şeklinin bilgisayarda tanımlanması için 8 köşenin koordinatlarının, 12 kenarı oluşturan köşelerin ve 6 yüzeyi oluşturan kenarların bilgilerinin tutulması gerekmektedir (Şekil 7.1). Bazı durumlarda ise yüzey bilgileri, kenar listesi kullanılmadan doğrudan köşe bilgilerinden oluşturulur. Bu durumlarda sadece köşe listesi ve çokgen yüzey listesinin tutulması yeterlidir.



Şekil 7.1 : Kenar uzunluğu 1 birim olan bir küpün köşe listesi, kenar listesi ve çokgen yüzey listesi yardımı ile kenar tabanlı sınır temsili.

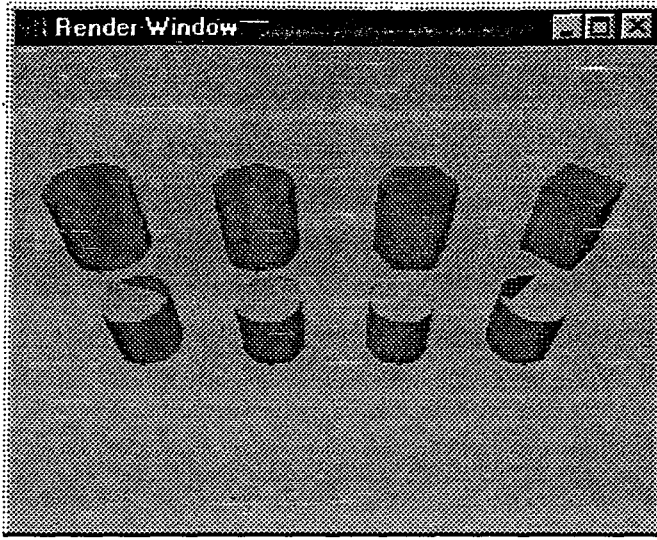
Örnek olarak bu tür listelerle tanımlanmış küp gibi üç boyutlu bir nesne ekranda gösterileceği zaman, tüm yüzeyler dolaşarak, sıra ile her bir yüzey, çokgen çizdiren komutlarla ekrana çizdirilir. Bir yüzeyi oluşturan kenar ve köşe bilgilerine çokgen yüzey listesi yardımı ile ulaşılır. Bu tür bilgilerin kullanılması ile üç boyutlu sahnelerin görüntülenmesi sırasında izlenen adımlar üç boyutlu görüntüleme konusunda incelenecektir.

Bir sahneyi oluşturan üç boyutlu nesnelerin bilgilerini içeren listeler değişik veri yapıları kullanılarak tutulabilirler. Nesneler, yüzeyler, kenarlar ve köşeler ayrı düzeyler olarak düşünülebilir (Şekil 7.2). Her bir nesneden kendisini oluşturan yüzeylere bağlantılar vardır. Yüzeylerden de kendilerini oluşturan kenar ve köşelere bağlantılar vardır. Kenarlardan kendilerini oluşturan iki uç köşeye ulaşılabilir. Her bir düzey ile alt düzeyler arasında bağlar olduğu gibi her bir düzeyin kendi içerisinde de bağlar vardır. Örnek olarak bir sahneyi oluşturan nesneler bağlı listelerle birbirine bağlanabilir.



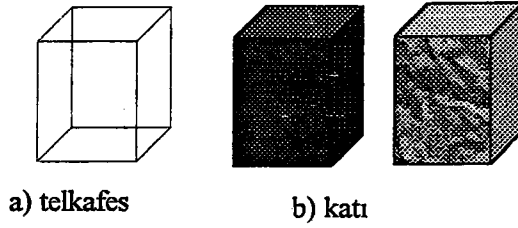
Şekil 7.2 : Çokgen yüzeyli nesne bilgilerini içeren listelerin düzeyleri ve bağlantıları

Eğrisel yüzeyler de çokgensel yüzey yaklaşımı yapılarak çizdirilebilir. Örnek olarak bir silindir, çokgen ağı (polygon-mesh) şeklinde dilim dilim çizilen çokgenler yardımıyla oluşturulabilir (Şekil 7.3). Çokgen dilimlerinin sayısı çoğaltıldıkça, köşeli parçalardan oluşan silindir gerçek bir silindir şekline dönüşecektir (4,6,8,12,36,72,180,360 dilim).



Şekil 7.3 : Çokgensel yüzey yaklaşımı ile, dilim sayısı değiştirilerek oluşturulmuş silindirler

Katı modelleme yapıldığında, nesnelerin özellikleri (yapıldığı madde, yansıtıcılık ve geçirgenlik özellikleri, ...), yüzeylerin özellikleri (renk, doku, resim ..) ve kenarların özellikleri de veri yapılarına uygun şekilde depolanmalıdır. Nesnelerin bu tür özellikleri dikkate alınarak görüntülenmeleri, *katı gösterim* (solid representation) olarak adlandırılır (Şekil 7.4-b). Nesnelerin yüzey yapısının sadece çerçevelerinden oluşan gösterim biçimine *telkafes gösterim* (wireframe representation) adı verilmektedir (Şekil 7.4-a). Telkafes gösterimde sadece kenarlar çizildiğinden ve yüzeyleri oluşturan her bir noktanın renginin hesaplanması ve çizilmesi gerekmediğinden, genelde önizlemelerde nesnelerin genel görüntüsünü veren bu gösterim biçimi kullanılmaktadır. Kenar listesinin tutulması telkafes görüntülerin doğrudan oluşturulmasını sağlar.

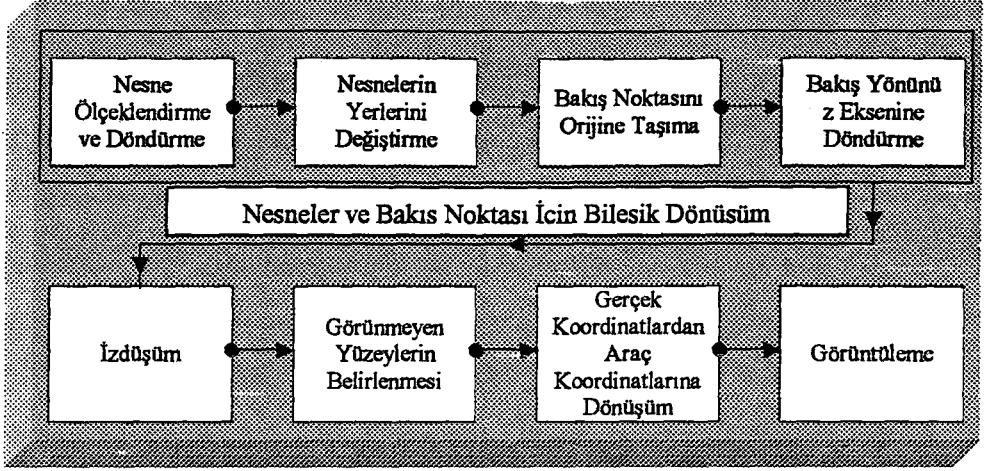


Şekil 7.4 : Telkafes gösterim ve katı gösterim

7.3.3 Üç Boyutlu Nesnelerin Ekranda Görüntülenmesi

Üç boyutlu nesnelerin ekranda görüntülenmesi, iki boyutlu nesnelerin ekranda görüntülenmesine oranla çok daha zordur. Bunun nedeni, görüntü yüzeylerinin iki boyutlu olmasıdır. Üç boyutlu nesnelerin koordinatlarının iki boyutlu koordinatlara dönüştürülmesi gerekmektedir. Üç boyutlu uzayda, bakış noktası ve yönü, arkada kalan nesneler ve nesnelerin arkada kalan yüzeyleri gibi faktörleri de göz önünde bulundurmak gerekmektedir (Hearn ve Baker, 1997) (Watt, 2000). Buna göre üç boyutlu nesneler içeren bir sahnenin görüntülenmesi için izlenen işlem sırasının basitleştirilmiş hali Şekil 7.5'teki gibidir:

- Nesneler, tanımlandıkları başlangıç noktasına (orijine) göre ölçeklendirilir (scaling) ve döndürülürler (rotation). Ölçeklendirme ve döndürme değerleri, nesnelerin uzaydaki büyüklüğünü ve yönünü belirler. Bu adım sonunda nesneler istenilen boyut ve açıya getirilmiş olurlar.
- Nesneler, yer değiştirme (translation) işlemi ile tanımlandıkları uzaydan gerçek uzaydaki konumlarına taşınırlar.



Şekil 7.5 : Üç boyutlu nesnelerin görüntülenmesi sırasında gerçekleştirilen işlemler (View Pipeline)

- Bakış noktasının orijine taşınması ile nesneler, bakış noktasının başlangıç noktası kabul edildiği uzaydaki konumlarına getirilirler. Bu, tüm nesneler için bir yer değiştirme işlemidir.
- Bakış yönünün z eksenine alınmasıyla tüm nesnelerin uzaydaki bir kameranın veya bir gözün bakış yönüne göre yerleştirilmesi sağlanır. Bu, tüm nesneler için bir döndürme işlemidir.
- Üç boyutlu nesnelerin iki boyutlu görüntü yüzeyinde görüntülenmek üzere izdüşümlerinin alınmasıyla z koordinatları ortadan kaldırılır. İzdüşüm, uzayın boyutunu azaltma işlemidir. Nesnelerin boyutlarının göreceli oranları korunarak değişik yer ve bakış açılarından bakılarak elde edilen görüntülerden yararlanan mimari ve mühendislik uygulamalarında kullanılan paralel izdüşüm yönteminin yanında, görüntülerde derinlik hissi uyandıran perspektif izdüşüm gibi daha gerçekçi yöntemler de kullanılabilir.

- Görünmeyen yüzeylerin ortadan kaldırılmasıyla görünen yüzeyler belirlenir. Bakış noktasından görünmeyen yüzeylerin ekranda görüntülenmemesi istenmektedir. Nesnelerin arka yüzeyleri ve diğer nesnelerin arkasında kalan nesneler görünmemelidir. Yüzey normali arkaya bakan nesnelerin belirlenmesiyle arka yüzeyler ortadan kaldırılır. Diğer nesnelerin arkasında kalan nesnelerin ortadan kaldırılması için ise değişik yöntemler kullanılabilir. Bunlardan birisi, tüm nesnelerdeki görüntülenecek tüm yüzeylerin bakış noktasına göre uzakta olandan yakına olana doğru sıralanmasıdır. Böylece yakında olan nesneler daha sonra çizileceklerinden ekranda diğerlerinin üzerini kaplarlar. Üçgen veya dörtgen şeklindeki çokgenleri uzaklıklarına göre sıralamak için kullanılan en basit yöntem, köşelerin z koordinatlarının derinlik ortalamalarının kullanılmasıdır. Fakat bu yöntem karmaşık nesneler ve nesne grupları için gerçek sonuçları vermez. Ortalama z değeri, n köşeli bir çokgende i köşe numarası ve z_i , köşenin z değeri olmak üzere şu şekilde hesaplanır :

$$z_{ortalama} = \frac{1}{n} \sum_{i=0}^{n-1} z_i$$

- İki boyutlu görüntü yüzeyine aktarılmış ve görünecek olan nesneler için bir çeşit ölçeklendirme yapılarak, gerçek dünyadaki pencerenin ekran penceresine eşlenmesi sağlanır. Bu işlem, gerçek koordinatlardan araç koordinatlarına dönüşüm olarak adlandırılır.
- Son aşama, nesnelerin ekranda görüntülenmesidir. Yüzeylerin çizimi, çokgen gibi basit iki boyutlu nesnelerin çizilmesini sağlayan çizim fonksiyonları yardımı ile gerçekleştirilir. Nesnelerin telkafes (wireframe) şeklinde sadece kenarlarının çizilmesi isteniyorsa çizgiler veya içi boş çokgenler, katı (solid) bir nesne şeklinde renkli

olarak çizilmesi isteniyorsa içi dolu çokgenler kullanılır. Nesnelerin yüzeylerine renk, desen ve doku verilebilir.

Bilgisayar grafiklerinde gerçekçi görüntüler oluşturmak için, yukarıda anlatıldığı gibi nesnelerin arka ve görünmeyen yüzeylerinin elenmesi ve perspektif izdüşüm gibi yöntemler kullanılarak tam olarak temsil edilmesi dışında, ışın izleme (ray tracing) ve ışıma (radiosity) yöntemlerinin de kullanılması gerekmektedir. Bir nesne üzerinde görülen renkleri ve aydınlatma etkilerini (ışık yansımaları, saydamlık, gölgelendirme gibi konular da dahil) modellemek karmaşık bir işlemdir; fiziksel ve psikolojik yönleri vardır. Aydınlatma etkileri temelde, nesne yüzeylerinin elektromanyetik enerji etkileşimini dikkate alan modellerle tanımlanır. Işık, insan gözüne ulaştığında, gerçek ortamda görülenleri tanımlayan algılama işlemleri başlar. Fiziksel aydınlatma modellerinde, nesnelerin türlerinin, ışık kaynaklarına ve diğer nesnelere göre nesnelerin yerlerinin, ışık kaynaklarının özelliklerinin bilinmesi gerekir. Nesneler, saydam olmayan maddelerden yapılmış olabilecekleri gibi, çok veya az saydam da olabilirler. Parlak veya donuk yüzeyli olabilecekleri gibi, değişik yüzey desenlerine de sahip olabilirler. Nesneler gibi, ışık kaynaklarının da değişik konum, renk ve nitelikleri olabilir (Hearn ve Baker, 1997).

7.4 Üç Boyutlu Uygulama Geliştirme Arayüzleri

Yıllar boyunca pek çok grafik standardı geliştirilmiş ve bunlara dayanan uygulama geliştirme arayüzleri kullanıma sunulmuştur. İlk grafik standardı 1977 yılında hazırlanıp, 1979 yılında düzeltilen "Core" grafik standardıdır. Bu tarihten günümüze kadar, 4 nesil grafik uygulama geliştirme arayüzü ortaya çıkmıştır. Aşağıdaki çizelgede bu grafik API'leri ve özellikleri listelenmiştir (Çizelge 7.1) :

Çizelge 7.1 : Grafik API'lerinde 4 nesil (Sun Microsystems,1999).

API	Donanım	Dil	Bağlantı	Üç Boyutlu Grafikler	Yeni Kullanıcılar	Yıl
Siggraph Core	Minibilgisayarlar	Fortran	Seri G/Ç	Üç Boyutlu Primitive'ler	Şirketler	1970'ler
				Yüzeyler		
PHIGS/ PEX	İş İstasyonları	C	G/Ç taşıtları ve Ethernet	Katı Nesneler	İş istasyonu kullanıcıları	1980'ler
				Temel Üç Boyut		
				Hızlandırma		
Open GL	İleri Üç Boyutlu Hızlandırılmış İş İstasyonları	C++	Entegre üç boyutlu Grafik Sistemleri	Hızlı mod	Animasyon Simülasyon	1980 sonları
				Örüntü Eşleme		
				İleri Aydınlatma		
				Hızlandırılmış üç boyutlu donanım		
Java 3D	Java Terminalleri Ağ Bilgisayarları Oyun Konsolları	Java	İnternet Ağları	Görüntü Ağacı	Oyun geliştiriciler Web sayfası tasarımcıları	1990 sonları
				Geometri Sıkıştırma		

Yaklaşık olarak her on yılda, bilgisayar grafikleri ile uğraşan kişiler üç boyutlu API standartlarını yenilemek durumunda kalmışlardır. Bunu etkileyen temel faktörler şunlardır :

- Bilgisayar grafiklerinde oluşan bilgi birikiminin hızlı bir şekilde artışı
- Grafik donanımlarının çok hızlı bir şekilde gelişmesi
- İşletim sistemlerindeki gelişmeler
- Programlama dillerindeki gelişmeler ve yeni geliştirilen programlama dilleri
- Üç boyutlu grafik uygulamaları geliştirmek isteyen topluluk ve kişilere yenilerin eklenmesi

Günümüzde yaygın olarak kullanılan ve gelişmiş üç boyut desteği olan 3B API'lerden ikisi OpenGL ve Java 3D API'dir.

OpenGL, kaliteli ve yüksek düzeyli grafikler oluşturulmasını sağlayan, iki ve üç boyutlu grafikleri destekleyen, işletim sisteminden bağımsız bir uygulama geliştirme arayüzüdür. Etkileşimli üç boyutlu grafik uygulamalarında gerekli olan yaklaşık 250 farklı komutu içerir (Woo ve ark., 1999). Birçok platformda çalışması, geniş ve iyi tanımlanmış özellikleri, hızı, sürekli gelişmesi ve ölçeklenebilir olması sayesinde oldukça yaygınlaşmıştır. Güvenilir ve kullanımı kolay bir donanım soyutlama katmanıdır. 1992 yılında ortaya çıkışından bu yana bilimsel programlardan görsel simülasyonlara, animasyonlardan oyunlara kadar çok geniş bir yelpazedeki uygulamanın yüksek performansla bilgisayar ortamına taşınmasını sağlamıştır.

Java 3D API, Java programlama diline üç boyutlu grafik ve görüntüleme yetenekleri kazandırmak üzere geliştirilmiştir. Java 3D API, etkileşimli üç boyutlu grafikleri içeren uygulamalar için bir dizi nesneye yönelik yüksek performanslı bir arayüz sağlar. Yazılım geliştiriciler, üç boyutlu nesneler ve görsel ortamlar oluşturmayı, bunlar üzerinde işlemler yapmayı ve davranışlarını denetlemeyi kolaylaştırıp güçlendiren yüksek düzeyli ve kolay programlama modeliyle Java 3D'nin avantajlarından yararlanırlar. Bu model, görüntüleme tekniklerinden çok yazılım içeriğine odaklanmayı sağlar. Sonuçta yazılım geliştiriciler, yüksek kaliteli, ölçeklenebilir ve platform bağımsız üç boyutlu grafikleri Java tabanlı uygulamalar ve appletler ile kolaylıkla bütünleştirebilir (Brown ve Petersen, 1998) (Day, 1998) (Sowizral ve ark., 1998). Web tarayıcılarının birçoğu Java 3D Applet'lerini doğrudan çalıştıramamakla birlikte, Java 3D API yüklendikten sonra yapılan ayarlamalarla bu sorun da ortadan kalkmaktadır. Donanımların gelişmesi ile kısa sürede Java 3D'nin yaygın bir kullanım alanı bulacağı açıktır.

Bunlar dışında VRML ve OpenInventor gibi birçok grafik uygulama geliştirme arayüzü vardır.

Günümüzde yüksek düzeyli üç boyutlu uygulama geliştirme arayüzlerinin sağladığı olanaklar ve özellikler şu şekilde belirtilebilir :

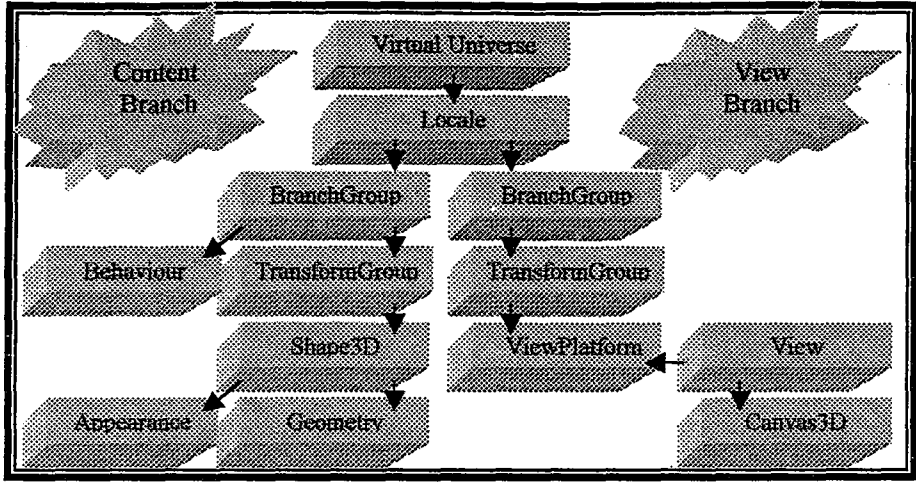
- Üç boyutlu sanal geometrik nesneler tanımlamayı, sahneye eklemeyi, çıkarmayı ve gruplandırma işlemlerini kolaylaştıran altyapıyı sunmaktadır. Matematiksel veri tipleri, veri yapıları ve bunlar üzerinde işlem yapacak çeşitli metotlar tanımlıdır. Programcılar kendi geometrik şekillerini tanımlayabilmek için indeks tabanlı olan veya olmayan geometrileri kullanabilirler.
- Üç boyutlu içerik eklemeyi kolaylaştırır :
 - Üç boyutlu değişik grafik yazılımları ile oluşturulmuş dosya biçimlerindeki üç boyutlu nesnelerin eklenmesi
 - İki ve üç boyutlu yazılar eklenmesi
- Üç boyutlu nesneler üzerinde değişik dönüşüm işlemleri ve değişik türlerde animasyonlar yaptırılmasını sağlarlar. Üç boyutlu nesneler üzerinde özellikle fare ve klavye gibi çevre aygıtları yardımı ile etkileşimli bir şekilde işlemler yapılabilmesini sağlayan olanaklar vardır.
- Üç boyutlu nesnelerin niteliklerinin kolaylıkla belirlenmesini ve değiştirilmesini sağlarlar. Geniş bir nesne nitelikleri kütüphanesine sahiptirler.
- Nesneler üzerine değişik türlerde desenler kaplanmasını kolaylaştırır.
- Değişik türlerde ışık kaynaklarının tanımlanmasını ve aydınlatma ve sis gibi etkilerin verilmesini kolaylaştırır.

- Kameraya yani bakış noktasına ilişkin işlemlerin yapılmasını kolaylaştırırlar. Üç boyutlu değişik görüntüleme olanakları (paralel, perspektif) gibi esnek kamera özelliklerini sunarlar.
- Üç boyutlu grafik ve animasyonlara ses özellikleri eklemeyi kolaylaştırırlar.
- Hazır geometri sıkıştırma özellikleri ile etkinlik sağlarlar.
- Güvenlik ve güvenilirliği sağlayıcı özellikleri mevcuttur.
- Altyapıları, grafiklerin oluşturulmasında ve diğer işlemlerde olabilecek en iyi hızı sunmaya çalışmaktadır

Bu bölümde, yüksek düzeyli üç boyutlu uygulama geliştirme arayüzlerinin en yaygın olanlarından Java 3D API kısaca incelenmiştir. Java 3D API'nin geometrik özellikleri ayrıntılı olarak Bölüm 8'de anlatılmaktadır.

7.4.1 Java 3D Görüntü Ağacı (Scene Graph)

Java 3D'nin Görüntü Ağacı (Scene Graph) modeli, deneyimli olmayan grafik programcılarının, geliştirdikleri uygulamalara üçüncü boyutu eklemelerini sağlar. Üç boyutlu bir uygulama geliştirmek isteyen kişi Java 3D Görüntü Ağacını (Şekil 7.6) kullandığında, nesneleri ve görüntüleme komutlarını temsil etmek için yapması gereken tek şey sadece ağaç benzeri yapıyı oluşturmaktır.



Şekil 7.6 : Java 3D Görüntü Ağacı

Java 3D, görüntü ağacı tabanlı bir API'dir. Yani üç boyutlu bir resim, üç boyutlu uzaydan ve sınıf nesnelerinin eşlendiği gerçek üç boyutlu nesnelerden oluşmaktadır. Java 3D programı işletildiğinde gerçek üç boyutlu görüntü oluşur. Programcının gerçekleştirmesi gereken ilk işlem sanal dünya (*Virtual Universe*) nesnesini oluşturmaktır. Ardından bir veya daha fazla *Locale* nesnesi ekleme aşaması gelir. Bir *Locale* nesnesi koordinat sistemini tanımlar. Koordinat biriminin 1.0f olarak tanımlanması gerçek fiziksel dünyada 1 metre uzunluğa karşılık gelir. Bir veya daha fazla alt grafik oluşturulup *Locale* nesnesine eklenir. (Brown ve Petersen, 1998).

Görüntü ağacının iki dalı veya alt ağacı vardır. Her biri, davranışlar, dönüşümler, görünüm nesneleri ve şekil nesneleri gibi ek nesneler içerir. Bunların eklenmesi de program içinde gerçekleştirilmektedir. Bu nesneler daha sonra üç boyutlu görüntünün ekranda oluşturulması için işleme sokulur. Her Java 3D programında bir alt ağaç grubu (*BranchGroup*) nesneler dalını (*content branch*), diğer alt

ağaç grubu da görünüm dalını (*view branch*) oluşturur. Nesneler dalı, şekiller, dokular, ışıklar vb. içerirken, görünüm dalı büyütme (zoom) gibi görünüm bağımlı etkinlikler ile ilgilidir (Brown ve Petersen, 1998).

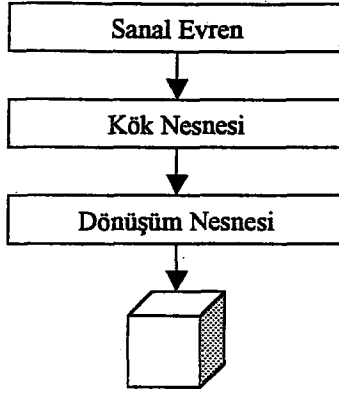
Görünüm Platformu (*View Platform*) nesnesi, Java 3D görüntü ağacında bir yaprak düğümdür ve bakış noktasına (viewpoint) karşılık gelir. Belirli bir görünüm platformu nesnesinin üzerindeki dönüşüm grubunun (*TransformGroup*) değişimi, sanal dünyadaki bakış noktasının yerini ve açısını değiştirir. Görünüm platformu, sanal dünyadaki koordinat sistemini tanımlar. Görünüm (*view*) nesnesi görünüm modelinin tüm yönleri ile koordinatlandırılmasında kullanılan ana nesnedir. *Canvas3D* ise “awt canvas” sınıfının üç boyutlu şeklidir. Java 3D’nin nesneleri görüntüleyeceği alandır (Brown ve Petersen, 1998).

Shape3D, üç boyutlu nesneler oluşturmak için kullanılır. *Appearance*, renk gibi görüntüleme özelliklerini tanımlar, *Shape3D* gibi nesneler tarafından kullanılır. Geometri (*Geometry*), *Shape3D* düğümü tarafından kullanılan geometri bileşenleri soyut sınıftır. Davranış (*Behaviour*) nesnesi, nesnelerin hareketlendirilmesi ile klavye ve fare olaylarının denetlenmesi gibi etkinlikleri yapmak için kullanılır. Dönüşüm Grubu (*TransformGroup*) aynen görünüm dalında olduğu gibi alt ağaçlar üzerinde dönüşüm (yer değiştirme, döndürme, ölçeklendirme) işlemleri yapmak için kullanılır (Brown ve Petersen, 1998).

Pratikte görünüm dalı, nesneler dalı ile kıyaslandığında oldukça az yer kaplar. Görünüm dalı genelde sadece birkaç düğüm içerirken, nesneler dalı binlerce karmaşık üç boyutlu dünyadan oluşabilir.

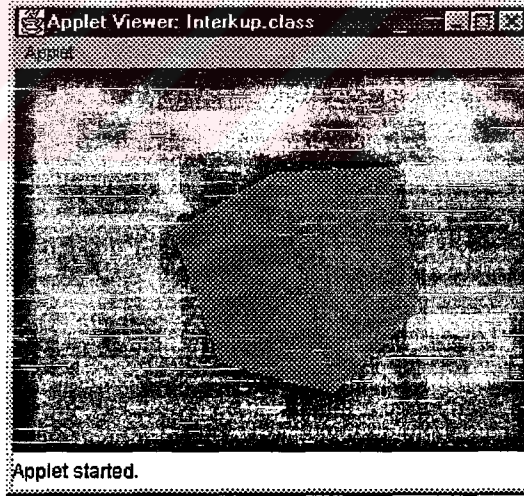
7.4.2 Java 3D ile 3B Sahnelerin Oluşturulması

Java 3D ile üç boyutlu sahneler, görüntü ağacına nesnelerin ve özelliklerinin konulması ile oluşturulur (Şekil 7.7).



Şekil 7.7 : Bir küp içeren “Görüntü Ağacı”

Şekil 7.8’de applet penceresi görülen etkileşimli küpün oluşturulması, görüntü ağacına hazır 3B ilkel (primitive) olan renkli küp nesnesinin düğüm olarak eklenmesi ile gerçekleştirilir.



Şekil 7.8 : Etkileşimli bir küp oluşturan programın applet penceresi

Küp nesnesi ilgili ayarlamalardan sonra üç boyutlu uzaydaki konumunu ve eğimini tutan dönüşüm grubu nesnesinin (objTrans) çocuk düğümü olarak sisteme eklenmektedir :

- **Dönüşüm grubu ekleme**

```
TransformGroup objTrans = new TransformGroup();
objRoot.addChild(objTrans);
```

- **Renkli bir küp ekleme**

```
objTrans.addChild(new ColorCube(0.4));
```

objTrans dönüşüm grubu nesnesi kök düğüme bağlanmaktadır. Değişik konumlarda, eğimlerde ve şekillerde birden çok 3B nesne içeren sahnelerde kök düğümün altına, dönüşüm grubu ile beraber birden çok 3B nesne düğümü konulabilir. Ayrıca gruplama amacı ile, birden çok 3B nesne düğümü, bir dönüşüm grubu altına da eklenebilir. Böylece nesnelerin 3B uzayda ortak hareket etmeleri sağlanır.

Nesnelere aradeğerleyiciler (Interpolator) yardımı ile hareketler de verilebilir. Etkileşim vermek için MouseRotate, MouseTranslate ve MouseZoom sınıflarından yararlanılır. Böylece programı kullanan kişiler, sahnedeki nesneleri fare yardımı ile üç boyutlu uzayda herhangi bir ekseninde döndürebilirler ve taşıyabilirler. Aşağıda, dönüşüm grubu üzerinde etkili olacak bir fare etkileşiminin verilmesini sağlayan kod parçası görülmektedir :

- **Etkileşim ekleme**

```
MouseRotate behavior1 = new MouseRotate();
behavior1.setTransformGroup(objTrans);
objTrans.addChild(behavior1);
```

Hazır illkellere veya oluşturulan nesnelere birçok metot aracılığı ile aydınlanma özellikleri verilebilir, resim kaplanabilir, renklendirme yapılabilir veya geçirgenlik verilebilir.

Nesneler ve kamera üzerinde etkili olacak dönüşüm işlemlerini (taşıma, döndürme ve ölçeklendirme) gerçekleştiren birçok metot da vardır. Java 3D'nin geometrik özelliklerine 8. bölümde değinilmektedir.

7.5 Üç Boyutlu Grafik Yazılımları Geliştirmede 2B ve 3B API'lerin Karşılaştırılması

3B grafik yazılımları geliştirmede 3B API'ler hazır olanaklar sağlamaktadır. Buna karşın, 3B yazılımların geliştirilmesinde, 2B API'lerin kullanımı ileri derecede matematik ve grafik bilgisi gerektirmektedir. İki boyutlu ilkeller yardımı ile üç boyutlu nesnelerin oluşturulması, oluşturulan nesneler için veri yapılarının tasarlanması, izdüşüm yöntemlerinin uygulanması, görünen yüzeylerin belirlenmesi, aydınlatma ve ışık kaynaklarının durumuna göre nesnelerin yüzeylerindeki renklerin değerlerinin hesaplanması, yapılması gereken işlemlerden sadece birkaçıdır. Üç boyutlu grafik yazılımları geliştirmek bu nedenle yazılım geliştiriciler için zor olmuştur. Donanım ve yazılımlardaki gelişmeler sonucunda üç boyutlu uygulama geliştirme arayüzlerinin kişisel bilgisayar ortamına taşınması grafik yazılımları geliştirme sürecini hızlandıracak ve kolaylaştıracaktır.

8. JAVA 3D VE GEOMETRİ

8.1 Java 3D'nin Geometri Özellikleri

Java 3D kullanılarak hazırlanan grafik programlarına üç boyutlu geometrik içerik eklemenin üç yolu vardır : (Bouvier, 1999)

- Kutu (Box), koni (Cone), silindir (Cylinder), ve küre (Sphere) gibi hizmet sınıflarını kullanmak.
- Köşe noktalarının, doğru parçalarının ve çokgensel yüzeylerinin tanımlanması ile üç boyutlu nesneyi programlayarak oluşturmak.
- Yükleyicileri kullanarak hazır üç boyutlu nesneleri getirmek.

Büyük ölçekli karmaşık nesnelerden oluşan geometrik içeriği daha kolay oluşturmak amacıyla yükleyiciler kullanılabilir. Bu sayede grafik yazılımlarında daha az kod yazılacak, daha az hata yapılacak ve yazılım daha hızlı bir şekilde geliştirilecektir. Java 3D nesnelerini, geometriyi elle kodlamak yerine 3D modelleme yazılımları ile oluşturulmuş dosyalardan almak daha uygundur. VRML, Lightwave, AutoCAD, Wavefront ve diğer grafik yazılımları ile oluşturulmuş üç boyutlu grafik modellerini Java 3D programlarına getiren birçok yükleyiciler yazılmıştır. Java 3D programına yükleyiciler aracılığıyla alınan nesneler üzerinde animasyonlar, etkileşim işlemleri, renk ve görünüm değiştirme gibi işlemler yapılabilir.

8.2 Java 3D Geometrisi

Java 3D, geometrik işlemleri kolaylaştırmaktadır. Bu amaçla geometrik hizmet sınıfları, matematiksel sınıflar ve geometri sınıfları hazırlanmıştır. Geometrik hizmet sınıfları, üç boyutlu temel şekillerin çok kolay bir şekilde eklenmesini sağlar. Köşe, vektör ve çokgen gibi bilgileri tanımlamak ve üzerlerinde işlem yapmak için matematiksel sınıflardan yararlanılır. Üç boyutlu nesne modelleri oluşturmak için ise

geometri sınıfları kullanılır. Bunlar dışında görünüm ve niteliklerin de ayarlanması için sınıflar vardır. Türlerine göre bu sınıflardan bazıları aşağıdaki gibidir :

Geometrik hizmet sınıfları

- Box (dikdörtgenler prizması)
- Cone (koni)
- Cylinder (Silindir)
- Sphere (Küre)
- ColorCube (Renkli Küp)

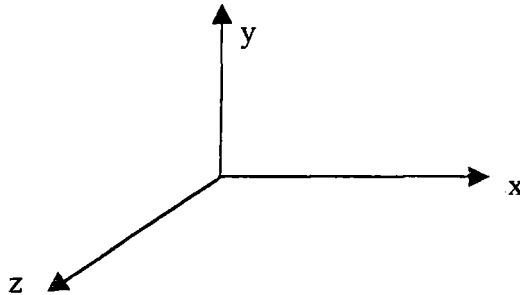
Matematiksel Sınıflar

- Point sınıfları
- Color sınıfları
- Vector sınıfları
- TexCoord sınıfları

Geometri Sınıfları

- GoemetryArray sınıfı
- GeometryArray sınıfının alt sınıfları
- GeometryStripArray sınıfının alt sınıfları
- IndexedGeometryArray sınıfının alt sınıfları

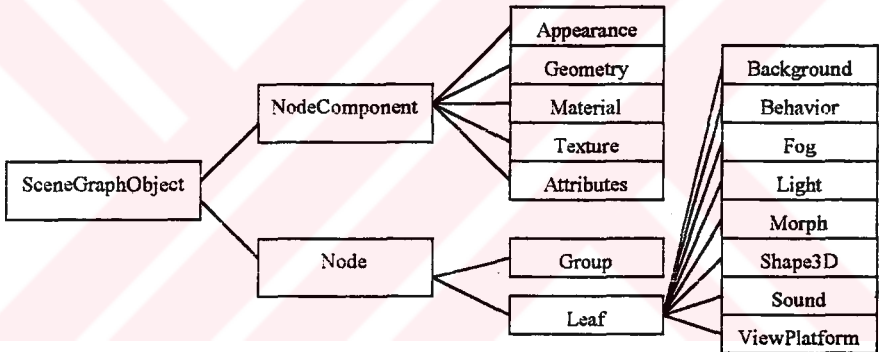
Java 3D sanal evreninin koordinat sistemi sağ el koordinat sistemidir (Şekil 8.1). Bu sistem x eksenini sağa, y eksenini yukarıya doğru çıktığında, z ekseninin bakış noktasına doğru arttığı koordinat sistemidir.



Şekil 8.1 : Java 3D koordinat sistemi (sağ el)

Java 3D kullanarak üç boyutlu bir şekil tanımlamak için Shape3D sınıfı kullanılır (Şekil 8.2). Shape3D nesneleri, görüntü ağacında sadece yaprak düğümler olabilir ve üç boyutlu nesnenin şekli ve rengi konusunda bilgi içermez. Bu bilgiler Geometry ve Appearance düğüm bileşenlerinde tutulur ve bir Shape3D, gerektiğinde bu bileşenlere başvuru yapar. Ayrıntılı bilgi Bouvier (1999)'da yer almaktadır.

Sınıf hiyerarşisinde üç boyutlu nesnelerin geometrisi ile özellikleri, görünümü, madde özellikleri ve görünümü ayrı yerlerde dir. Geometri içerisinde nitelikler ayrıntılı olarak verilemez. Oluşturulan üç boyutlu şeklin geometrisi dışındaki diğer özellikleri, dışarıdaki diğer sınıflara başvuru yapılarak alınmaktadır.



Şekil 8.2 : Java 3D Sınıf sıradüzeni

8.2.1 Sanal Nesne Tanımlama

Sanal nesneler aşağıdaki gibi sınıflar oluşturularak tanımlanabilir (Sanal nesne sınıfının iskelet kodu) :

```
public class VisualObject extends Shape3D {
    private Geometry voGeometry;
    private Appearance voAppearance;

    // Üç boyutlu bir nesneyi (Shape3D) geometrisi ve
    // görünümü ile oluştur
    // Geometri, createGeometry() metodu ile
    // oluşturulmaktadır.
    // Görünüm createAppearance() metodu ile
    // oluşturulmaktadır.

    public VisualObject() {
        voGeometry = createGeometry();
        voAppearance = createAppearance();
        this.setGeometry(voGeometry);
        this.setAppearance(voAppearance);
    }

    private Geometry createGeometry()
    { Sanal nesnenin geometrisini yapılandıran kod };
    private Appearance createAppearance()
    { Sanal nesnenin görünümünü yapılandıran kod };
}
```

Oluşturulan bu sanal nesne, görüntü ağacına “objRoot.addChild(new VisualObject);” komutu ile yerleştirilir.

8.2.2 Geometrik Hizmet Sınıfları

Sanal evrende üç boyutlu nesne oluşturmanın en kolay yolu kutu, silindir, koni ve küre hazır sınıflarını kullanmaktır. Bu hazır sınıflar aşağıdaki gibidir;

- Box (dikdörtgenler prizması) sınıfı : Dikdörtgenler prizması oluşturur. Merkezi orijinde bulunan kutunun uzunluk, genişlik ve yüksekliği 2’şer metredir : (-1,-1,-1)’den (1,1,1)’e kadar.

Box();

Box(float xdim, float ydim, float zdim, Appearance appearance);

- **Cone (Koni) sınıfı :** Yarıçapı 1.0 m ve yüksekliği 2 m olan bir koni oluşturur.

Cone();

Cone(float yarıcap, float height);

- **Cylinder(Silindir) sınıfı :** Merkez eksenini y ekseninde, merkezi orijinde bulunan $r=1m$ ve $h=2m$ olan bir silindir oluşturur.

Cylinder();

Cylinder(float yarıcap, float height);

- **Sphere (Küre) sınıfı :** Merkezi orijinde bulunan $r = 1m$ yarıçaplı bir küre oluşturur

Sphere();

Sphere(float yarıcap);

8.2.3 Matematiksel Sınıflar

Nesne oluşturmak için geometri sınıfları ve alt sınıfları kullanılır. Grafik programlamada değişik türlerde matematiksel sınıflardan yararlanılır. Örnek olarak Point, tanımlanan nesnelerin her köşe noktasının tutulmasında kullanılır. Bu tür hizmet sınıflarından bazıları aşağıdaki gibi belirtilebilir :

- Point (koordinatlar için)
- Color (renkler için)
- Vector (yüzey normalleri için)
- TexCoord (desen koordinatları için)

8.2.4 Geometri Sınıfları

Üç boyutlu bilgisayar grafiklerinde basit bir üçgenden karmaşık bir uçağa kadar herşey köşe tabanlı verilerle modellenerek oluşturulur. “Geometry” sınıfının alt sınıfları üçe ayrılır :

- İndeksli olmayan köşe tabanlı geometri (Nesne görüntüleneceğinde her köşe sadece bir kere kullanılır)
- İndeksli köşe tabanlı geometri (köşeler birden fazla kullanılabilir)
- Diğerleri (üç boyutlu metin, sıkıştırılmış geometri gibi)

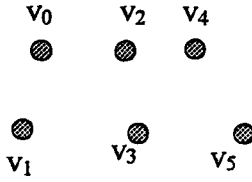
GeometryArray Sınıfı

“Geometry” alt sınıfları nokta, çizgi ve çokgenleri tanımlamakta kullanılabilir. Bu köşe tabanlı sınıflar “GeometryArray” sınıfının alt sınıflarıdır. Her köşe noktasına ilişkin bilgileri tutan diziler vardır.

Örnek olarak, “GeometryArray” sınıfı sadece bir üçgeni tutacak şekilde belirtilirse üç elemanlı bir dizi tanımlanır (her köşe için bir eleman olmak üzere). Dizinin her elemanı bir köşenin koordinat konumunu tutar (Noktalar point nesnesi ile tanımlanabilir). Koordinat konumu dışında istenirse rengi, yüzey normalini ve desen verisinin koordinatlarını tutmak üzere üç farklı dizi de tanımlanabilir (Bouvier, 1999).

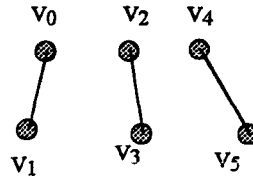
GeometryArray Alt Sınıfları

“GeometryArray” alt sınıfları : PointArray, LineArray, TriangleArray ve QuadArray’dir (Şekil 8.3).



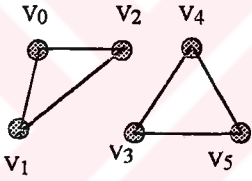
Point Array

6 tane nokta



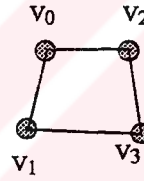
Line Array

3 tane çizgi



Triangle Array

2 üçgen



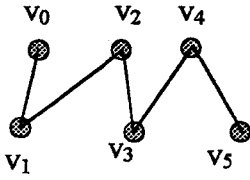
Quad Array

1 dörtgen

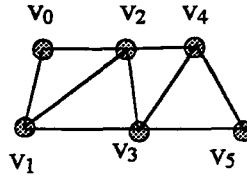
Şekil 8.3 : “GeometryArray” alt sınıfları

“GeometryArray” alt sınıfları, aynı köşenin tekrar kullanımına izin vermez. Bazı nesne tanımlarında aynı köşenin defalarca kullanılması gerekir. “GeometryStripArray”, şerit ilkellerinden (bileşik çizgi ve yüzeyler için) türetilen soyut bir sınıftır. LineStripArray, TriangleStripArray ve TriangleFanArray’in üst sınıfıdır (Şekil 8.4). “LineStripArray” bağlı çizgilerden oluşur, “TriangleStripArray” bir

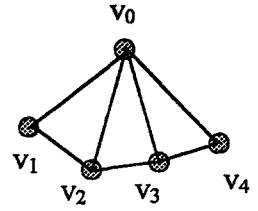
kenarı paylaşan üçgenlerden oluşur. “TriangleFanArray” ilk köşeyi her üçgende kullanır.



LineStripArray



TriangleStripArray



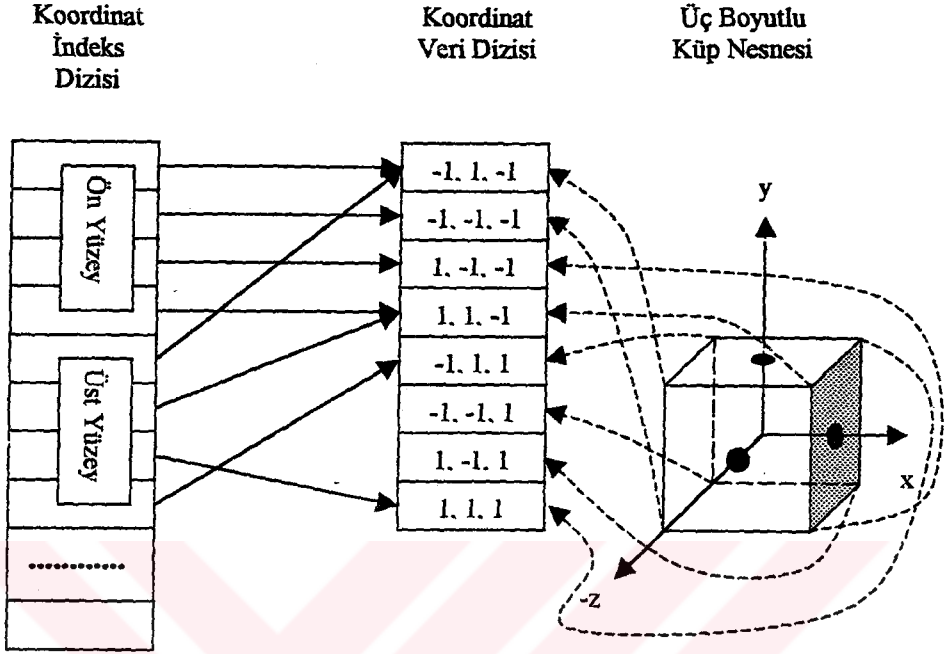
TriangleFanArray

Şekil 8.4 : “GeometryStripArray” alt sınıfları

“IndexedGeometryArray” Alt Sınıfları

“GeometryArray” alt sınıfları köşeleri verimsiz bir şekilde kullanmaktadır. “GeometryStripArray” alt sınıflarında ise köşelerin sınırlı kullanımı vardır. Birçok geometrik nesnede ise önemli ölçüde yeniden kullanım söz konusudur. Örnek olarak bir küp tanımlamasında, sekiz köşenin her biri üçer yüzeyde kullanılmaktadır. 8 köşe tanımlanmış halde, 6 yüzey için 3'er köşeden 24 köşe tanımlamak etkin değildir.

Bu problem ek indeks dizileri tutularak çözülür (Şekil 8.5). İndeks dizileri, veri dizilerindeki aynı köşeye başvuru yapabilir. İndeks dizilerindeki değerler, kaplama sırasında izlenecek sırayı belirler. Bu yöntemin dezavantajı ise performansı azaltmasıdır.



Şekil 8.5 : İndeks Dizisi tutularak üç boyutlu nesne tanımlama

8.3 Sonuç

Java 3D'nin grafik programlamada sunduğu geometrik olanaklar şu şekilde belirtilebilir :

- Hazır matematiksel sınıflar vardır. Farklı boyutlardaki köşe noktalarını, vektörleri, matrisleri tutacak veri yapılarının ve sınıfların oluşturulması ve işlemlerinin kodlanması gerekmez.
- Değişik türlerdeki üç boyutlu grafiksel nesneleri tanımlamak için, indeksli olan veya olmayan geometri olanaklarını sunar.
- Temel üç boyutlu nesneleri eklemek için hazır olanaklar sunar.

- Grafik ve modelleme programlarında hazırlanan karmaşık geometrik nesne veya modellerin de görüntü ağacına eklenmesini sağlayarak program kodu yazma yükünü ortadan kaldırır.
- Üç boyutlu yazılar gibi özel üç boyutlu modellerin görüntü ağacına eklenmesini sağlar.
- Etkileşim ve animasyon işlemlerinde temel dönüşüm hesaplamalarının otomatik olarak yapılmasını sağlar. Matris işlemlerini de içeren uzun programlar yazılmasına gerek bırakmaz.



9. TEZ KAPSAMINDA GELİŞTİRİLEN YAZILIMLAR

Tez projesinin gerçekleştirim kısmında Java programlama dili kullanılarak, adları aşağıda verilen dört adet yazılım geliştirilmiştir.

- Kup3B (Java 2D sürümü)
- Model3B (Java 2D sürümü)
- Kup3B (Java 3D sürümü)
- Model3B (Java 3D sürümü)

Üç boyutlu geometrik nesneler oluşturarak üzerlerinde işlemler yapılmasını sağlayan bu yazılımlardan ikisi Java 2D ve ikisi de Java 3D kullanılarak gerçekleştirilmiştir. Tez çalışması kapsamında geliştirilmiş olan tüm yazılımlara "bornova.ege.edu.tr/~ugur" adresinden ulaşabilmekte ve programlar etkileşimli olarak İnternet üzerinden çalıştırılabilmektedir.

Bu bölümde, geliştirilen yazılımların özellikleri ve kullanımları anlatılarak, üç boyutlu grafik yazılımları geliştirmede iki ve üç boyutlu API'lerin kullanımıyla ortaya çıkan farklılıklar, avantajlar incelenmiştir. Elde edilen sonuçlar grafiklerle beraber değerlendirilmiştir. Yazılımların gerçekleştiriminde iki boyutlu uygulama geliştirme arayüzü olarak Java 2D ve üç boyutlu uygulama geliştirme arayüzü olarak Java 3D'nin seçilmesinin en önemli nedenleri aşağıda belirtilmektedir :

- Arayüzlerin en gelişmiş özellikleri içermeleri ve niteliklerinin yüksek olması
- Son beş yıl içerisinde geliştirilmiş arayüzler olmaları ve yeni geliştirilen grafiksel uygulama geliştirme arayüzleri arasında yapılan performans karşılaştırmalarının günümüzde yeterli düzeyde yapılmamış olması

- Karşılaştırmaların aynı programlama dili altından (Java gibi) çağrılan arayüzler arasında, daha etkin olarak yapılabilmesi olanağını sağlaması

Tez kapsamındaki çalışmalar sonucunda, Java 3D ve yüksek düzeyli diğer arayüzlerin öneminin büyük olduğu belirlenmiş ve gelecekte grafik yazılımları geliştirmede daha büyük rol oynayacakları sonucuna ulaşılmıştır.

9.1 Java 2D Kullanılarak Geliştirilen Yazılımlar

Java 2D iki boyutlu uygulama geliştirme arayüzü kullanılarak geliştirilen Kup3B ve Model3B yazılımlarında modelleme ve görüntüleme işlemleri, üç boyutlu nesnelerin görüntülenmesinde kullanılan standart işlem sırası Java programlama dili üzerinde gerçekleştirilerek yapılmıştır.

9.1.1 Kup3B (Java 2D sürümü)

Tez çalışması kapsamında Java 2D kullanılarak geliştirilen bu applete internet üzerinden erişilerek, bir küp üzerinde dönüşüm işlemleri (taşım, döndürme, ölçeklendirme) etkileşimli olarak klavye veya düğme takımı yardımı ile yapılabilmekte, perspektif, paralel görüntüler oluşturulabilmekte ve küp telkafes (wireframe) veya sabit gölgelendirmeli katı (solid) bir şekle getirilebilmektedir.

Bu çalışma kapsamında yazılmış olan appletin ilk sürümünde, daha çok kişi tarafından anlaşılmasını sağlamak amacıyla, köşe, kenar ve yüzey listelerini tutmak için daha esnek ve uygun olan bağlı listeler kullanmak yerine dizi kullanılmıştır. Bir küp nesnesinin 8 köşesinin x,y,z koordinatları $v[8][3]$ matrisinde tutulmaktadır. İlgili atamaları yapan metod aşağıda verilmiştir:

```

public void set_vertices()
{
    v[0][0] = 0 ; v[0][1] = 0 ; v[0][2] = 0;
    v[1][0] = 0 ; v[1][1] = 0 ; v[1][2] = 1;
    v[2][0] = 0 ; v[2][1] = 1; v[2][2] = 0;
    v[3][0] = 0 ; v[3][1] = 1; v[3][2] = 1;
    v[4][0] = 1; v[4][1] = 0 ; v[4][2] = 0;
    v[5][0] = 1; v[5][1] = 0 ; v[5][2] = 1;
    v[6][0] = 1; v[6][1] = 1; v[6][2] = 0;
    v[7][0] = 1; v[7][1] = 1; v[7][2] = 1;
}

```

Küpün 12 kenarının uç köşelerini belirten atamaları yapan metot aşağıda verilmiştir :

```

public void set_edges()
{
    e[0][0] = 0; e[0][1] = 1;
    e[1][0] = 0; e[1][1] = 2;
    e[2][0] = 0; e[2][1] = 4;
    e[3][0] = 7; e[3][1] = 3;
    e[4][0] = 7; e[4][1] = 5;
    e[5][0] = 7; e[5][1] = 6;
    e[6][0] = 6; e[6][1] = 2;
    e[7][0] = 6; e[7][1] = 4;
    e[8][0] = 1; e[8][1] = 3;
    e[9][0] = 1; e[9][1] = 5;
    e[10][0] = 2; e[10][1] = 3;
    e[11][0] = 4; e[11][1] = 5;
}

```

Küpün 6 yüzeyinin 4'er kenarını belirten atamaları yapan metot aşağıda verilmiştir :

```

public void set_surfaces()
{
    s[0][0] = 0; s[0][1] = 8; s[0][2] = 10; s[0][3] = 1;
    s[1][0] = 0; s[1][1] = 2; s[1][2] = 11; s[1][3] = 9;
    s[2][0] = 10; s[2][1] = 3; s[2][2] = 5; s[2][3] = 6;
    s[3][0] = 5; s[3][1] = 4; s[3][2] = 11; s[3][3] = 7;
    s[4][0] = 2; s[4][1] = 1; s[4][2] = 6; s[4][3] = 7;
    s[5][0] = 8; s[5][1] = 9; s[5][2] = 4; s[5][3] = 3;
}

```

Küpün tüm köşe noktaları için dönüşüm işlemlerini gerçekleştiren yöntem aşağıda verilmiştir. Küp nesnesinin tüm köşelerine uygulanacak işlemin dönüşüm matrisi x, y, z eksenlerindeki taşıma, ölçeklendirme ve döndürme parametre değerlerini içeren “parameter” dizisi kullanılarak elde edildikten sonra, tüm köşe noktalarının konumlarını gösteren vektörlerle tek tek çarpılır. Böylece köşe noktalarının dönüşümden sonraki konumlarının değerleri (ekran değerleri) bulunmuş olur. Perspektif izdüşümü alınarak görüntülenmesi gerektiğinde ek olarak perspektif izdüşüm dönüşüm işleminden de geçirilmektedir.

```
public void transform_all_points()
{
    int i, j;
    double t_4[] = new double[4];
    double i_4[] = new double[4];
    double t_16[] = new double[16];
    transform(t_16, parameter);

    for(i=0; i<8; ++i)
    {
        t_4[0] = v[i][0];
        t_4[1] = v[i][1];
        t_4[2] = v[i][2];
        t_4[3] = 1;
        mult(t_4, t_16, i_4);

        if (perspective_flag==false)
        { t_v[i][0] = i_4[0];
          t_v[i][1] = i_4[1]; } else
        { t_v[i][0] = i_4[0]*(zprp-zvp)/(zprp-i_4[2]);
          t_v[i][1] = i_4[1]*(zprp-zvp)/(zprp-i_4[2]);
        }
    }
}
```

Üç boyutlu küpün ekranda görüntülenmesini gerçekleştiren metod aşağıda verilmiştir. Küpün 6 yüzeyi, her birinin 4 köşesinin koordinatları saat yönünün tersine sırayla bulunarak ekrana çizilir. İlgili yüzeyin

normali, kameranın yönüne göre arka yüzeyse çizdirilmez. Nesne katı olarak görüntülenecekse, yüzeyleri Java'nın "fillPolygon" metodu (Deitel, 1999) (Knudsen, 1999) ile ve içi ilgili renkle dolu bir çokgen olarak çizdirilir. Telkafes olarak görüntülenecekse Java'nın "drawPolygon" metodu ile çizdirilir (Deitel, 1999) (Knudsen, 1999).

```
public void paint(Graphics g)
{
    int i,j,k;
    Dimension d;
    d = getSize();
    showStatus("Copyright by Aybars UGUR, [Ctrl,shift] yön
    tuslari veya PgUp PgDown ");

    for(i=0;i<6;++i)
    {
        temp_v[0] = e[s[i][0]][0];
        temp_v[1] = e[s[i][0]][1];
        temp_v[2] = e[s[i][1]][0];
        if(temp_v[2]==temp_v[0])
        {
            temp_v[0]=temp_v[1];
            temp_v[1]=temp_v[2];
            temp_v[2]=e[s[i][1]][1];
        } else
        if(temp_v[2]==temp_v[1])
        {
            temp_v[2]=e[s[i][1]][1];
        } else;
        temp_v[3] = e[s[i][2]][0];
        if(temp_v[3]==temp_v[2])
            temp_v[3] = e[s[i][2]][1];
        for(j=0;j<4;++j)
        {
            xValues[j] = (int)t_v[temp_v[j]][0];
            yValues[j] = d.height-(int)t_v[temp_v[j]][1];
        }
        if(solid_flag)
        {
            g.setColor(colors[i%6]);
            if( ((double) xValues[0]*(yValues[1]-yValues[2])+
                (double) xValues[1]*(yValues[2]-yValues[0])+
```

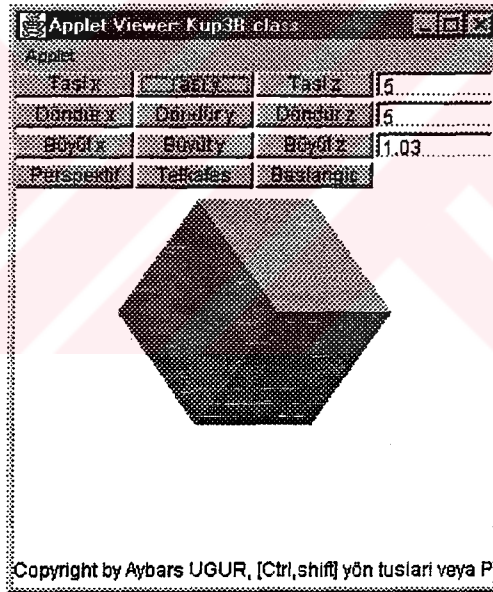
```

        (double) xValues[2]*(yValues[0]-yValues[1]))<0)
        g.fillPolygon(xValues,yValues,4);
    } else
    {
        g.setColor(Color.black);
        g.drawPolygon(xValues,yValues,4);
    };
}
}

```

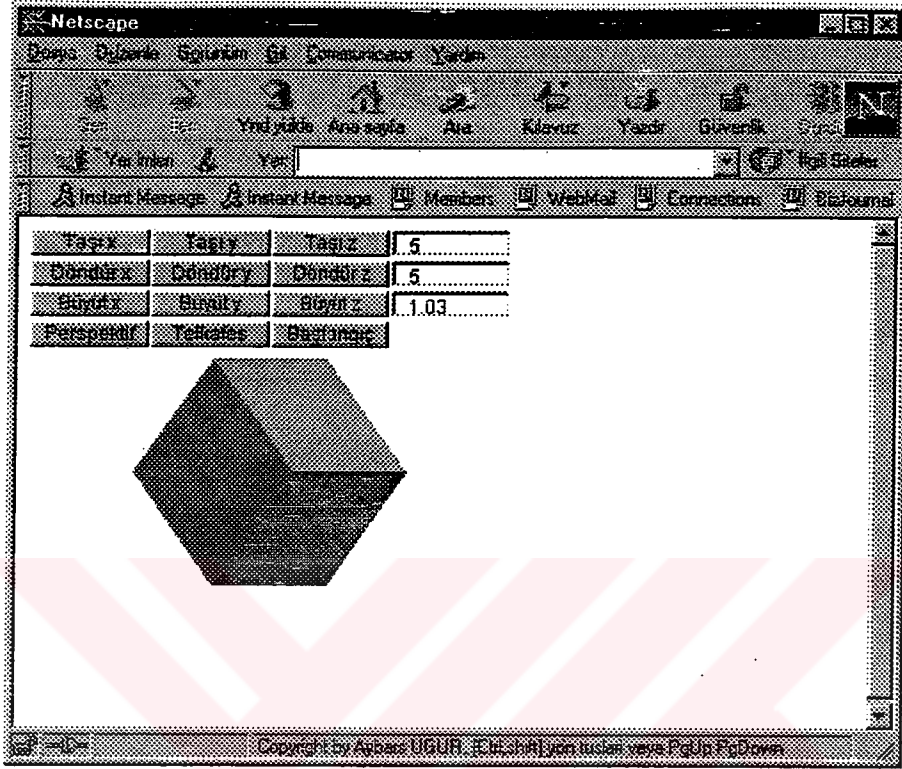
Applet'in İşletimi

Program appletviewer ile çalıştırıldığında ortaya çıkan ilk ekran Şekil 9.1'de gösterilmektedir :



Şekil 9.1 : Kup3B çalıştırıldığında ortaya çıkan appletviewer penceresi

Applet, Netscape ve Explorer gibi standart bir web tarayıcısı ile görüntülendiğinde ortaya çıkan ilk ekran da Şekil 9.2'deki gibidir :



Şekil 9.2 : Netscape gibi bir web tarayıcısı kullanılarak appletin yüklenmesi

Applet, web tarayıcısına yüklendiğinde, klavye ve/veya düğme paneli yardımı ile üç boyutlu uzayda x , y , z eksenlerinde taşınabilen, döndürülebilir ve ölçeklendirilebilir (büyütülüp küçültülebilen) üç boyutlu bir küp nesnesi ortaya çıkmaktadır. Küpün paralel (parallel) ve perspektif (perspective) görüntüsü alınabilmekte ve küp, telkafes (wireframe) ve katı (solid) biçimde görüntülenebilmektedir. Döndürme işlemi için ok tuşları ve PgUp ve PgDn tuşları kullanılmaktadır. Ölçeklendirme için bu tuşlar üst karakter (shift) tuşuna basılı iken kullanılmaktadır. Taşıma işleminde ise aynı tuşlar kontrol (ctrl) tuşuna basılı iken kullanılmaktadır. Ayrıca pencerenin üst bölümüne

yerleştirilmiş olan düğme seti yardımı ile de aynı işlemler yapılabilmektedir.

Panelde 12 tane düğme, 3 tane metin kutusu bulunmaktadır. İlk satırdaki üç düğme x , y , z eksenlerinde taşıma için kullanılmaktadır (Taşı x , Taşı y , Taşı z). İkinci satırdaki (döndür) üç düğme x , y , z eksenleri etrafında döndürme için kullanılmaktadır. Üçüncü satırdaki (büyüt) üç düğme x , y , z eksenlerinde ölçeklendirme için kullanılmaktadır.

İlk satırdaki (taşı satırı) metin kutusundaki değer bir yer değiştirme işleminde ne kadar taşıma yapılacağını belirtmekte ve kullanıcı tarafından değiştirilebilmektedir. Bu metin kutusuna herhangi bir gerçel sayı yazılabilmektedir. Yer değiştirme değeri pozitif olduğunda, x , y , z eksenlerinde taşıma yapmak için kullanılan düğmelerden birisine basıldığında küp ilgili ekseninde pozitif yönde taşınacaktır. Negatif olduğunda ise negatif yönde taşınacaktır. Bu metin kutusundaki başlangıç değeri 5'tir. Bu satırdaki herhangi bir düğmeye basılması, küpü ilgili ekseninde pozitif yönde 5 birim taşıyacaktır.

İkinci satırdaki (döndür satırı) metin kutusundaki değer, bir döndürme işlemindeki döndürme açısının miktarını belirtir ve kullanıcı tarafından değiştirilebilmektedir. Pozitif değerleri pozitif yönde, negatif değerleri negatif yönde belirtilen açı miktarınca (derece cinsinden) küpün döndürülmesini sağlar. Bu metin kutusundaki başlangıç değeri 5'tir. Bu satırdaki herhangi bir düğmeye basılması, küpü ilgili ekseninde pozitif yönde 5 birim döndürecekler.

Üçüncü satırdaki (büyüt satırı) metin kutusundaki değer, ölçeklendirme miktarını belirtir. Değerin 1'den büyük olması, bu satırdaki x , y , z eksenlerinde ölçeklendirmeyi sağlayan bir düğmeye basıldığında küpün o ekseninde yüzde olarak o değerde büyütülmesini, 1'den küçük olması ise küçültülmesini sağlar. Bu metin kutusundaki

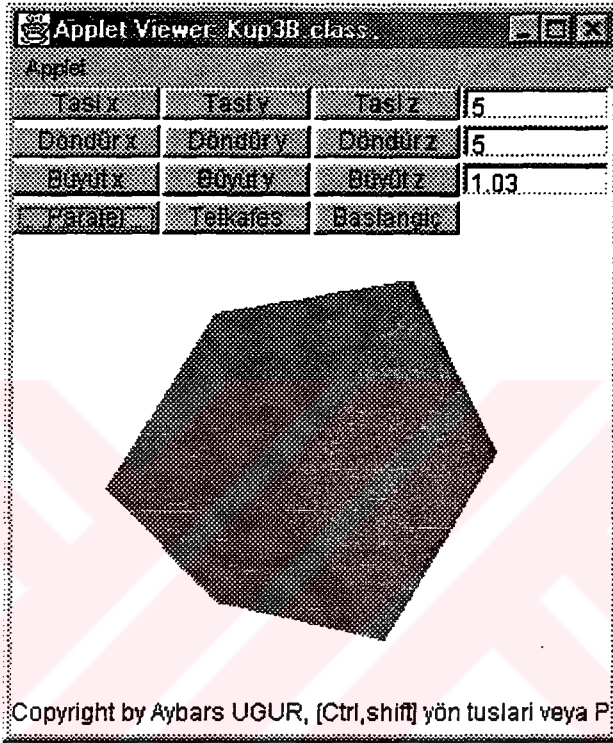
değer 1 olduğunda ise bu düğmeler küpün boyutlarında herhangi bir değişmeye neden olmazlar. Bu metin kutusundaki başlangıç değeri 1.03'tür. Bu satırdaki herhangi bir düğmeye basılması, küpün ilgili ekseninde 3% büyümesini sağlayacaktır. Kullanılan bazı düğmeler ve görevleri Çizelge 9.1'de listelenmiştir :

Çizelge 9.1 : Düğme panelindeki düğmeler ve görevleri

Düğmeler ve Görevleri	
Taşı x	x ekseninde taşıma
Taşı y	y ekseninde taşıma
Taşı z	z ekseninde taşıma
Döndür x	x ekseninde döndürme
Döndür y	y ekseninde döndürme
Döndür z	z ekseninde döndürme
Büyüt x	x ekseninde ölçeklendirme
Büyüt y	y ekseninde ölçeklendirme
Büyüt z	z ekseninde ölçeklendirme

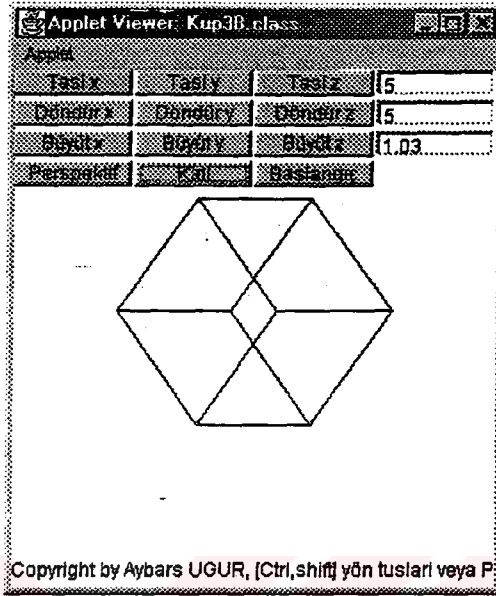
Küp, başlangıçta paralel izdüşümü alınarak çizdirilmektedir. Perspektif düğmesine basıldığında perspektif olarak görüntülenecektir ve düğme üzerindeki metin Paralel'e dönüşecektir (Şekil 9.3). Bu, düğmeye basıldığında küpün paralel izdüşüme göre görüntüleneceğini belirtmektedir. Perspektif izdüşüm, uzakta olan nesnelerin küçük, yakındakilerin daha büyük görüntülenmesini sağladığından dolayı gerçekçidir. Paralel izdüşümde ise nesnelerin uzaklığı büyüklüklerini

etkilemez. Teknik çizimlerde, örnek olarak makine parçalarının göreceli gerçek boyutlarının görüntülenmesinde de paralel izdüşüm kullanılabilir.



Şekil 9.3: Küpün perspektif izdüşüm alınarak çizdirilmesi sonucu oluşan görüntü

Küp, başlangıçta katı olarak her bir yüzeyi farklı bir renkte olacak şekilde çizilmektedir. Telkafes düğmesine basıldığında, küpün sadece sınır çizgileri çizilecektir ve düğme üzerindeki metin Katı'ya dönüşecektir (Şekil 9.4). Telkafes görünüm çoğunlukla nesnelerin genel görüntüsünü veren hızlı önizlemeler elde etmek için kullanılır. Katı görüntüler ise daha gerçekçidir ve küpün tam gösterimini sağlar. Çünkü, telkafes gösterimde örnek olarak hangi kenarların yüzey oluşturduğu kesin olarak belirlenemeyebilir.



Şekil 9.4 : Küpün telkafes (wireframe) gösterimi ile çizdirilmesi

Başlangıç düğmesi, küpü ilk konumuna getirir. Birçok taşıma, döndürme ve ölçeklendirme işleminden sonra küp orijinal konumundan çok uzaklaşabilir. Bu düğmeye basılarak küpün ilk konum, açı ve büyüklüğüne getirilmesi sağlanır.

Gerçekçi bir küp görüntüleme işlemi için, perspektif izdüşüme çeviren düğmeye basılarak küp perspektif izdüşüme göre çizdirilir. Ardından Ctrl+PgUp tuş bileşimine basılarak bakış noktasına yaklaştırılır (z ekseninde). Ctrl+PgDown tuş bileşimine basılarak uzaklaştırılır.

x, y, z eksenlerinde taşıma, döndürme ve ölçeklendirme işlemleri tuş bileşimleri yardımı ile de yapılabilmektedir. Tuşlara basıldığında ilgili işlem için artış miktarı değerleri aynen düğme panelinde olduğu gibi metin kutularından alınmaktadır. Kullanılan bazı tuş bileşimleri ve yaptıkları işlemler Çizelge 9.2’de belirtilmiştir :

Çizelge 9.2 : Kup3B yazılımında kullanılan tuşlar ve işlevleri

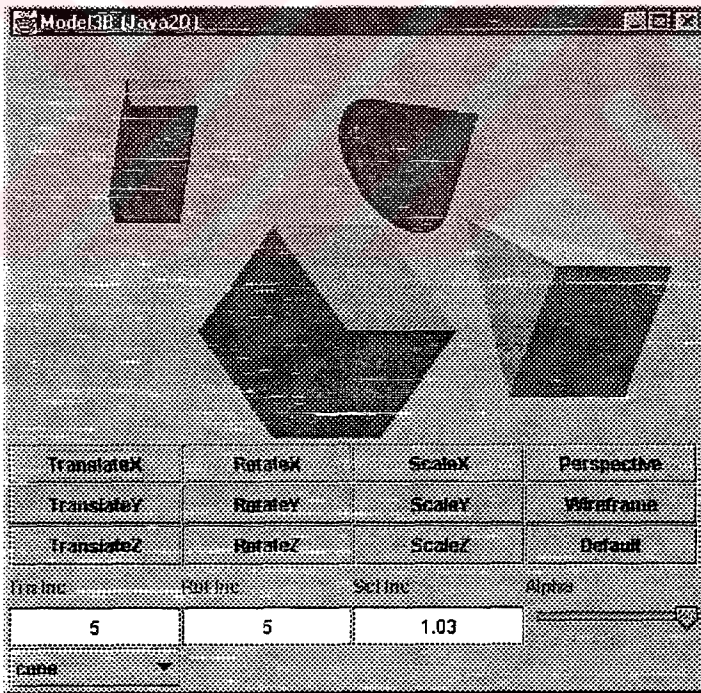
Tuşlar ve İşlevleri	
Sağ ok	x ekseninde pozitif yönde döndürme
Sol ok	x ekseninde negatif yönde döndürme
Üst ok	y ekseninde pozitif yönde döndürme
Alt ok	y ekseninde negatif yönde döndürme
PgUp	z ekseninde pozitif yönde döndürme
PgDn	z ekseninde negatif yönde döndürme
Ctrl+Sağ ok	x ekseninde pozitif yönde taşıma
Ctrl+Sol ok	x ekseninde negatif yönde taşıma
Ctrl+Üst ok	y ekseninde pozitif yönde taşıma
Ctrl+Alt ok	y ekseninde negatif yönde taşıma
Ctrl+PgUp	z ekseninde yakına taşıma
Ctrl+PgDn	z ekseninde uzağa taşıma
Shift+Sağ ok	x ekseninde büyütme
Shift+Sol ok	x ekseninde küçültme
Shift+Üst ok	y ekseninde büyütme
Shift+Alt ok	y ekseninde küçültme
Shift+PgUp	z ekseninde büyütme
Shift+PgDn	z ekseninde küçültme

Ayrıca Java’da kullanılan birtakım standart tuşlar vardır. Sekme (tab) tuşu bir sonraki düğme veya metin kutusunun etkinleştirilmesini sağlar. Boşluk tuşuna (ara çubuğu) basıldığında etkin tuş işletilir. Fare kullanılarak doğrudan ilgili düğmenin üzerine gelinip tıklanması yerine, düğmeye ulaşıp boşluk tuşuna basmak da aynı işlevi görür.

9.1.2 Model3B (Java 2D Sürümü)

Tez çalışması kapsamında geliştirilen ikinci yazılımda bir küp dışında sahneye farklı nesnelerin de eklenebilmesi ve nesnelere geçirgenlik (saydamlık) özelliği verilebilmesi sağlanmıştır (Şekil 9.5). Bu yazılımın işlevleri aşağıda verilmektedir :

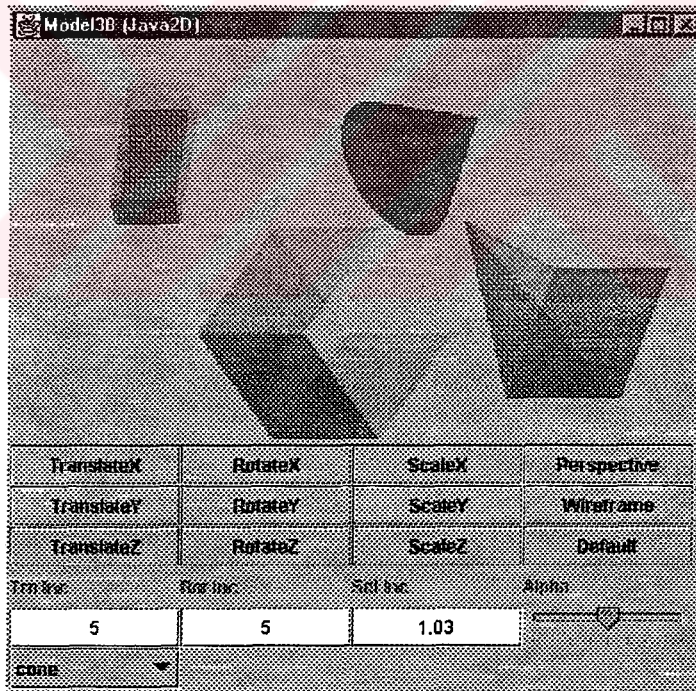
- Temel şekil (küp, piramit, koni) ekleme
- Şekiller üzerinde tüm eksenlerde dönüşüm işlemlerini gerçekleştirme
- Sahnede perspektif / paralel izdüşüm arasında geçiş
- Sahnede katı / telkafes görünüm arasında geçiş
- Sahnedeki nesnelere değişik oranlarda geçirgenlik özelliği verebilme



Şekil 9.5 : Model3B yazılımı Java 2D sürümü ile oluşturulmuş bir sahne

Küp ve piramit gibi çokgensel yüzeyli nesneler yanında koni de eklenebilmektedir. Koni için, eğrisel nesnelere çokgensel yüzey yaklaşımı yapılması benimsenmiştir. Nesnelere ayrı ayrı dönüşüm işlemleri uygulanabilmektedir. Sisteme farklı sayılarda nesneler eklenebilmektedir. “Alpha” değerinin değiştirilmesi ile nesnelere geçirgenlik özelliği verilebilmektedir (Şekil 9.6).

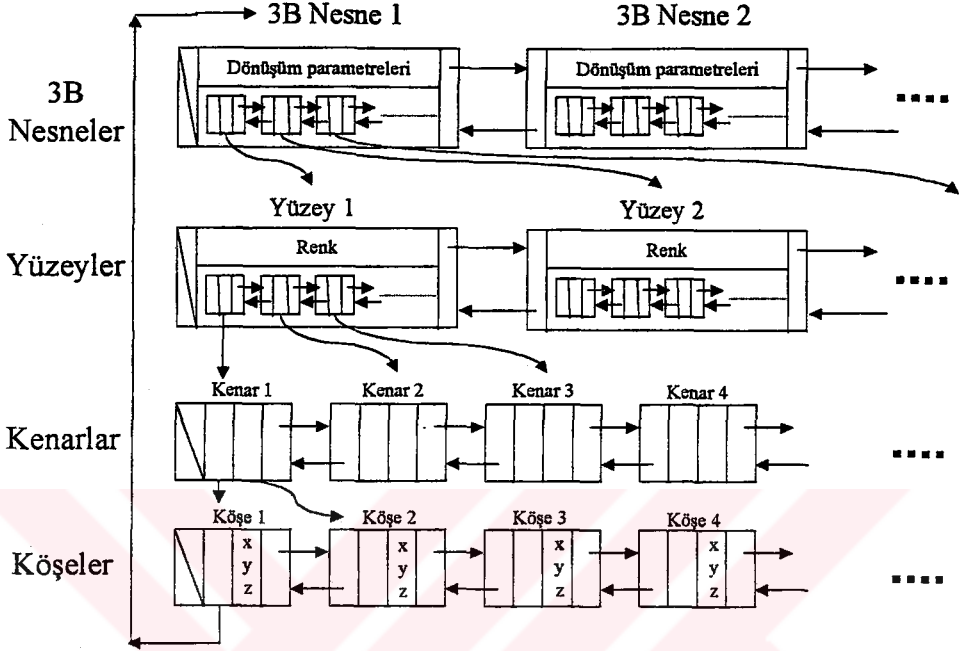
“Enter” tuşuna basılarak bir sonraki nesneye geçilebilmekte ve tüm nesneler üzerinde dönüşüm işlemleri ayrı ayrı uygulanabilmektedir. Dönüşüm işlemleri Kup3B yazılımında olduğu gibi hem paneldeki grafik kullanıcı arayüzü elemanları hem de klavye aracılığı ile gerçekleştirilebilmektedir.



Şekil 9.6 : Model3B ile nesnelere geçirgenlik özelliği verilmesi

Üç boyutlu nesne bilgileri, bu çalışma kapsamında tasarlanmış olan esnek ve genişletilebilir bir veri yapısına yerleştirilmektedir (Şekil 7.12). Bu veri yapısında üç boyutlu nesneler, yüzeyler, kenarlar ve köşeler, ayrı düzeylerde çift bağlı listeler olarak tutulmaktadırlar. Sisteme bir nesne ekleneceği zaman, önce nesnenin köşeleri, sonra köşeleri bağlayan kenarlar, daha sonra da kenarlardan oluşan yüzeyler eklenmektedir. Her bir düzeydeki elemanlar çift bağlı listelerden oluştukları için eleman ekleme çıkarma işlemleri kolaylıkla yapılabilir. Köşeler, konumlarını gösteren x, y, z değerlerinden oluşmaktadırlar. Ayrıca, bağlı bulundukları üç boyutlu nesne düğümünü gösteren bir bağları da vardır. Kenarlardan, kendilerini oluşturan iki köşeye bağlar bulunmaktadır. Bu şekilde bir kenarın uç noktalarına ulaşılabilir. Bir yüzey düğümünde, renk niteliğinin ve kendisini oluşturan kenarlara bağların yer aldığı bir çift bağlı liste tutulmaktadır. Bu sayede, kenar sayıları farklı çokgenlerden oluşan yüzeylere sahip nesnelerin de sisteme eklenebilmesi sağlanmıştır. Üç boyutlu nesneler düzeyindeki çift bağlı listedeki her bir düğümde ise ilgili nesnenin yerini, eğimini ve büyüklüğünü belirten dönüşüm parametrelerinin yanında, nesneyi sınırlayan yüzeylere bağların yer aldığı bir de çift bağlı liste tutulmaktadır (Şekil 9.7).

Üç Boyutlu Nesne Bilgilerini Tutan Veri Yapısı



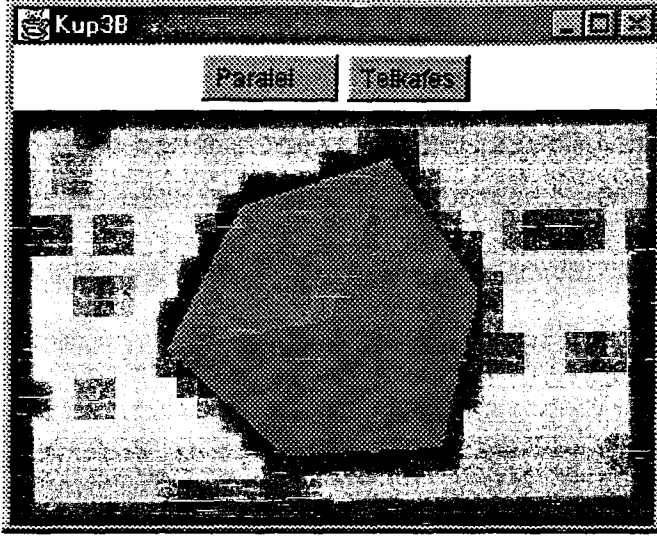
Şekil 9.7 : Sahnedeki üç boyutlu nesnelerin bilgilerini tutan veri yapısı

9.2 Java 3D Kullanılarak Geliştirilen Yazılımlar

Java 3D üç boyutlu uygulama geliştirme arayüzü kullanılarak geliştirilen Kup3B ve Model3B yazılımlarında, Java 3D'nin görüntü ağacı yapısı, yazılım geliştirme sürecini etkinleştirmiştir.

9.2.1 Kup3B (Java 3D sürümü)

Bu tez çalışmasında Java 3D kullanılarak geliştirilen bir applete internet üzerinden erişilerek bir küp üzerinde etkileşimli olarak fare yardımı ile dönüşüm işlemleri yapılabilir (Şekil 9.8).



Şekil 9.8 : Java 3D kullanılarak hazırlanan Kup3B programının pencere görünümü

Küp fare yardımı ile üç boyutlu uzayda değişik eksenlerde döndürülebilmektedir. Küp üzerinde tanımlı fare hareketleri ve işlevleri Çizelge 9.3'te verilmiştir.

Çizelge 9.3 : Kup3B yazılımı Java 3D sürümünde küp üzerinde uygulanabilen fare hareketleri ve işlevleri

Fare Hareketleri	İşlevi
Farenin sol tuşu ile sola/sağa sürükleme	y ekseninde döndürme
Farenin sol tuşu ile yukarı/aşağı sürükleme	x ekseninde döndürme
Farenin sağ tuşu ile sola/sağa sürükleme	x ekseninde taşıma
Farenin sağ tuşu ile yukarı/aşağı sürükleme	y ekseninde taşıma
Alt+Farenin sol tuşu ile yukarı/aşağı sürükleme	z ekseninde taşıma

Paneldeki ilk düğme paralel/perspektif izdüşüm arasında, diğeri ise telkafes/katı görünüm arasında geçiş yapılabilmesini sağlamaktadır.

Aşağıda, küpün oluşturularak sahneye eklenmesine ve küp üzerinde fare etkileşiminin tanımlanmasına ilişkin kod parçaları görülmektedir :

```
// Bir dönüşüm grubu oluşturulup özelliklerinin belirlenerek sisteme
eklenmesi
    TransformGroup objTrans = new TransformGroup();
    objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
    objTrans.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
    objRoot.addChild(objTrans);

// Üç boyutlu bir küp nesnesinin oluşturulması ve Java 3D görüntü ağacına
yerleştirilmesi
    objTrans.addChild(new ColorCube(0.4));
    BoundingSphere bounds =
        new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);

// Kullanıcının fare kullanarak nesnenin dönüşünü kontrol etmesini
sağlayan düğümün oluşturulması
    MouseRotate behavior1 = new MouseRotate();
    behavior1.setTransformGroup(objTrans);
    objTrans.addChild(behavior1);
    behavior1.setSchedulingBounds(bounds);

// Kullanıcının fare kullanarak nesneyi z ekseninde taşımalarını sağlayan
düğümün oluşturulması
    MouseZoom behavior2 = new MouseZoom();
    behavior2.setTransformGroup(objTrans);
    objTrans.addChild(behavior2);
    behavior2.setSchedulingBounds(bounds);

// Kullanıcının fare kullanarak nesneyi (x,y) eksenlerinde taşımalarını
sağlayan düğümün oluşturulması
    MouseTranslate behavior3 = new MouseTranslate();
    behavior3.setTransformGroup(objTrans);
    objTrans.addChild(behavior3);
    behavior3.setSchedulingBounds(bounds);
```

Bu yazılım aynı işlemlere sahip olan Java 2D sürümünden çok daha kısadır ve çok daha kısa sürede yazılmıştır. Bunun nedenleri;

- Görüntü ağacının esnek yapısı sayesinde yeni nesnelerin eklenip çıkarılmasının (ve nesnelere nitelik verilmesinin) kolay olması.

- Bilgileri tutan bir veri yapısının tasarlanıp gerçekleştiriminin yapılmasının gerekmeyişi, ve
- Özellikle programlara fare etkileşimi sağlayan standart özelliklerin doğrudan eklenebilmesidir.

Ayrıca Java 3D API'de fare etkileşimi standart olduğundan, daha etkin ve rahat kullanım söz konusudur. Java 3D kullanımı ile etkin olmayan tasarımların önüne geçilmiş olmaktadır.

9.2.2 Model3B (Java 3D sürümü)

Tez çalışmasında Java 3D Uygulama geliştirme arayüzü kullanılarak geliştirilen Model3B adlı yazılımın bazı özellikleri şunlardır :

1. Temel 3B Nesne Ekleme
2. Hazır 3B Model Ekleme
3. 3B Metin Ekleme
4. Nesnelere Nitelikler Verme
5. Desen Kaplama
6. Etkileşimli Dönüşüm
7. Paralel/Perspektif İzduşüm
8. Kamera Hareketleri
9. Basit Gruplandırma
10. Basit Aydınlatma

Yazılım yerel olarak appletviewer gibi bir applet görüntüleyici ile veya uygulama olarak çalıştırılabildiği gibi, internet üzerinden de “Netscape” ve “Internet Explorer” gibi web tarayıcıları ile görüntülenebilmektedir.

Yazılım aracılığı ile liste kutularından seçilerek sahneye eklenebilen temel 3B nesneler, 3B ilkeller (silindir, koni, küre, kutu) ve yüzeyleri farklı renklerde 4 yüzlü, küp, 8 yüzlü, 20 yüzlü gibi şekillerdir.

Sahneye, Java 3D’nin doğrudan desteklediği “Wavefront” OBJ 3B dosya biçimindeki nesneler de eklenebilmektedir. Yerel makinede çalıştırıldığında hazır nesneler ikincil bellek ortamlarından, internet üzerinde çalıştırıldığında internet üzerinden alınabilmektedir. Yazılımın önemli orijinal özelliklerinden birisi de kullanım sırasında dinamik olarak internet üzerinden hazır 3B model eklenmesini ve etkileşimli sanal dünyalar oluşturulmasını sağlamaktır. Kullanıcı, herhangi bir nesnenin üç boyutlu modelini, 3D Studio Max ve AutoCAD gibi istediği bir grafik programı ile oluşturup internet üzerinden ücretsiz temin edilebilen dönüştürücü bir program ile “obj” formatına dönüştürebilmektedir. Böylece modelin oluşturulacağı yazılımın seçilmesine ilişkin sınırlandırma ortadan kalkmaktadır.

Kullanıcının, oluşturacağı sahneye nesnelerin yanı sıra üç boyutlu yazıları da ekleyebilmesi sağlanarak çeşitlilik artırılmıştır.

Sahneye eklenmiş olan herhangi bir nesne bir liste kutusundan seçilerek istenen bir anda silinebilmektedir. Ayrıca tüm nesnelerin bir anda silinmesini sağlayan olanaklar da eklenmiştir.

Üç boyutlu nesneler sahneye eklenmeden önce renk (color), doku (texture) ve geçirgenlik (transparency) özellikleri verilebilmektedir. Nesnelere ayrıca resim de kaplanabilmektedir. Örnek olarak bir küreye dünya resmi kaplandığında küresel gerçekçi bir dünya oluşmaktadır.

Nesneler tek tek seçilerek fare yardımı ile üç boyutlu uzayda istenen konuma taşınabilmekte (sağ tuş) ve istenen eksen etrafında döndürülerek istenen eğim verilebilmektedir (sol tuş).

Nesneler gruplandırılarak sahneye eklenebilmektedir. Böylece grup içindeki nesnelerin hepsi aynı anda taşınıp döndürülebilir. Sistem şu an tek grubu desteklemektedir.

Seçim düğmeleri yardımı ile herhangi bir anda paralel/perspektif izdüşüm arası geçiş yapılabilmektedir. Ok tuşları ile kamera ileri ve geri hareket ettirilebilmekte ve sola ve sağa döndürülebilmektedir.

Sahnede ışık kaynakları ile uygun bir aydınlatma sağlanmıştır. Genel aydınlık ayarlanmış, yönlü ve noktasal ışık kaynakları eklenmiştir.

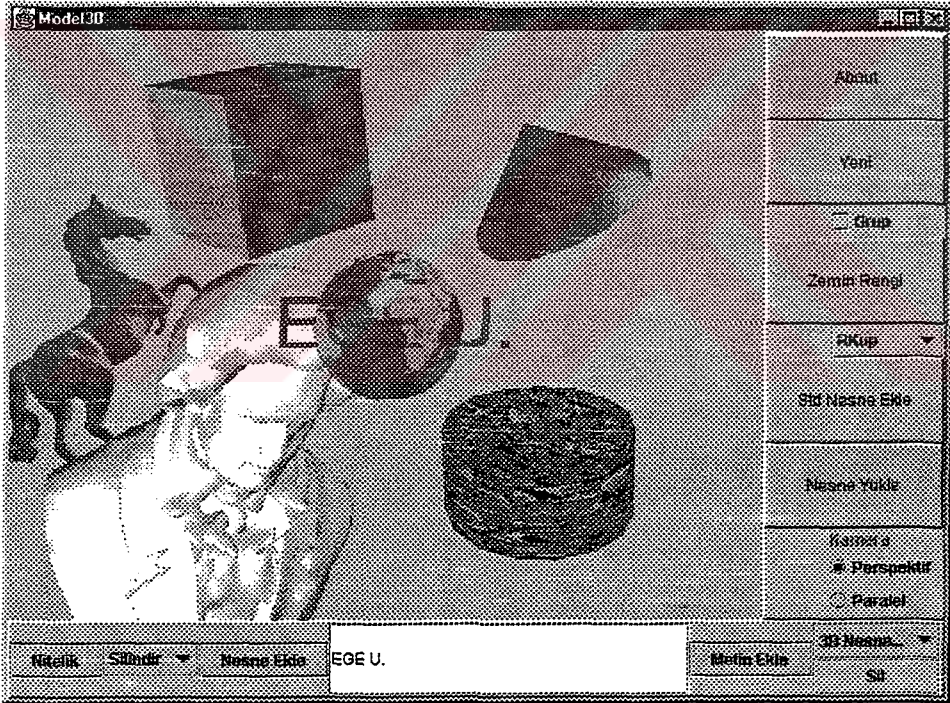
Grafik kullanıcı arayüzünün tasarımında Java 2 platformu ile gelen “Swing” bileşenlerinden yararlanılmıştır. Pencerenin sağ tarafına ve alt tarafına birer panel yerleştirilmiştir.

Alt taraftaki panelde sırası ile “Nitelik” düğmesi, eklenecek nesnelerin (silindir, koni, küre, kutu) seçildiği açılan liste kutusu, “Nesne Ekle” düğmesi, eklenecek metnin yazıldığı metin kutusu, “Metin Ekle” düğmesi, sahnede bulunan nesnelerin isimlerini tutan liste kutusu ve bu listede seçili nesneyi silmeyi sağlayan “Sil” düğmesi yer almaktadır. “Nitelik” düğmesine basıldığında “Renk” ve “Geçirgenlik” vermeyi sağlayan düğmelerin ve hazır dokuları içeren liste kutusunun yer aldığı pencere açılır. Renk, “Swing” JColorChooser penceresinden seçilebilmekte ve geçirgenlik de ölçekten ayarlanabilmektedir.

Sağ taraftaki panelde ise, yukarıdan aşağıya sırası ile “About” ve “Yeni” düğmeleri, gruplandırmayı sağlayan “Grup” onay kutusu, “Zemin Rengi” düğmesi, eklenebilen standart nesnelerin yer aldığı açılan liste kutusu, “Std Nesne Ekle” düğmesi, “Nesne Yukle” düğmesi ve Paralel/Perspektif izdüşüm geçişini sağlayan seçim düğmeleri yer

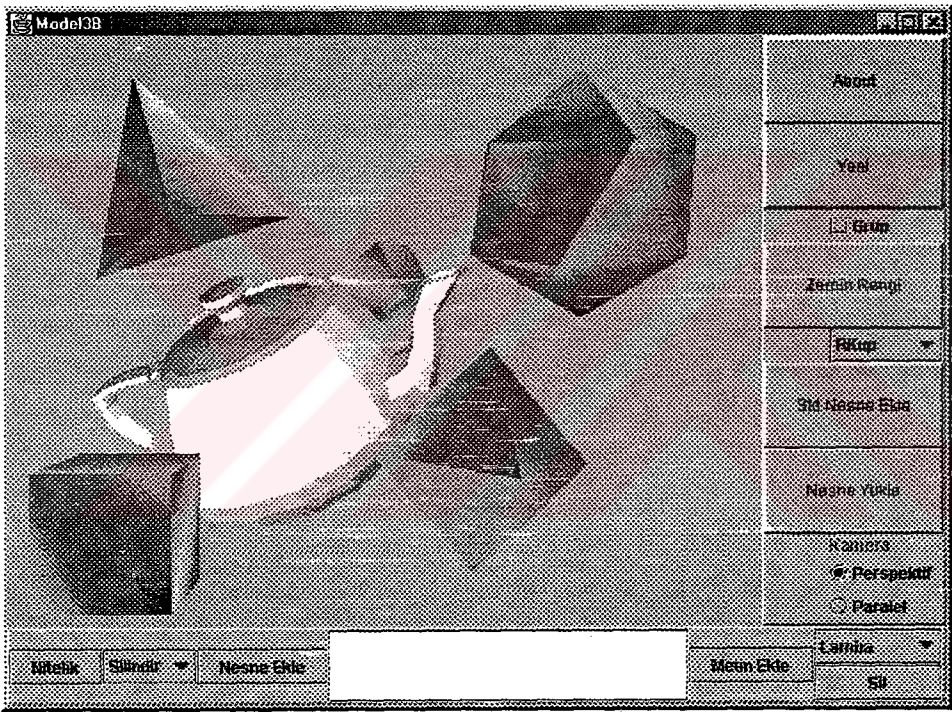
almaktadır. Zemin rengi düğmesine basıldığında seçim sağlayan hazır pencere gelmektedir. Nesne yükle düğmesi ile gelen liste kutusu / dosya aç penceresinden sahneye hazır model yüklenebilmektedir.

Yazılımın bazı özelliklerini özetleyen bir pencere görünümü aşağıdaki gibidir (Şekil 9.9). Pencerede sol üst köşede her yüzeyi farklı renkte olan bir küp şekli, sağda mor renkli bir koni şekli, ortada üç boyutlu “Ege U.” yazısı, sol alt köşede “Beethoven” büstü modeli ve at heykeli modeli, sağ alt köşede taştan yapılmış bir silindir görülmektedir. Ortada ayrıca yarı saydam bir dünya şekli görülmektedir.



Şekil 9.9 : Java 3D kullanılarak hazırlanan Model3B adlı yazılımın pencere görünümü

Değişik geometrik nesneler de sisteme eklenebilmektedir. Şekil 9.10'da sol üst köşede dörtyüzlü, sağ üst köşede yirmiyüzlü, sağ alt köşede sekizyüzlü ve sol alt köşede de bir çaydanlığın kameraya göre önündeki ve arkasındaki küp nesneleri görülmektedir. Çaydanlık, 3D Studio Max ile tasarlanarak ve üç boyutlu dosya dönüştürücü bir program yardımı ile OBJ formatına çevrilerek programa yüklenmiştir. Nesnelerin arkasında kalan başka nesne parçaları görülmemektedir.



Şekil 9.10 : Model3B yazılımının üç boyutlu grafiksel içerik eklemeye ilişkin bazı özelliklerini gösteren pencere görünümü

Model3B yazılımının kullanımının kolay olduğu açıkça görülmektedir. Herhangi bir internet kullanıcısı istediği üç boyutlu sahneleri oluşturup nesnelerle etkileşim kurabilmektedir.

9.3 Model3B Yazılımının Kullanım Alanları

Model3B yazılımına internetten erişilmesiyle, hazırlanmış modeller sisteme alınıp üç boyutlu olarak incelenebilmektedir. Yazılar ve temel şekillerle veya başka modellerle bütünleştirilebilmektedir. Bu yazılımı alan bir şirket, ürünlerinin modellerini hazırlayarak web sayfasına yerleştirdiğinde, html kodunu ayarlayarak ürünlerini internet üzerinden incelenebilir hale getirebilmektedir. Erişen kullanıcılar, ürün incelemenin yanı sıra üç boyutlu sanal dünyalar oluşturabilme amacıyla da yazılımı kullanabilmektedirler.

Model3B yazılımı üzerinde bazı değişiklikler yapılarak, özellikle

- E-ticaret ve e-reklam,
- Uzaktan öğrenme,
- Eğlence,
- Kültür ve sanat

ağırlıklı web sayfalarının üç boyutlu etkileşimli grafiklerle zenginleştirilmesi sağlanabilir. Özellikle ürünlerini tanıtan firmalar, ürünlerinin üç boyutlu modellerini internet sayfalarına yerleştirerek bu tür bir program aracılığı ile tüketicilerin karmaşık olan veya olmayan ürünlerle etkileşim kurmasını sağlayarak özelliklerini anlamasına katkıda bulunurlar. Özellikle araba ve diğer taşıtlar, elektronik (fotoğraf makinesi, müzik seti, bilgisayar, cep telefonu) ve beyaz eşyalar, diğer cihazlar (matkap), mobilyalar, spor malzemeleri, gıda ve tekstil malzemeleri, giyecekler, oyuncaklar modellenerek tüketicilerin incelemesi amacı ile internet üzerine yerleştirilebilir. Ayrıca kültürel ve bilimsel amaçlı olarak tarihi eserler, sanat eserleri, bitki ve hayvanlar ve kimyasal maddeler de modellenerek internette bu tür konuları öğrenmek isteyen internete bağlı bir PC'ye sahip tüm kullanıcılara sunulabilir.

Gerçekte dokunularak incelendiğinde zarar görebilecek cam eşyalar, tarihi eserler, sanat eserleri camekanlı bölmelere konulduğunda yeterince tanıtılamamaktadır. Modelleri bir kere internete yerleştirilen bu tür nesneler orjinallerine zarar verilmeksizin bilgisayar aracılığı ile internet üzerinden dünyanın dört bir tarafından etkin bir şekilde üç boyutlu olarak incelenebilir hale gelmektedir. İncelenmesi tehlikeli olabilecek maddeler ve belli bir coğrafyada bulunan taş türleri de risksiz ve maliyetsiz olarak çaba harcamadan incelenebilecektir. Yazılımda yapılması planlanan ekler, Öneriler bölümünde yer almaktadır.

Modelleme işlemi 3D Studio Max (*.3ds), AutoCAD (*.dxf), VRML (*.wrl) gibi popüler grafik programları aracılığı ile yapılarak oluşturulan modeller, "Crossroads" gibi bir üç boyutlu dosya biçimi dönüştürücü programı yardımı ile "OBJ" biçimine dönüştürülüp sisteme eklenerek görüntülenebilmekte ve etkileşim kurulabilmektedir.

9.4 Yazılım Geliştirme Süreci Sonuçları

İncelemeler ve yapılan bu çalışmalar ışığında ulaşılan sonuçlar Bölüm 10.4'te yer almaktadır. Üç boyutlu grafik uygulama geliştirme arayüzlerinin özellikleri, bilgisayar grafikleri alanına etkileri, beklenen genel etkileri, kullanımı ve belirlenen eksikleri, sonuçlar bölümünden incelenebilir. İzleyen bölümlerde, Kup3B ve Model3B yazılım geliştirme süreçlerinden alınan sonuçlara bakılarak yapılan karşılaştırmalar ve elde edilen grafikler sunulmuştur.

9.4.1 Kup3B Yazılımının Gerçekleştirilmesinde Java 2D ve Java 3D Karşılaştırması

Kup3B yazılımı iki farklı yöntemle gerçekleştirilmiştir : Java 2D API kullanılarak ve Java 3D API kullanılarak. Her iki sürümünün de 3 adet işlevi vardır :

- Kullanıcıların bir küp nesnesi üzerinde dönüşüm işlemlerini yapmalarını sağlamak,
- Görüntülemeye paralel/perspektif izdüşüm arasında geçiş yapmalarını sağlamak,
- Katı ve Telkafes görünümüler arasında geçiş yapmalarını sağlamak.

Bu yazılımların geliştirilmesi sürecinde elde edilen bilgiler karşılaştırmalı olarak Çizelge 7.4’te yer almaktadır. Elde edilen süre değerleri, yapılan son düzenlemeler için harcanan süreleri kapsamamaktadır.

Çizelge 9.4 : Kup3B yazılımlarının geliştirilmesinde Java 2D API ve Java 3D API karşılaştırılması

Kriter	Java 2D	Java 3D
Süre Analiz,Tasarım,Kodlama	300 dakika	15 dakika
Hatalardan Arındırma Süresi	110 dakika	2 dakika
Kaynak Kod Satır Sayısı	500 satır	200 satır
Programlamadaki Zorluk ve Önşartlar	Güçlü bir matematik ve programlama alt yapısı gerektiriyor	Java 3D API özelliklerini bilmek gerekliyor
Kullanım Kolaylığı	Kullanıcılar yapısına kolay alışıyor	Java 3D standart tuş ve fare özellikleri ile evrensel
Hız	İyi	İyi

Aynı yazılımın iki farklı yöntemle gerçekleştirilmesi ve doğrudan ölçütlerden olan kod satır sayılarının ölçülerek karşılaştırılması ile, yazılımları geliştirme sürecindeki hızlanma, yazılım geliştirmede harcanan çaba ve maliyetlerdeki azalma da ölçülmüş olmaktadır. Bu da üç boyutlu bilgisayar grafiklerinin günlük yaşantımızın doğal bir parçası gelmesi açısından önemlidir. Şirketler ve yazılım geliştiriciler üç boyutlu grafik yazılımları geliştirmede 3B API'lerin yazılım geliştirme sürecini hızlandırıp kolaylaştırdığını ifade etmektedirler. Bu çalışma da “ne kadar” sorusunun yanıtını aramak açısından değerlidir. Tez kapsamında yapılan araştırmalarda, benzeri bir çalışmaya rastlanmamıştır.

Burada üç boyutlu grafik programlamada 2B API ve 3B API kullanımı arasındaki karşılaştırma yapılmıştır. Analiz, tasarım, kodlama ve hatalardan arındırma süreleri ve üretilen programların kaynak kod satır sayıları ölçülmüştür. Üretilen programlar, kullanım kolaylığı ve hız açılarından karşılaştırılmışlardır. Farklı düzeydeki programcıların benzeri bir programı geliştirmek için harcayacağı süre ve kodlayacağı satır sayıları da farklı olacaktır. Fakat, bu tablodan da görüleceği gibi, üç boyutlu grafik programlamada yüksek düzeyli uygulama geliştirme arayüzlerinin kullanımının yararlı olduğu sonucu değişmeyecektir. Daha güç bir yazılımın geliştirilmesi durumunda da iki boyutlu ve üç boyutlu uygulama geliştirme arayüzleri kullanımının getireceği çaba, maliyet ve kalite farkının daha da açılacağı kesindir.

Benzeri işlemlere sahip olan ve tez çalışması kapsamında Java 2D ve Java 3D ile geliştirilen iki yazılımın bölümlerine göre kod satır sayıları Çizelge 9.5'te verilmiştir. Kod satır sayılarının ölçümünün tercih edilmesinin nedeni yazılım mühendisliğinde bilinen, yaygın ve doğrudan bir yöntem olması ve birçok kişi tarafından kolayca yorumlanabilmesidir. Çizelgedeki bölümlerde dikkate alınan etkinlikler aşağıdaki gibidir;

- **Arayüz** : Ekran pencerelerinin hazırlanması, düğmeler, metin kutuları ve diğer pencere bileşenlerinin oluşturularak eklenmesi, bu tür bileşenler ile, klavye ve fare ile etkileşim için oluşacak işlem kodları
- **Modelleme** : Veri Yapılarının oluşturulması, sahnenin ve üç boyutlu nesnelerin oluşturulması ve veri yapılarına yerleştirilmesi
- **Dönüşüm** : Matris çarpım metotları, taşıma, ölçeklendirme ve döndürme gibi dönüşüm işlemlerini gerçekleştiren metotlar ile dönüşüm grupları ile ilgili işlemler
- **Görüntüleme** : Üç boyutlu nesnelerin veri yapılarından alınarak görüntülenmesi için yapılan tanımlama ve işlem kodları
- **Diğer** : Genel açıklamalar, standart programlama dili deyimleri, genel amaçlı veya sınıflandırılmayan deyim satırları.

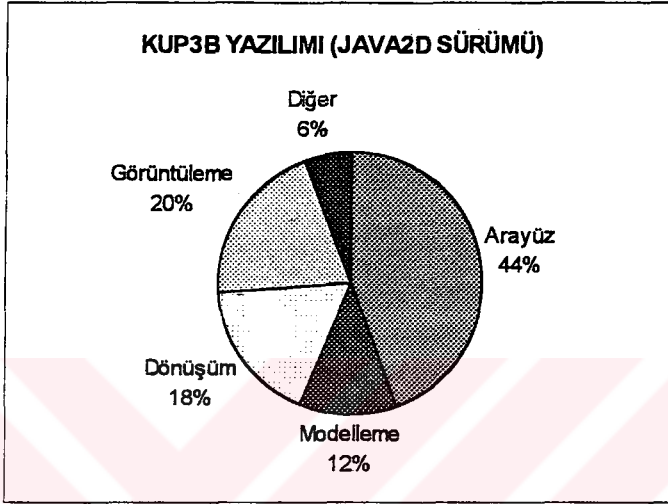
Çizelge 9.5 : Kup3B yazılımının (Java 2D ve Java 3D sürümleri) bölümlerine göre kaynak kodlarındaki satır sayıları.

Kup3B Yazılımının Bölümleri (Java 2D Sürümü)	
Bölüm	Satır Sayısı
Arayüz	221
Modelleme	59
Dönüşüm	89
Görüntüleme	102
Diğer	29
Toplam	500

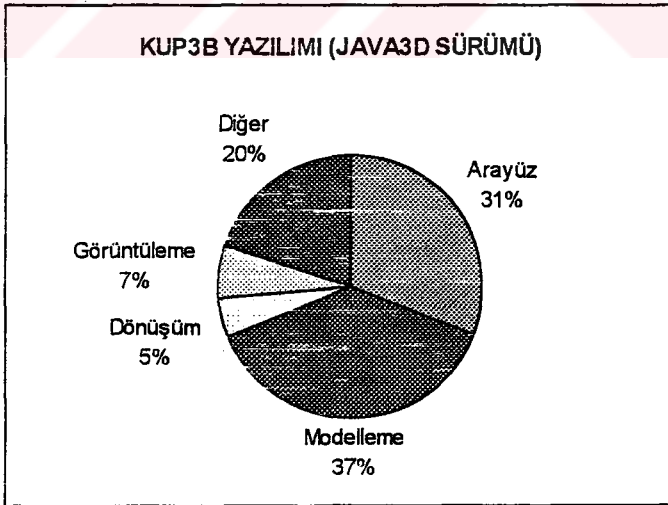
Kup3B Yazılımının Bölümleri (Java 3D Sürümü)	
Bölüm	Satır Sayısı
Arayüz	62
Modelleme	75
Dönüşüm	10
Görüntüleme	13
Diğer	40
Toplam	200

Deneyimli tüm grafik yazılımcıların geliştirdikleri yazılımların kod satır sayıları tablodaki değerlerden çok farklı olmayacaktır.

Şekil 9.11’de Java 2D, Şekil 9.12’de Java 3D sürümü için Kup3B kaynak kodundaki bölümlerin satır sayısı yüzdeleri gösterilmektedir.



Şekil 9.11 : Kup3B yazılımı kaynak kodundaki bölümlerin satır sayısı yüzdeleri



Şekil 9.12 : Kup3B yazılımı kaynak kodundaki bölümlerin satır sayısı yüzdeleri

Basit bir küp nesnesi diziler kullanılarak modellenenilmektedir. Bu nedenle, Java 2D kullanıldığında karmaşık veri yapıları kullanmak gerekmediğinden pasta grafikte modelleme için yazılan kod oranı düşüktür. Java 3D kullanıldığında ise, görüntü ağacı ile ilgili işlemler ve küp nesnesinin ilk durumu belirtilerek eklenmesi gerektiğinden modellemede kazanç olmamaktadır.

Üç boyutlu nesneler için fare etkileşim özelliklerinin doğrudan gelmesi, arayüzde bu amaçla yazılması gereken kod satır sayısını büyük ölçüde azalttığı gibi, tüm yazılımlarda belli bir standardı da sağlamakta ve yazılım kalitesini de artırmaktadır. Ayrıca karmaşık bir panel gerekliliğini de ortadan kaldırmıştır. Java 2D’de ise anlaşılır ve hatasız bir panel ve tuş bileşimi arayüzü için çok sayıda satırdan oluşan kod yazmak gerekmiştir.

Görüntüleme işlemi için ise özellikle gerçek koordinatların ekran koordinatlarına dönüşümü ve tüm yüzeylerin çizdirilmesi Java 2D’de önemli yer tutmaktadır. Java 3D bu olanağı doğrudan sunmaktadır. İki boyutlu arayüzlerde de matris işlemleri ve üç boyutlu dönüşüm işlemleri için standart bir kod yazmak gerekmektedir.

Genel olarak bakıldığında basit bir üç boyutlu programda üç boyutlu bir arayüzün kullanılmasıyla, yazılan toplam kod satır sayıları dikkate alındığında bu çalışma için $500/200 = 2.5$ oranı ortaya çıkmaktadır. Başka bir deyişle, grafik yazılımları geliştirmek için harcanan çaba yüksek düzeyli üç boyutlu bir uygulama geliştirme arayüzünden yararlanıldığında 60% azalarak 40%’ına inmiştir. İki boyutlu uygulama geliştirme arayüzleri yerine üç boyutluların tercih edilmesinin getireceği bu yarar, maliyetlere ve yazılım geliştirme süresine de doğrudan yansımaktadır. Tüm yazılımlarda verimin bu derece artması beklenemez. Bunu etkilemesi beklenen faktörler şöyle sıralanabilir:

- Geliştirilecek yazılımın doğası, verimliliği ve yazılım geliştirme süresini etkileyecektir. Bazı grafik yazılımları, doğrudan grafik, modelleme, boyama, çizim gibi işlemleri yaptıran yazılımlar olabildiği gibi, bazı yazılımlar da diğer alanlarda olup, içinde grafiklerin kullanımını içeren yazılımlardır. Yazılımın içeriğine ve amacına göre bu oran değişecektir.
- Yazılımın karmaşıklık düzeyi ve grafiklerdeki gerçekçiliğin düzeyi de önemli bir faktördür. Geliştirilecek yazılımın grafiksel gerçekçilik beklentisi de bu oranı değiştirecektir.
- Yüksek düzeyli üç boyutlu uygulama geliştirme arayüzlerinin ileri işlevlerinin kullanımının öğrenilmesi zordur. Yazılım geliştiricilerin bilgi ve deneyim farklılıkları, harcanan çaba, yazılım geliştirme süresi ve maliyetlerdeki bu oranı etkileyecektir.

Yazılımın karmaşıklık düzeyi arttığında oranların “ne kadar?” değişeceğinin belirlenmesi de önemli bir konudur. İzleyen bölümde, daha çok işleve sahip yazılımların karşılaştırılması bu amaçla yapılmıştır.

9.4.2 Model3B Yazılımının Geliştirilmesinde Java 2D ve Java 3D Karşılaştırması

Model3B yazılımı (Java 2D sürümü) Kup3B yazılımının sahip olduğu üç temel işleve ek olarak şu iki işlevi de gerçekleştirmektedir :

- Nesnelere Nitelikler Verme
- Temel 3B Nesne (küp, silindir, koni) Ekleme

Model3B yazılımı (Java 3D sürümü) ise bu 5 işleve ek olarak 5 işlev daha sunmaktadır :

- Hazır 3B Model Ekleme
- 3B Metin Ekleme

- Desen Kaplama
- Basit gruplandırma
- Basit aydınlatma

Model3B Yazılımı Java 2D sürümü toplam 5, Java 3D sürümü ise toplam olarak 10 işleve sahiptir. Geliştirilen iki yazılımın bölümlerine göre kod satır sayıları Çizelge 9.6’de verilmiştir.

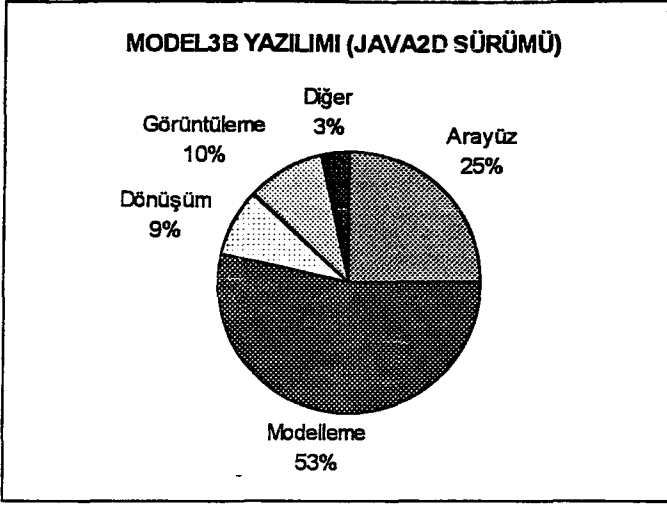
Şekil 9.13’te Java 2D sürümü için, ve Şekil 9.14’te Java 3D sürümü için Model3B yazılımının kaynak kodundaki bölümlerin satır sayısı yüzdeleri gösterilmektedir.

Çizelge 9.6 : Model3B yazılımının bölümlerine göre kaynak kodlarındaki satır sayıları.

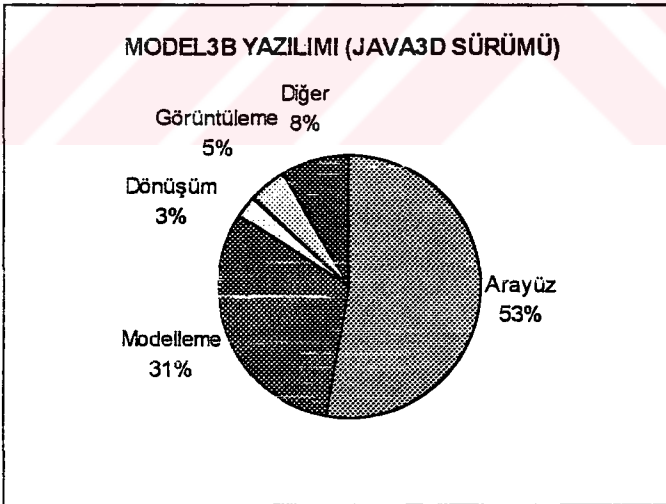
Model3B Yazılımı (Java 2D Sürümü)	
Bölüm	Satır Sayısı
Arayüz	390
Modelleme	830
Dönüşüm	135
Görüntüleme	151
Diğer	53
Toplam	1559

Model3B Yazılımı (Java 3D Sürümü)	
Bölüm	Satır Sayısı
Arayüz	421
Modelleme	250
Dönüşüm	21
Görüntüleme	38
Diğer	66
Toplam	796

Çizelge incelenecek olursa kullanıcı arayüzü için yazılan kod miktarları arasında Java 2D ve Java 3D için pek fark olmadığı görülecektir. Buna göre geliştirilecek yazılımın ölçeği büyüdükçe kullanıcı arayüzleri için yazılan kod miktarları arasında pek fark kalmadığı anlaşılmaktadır.



Şekil 9.13 : Model3B yazılımı (Java 2D sürümü) kaynak kodundaki bölümlerin satır sayısı yüzdeleri



Şekil 9.14 : Model3B yazılımı (Java 3D sürümü) kaynak kodundaki bölümlerin satır sayısı yüzdeleri

Model3B yazılımının Java 2D sürümü 5, Java 3D sürümü ise 10 farklı işlevi gerçekleştirebilmektedir. Model3B yazılımı Java 2D sürümü yaklaşık 1550 satır, Java 3D sürümü yaklaşık 800 satır kaynak kod içermektedir. Her iki yazılım da yaklaşık 50'şer saatlik yoğun bir geliştirme süreci sonucunda tamamlanmıştır. Bu duruma göre eşit sürelerde çalışıldığında orta ölçekli bir üç boyutlu grafik yazılımında üç boyutlu yüksek düzeyli bir API, yaklaşık 2 kat performans sağlamaktadır. Yazılım geliştiriciye, geliştirilen yazılımın niteliğine ve kullanılan API'in niteliklerine bağlı olarak bu oran değişecektir. Ayrıca, eşit sürelerde çalışıldığı halde Java 3D kullanıldığında geliştirilen yazılımın kod satır sayısı diğerinin yarısıdır. Bu da üç boyutlu API'lerin grafik yazılım geliştirmedeki başarısını göstermektedir.

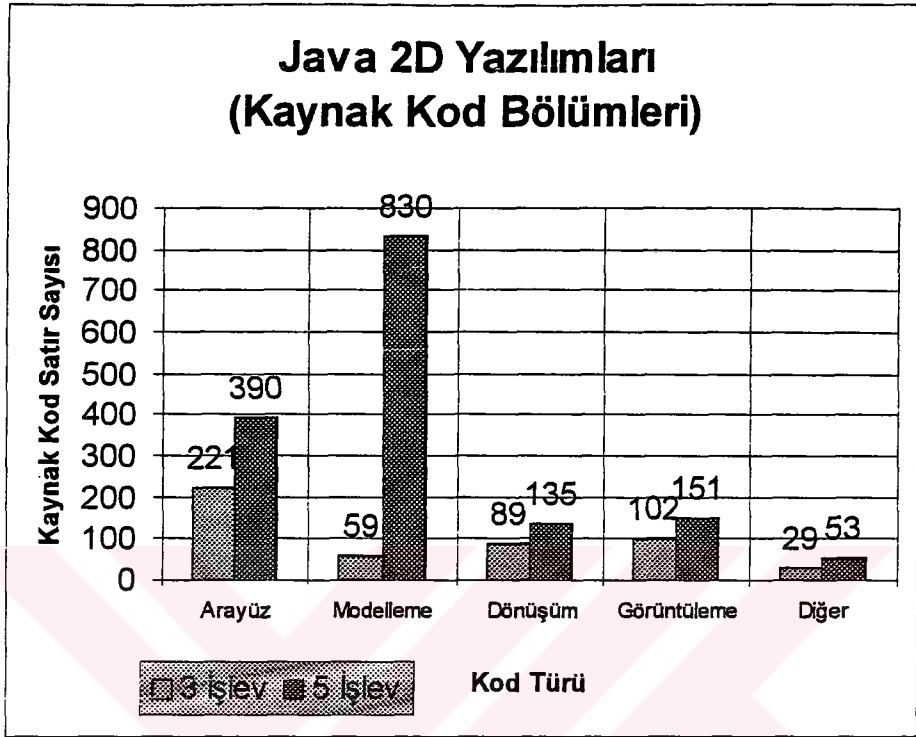
İki boyutlu uygulama geliştirme arayüzleri ile üç boyutlu grafik programlamada harcanan çabaların ve zamanın çoğu, nesneleri tutacak genel veri yapılarının tasarlanması, sisteme eklenecek üç boyutlu şekillerin oluşturulması, sistemdeki şekillerin zamana göre hareketlendirilmesi veya klavye ve fare gibi çevre aygıtları ile hareketlendirecek etkileşimli bir ortamın oluşturulmasıyla geçmektedir. Bu etkinliklere dikkat edilecek olursa, yazılım geliştiricilerin senaryodan çok modelleme ile uğraştıkları görülmektedir. Java 3D'de *görüntü ağacı* adlı yapı hazırdır. Bu yapı ile, sisteme nesneye yönelik bir şekilde üç boyutlu şekil eklemek ve nitelik vermek, şekilleri silmek, gruplandırmak, şekilleri etkileşimli olarak fare ile yardımı ile hareket ettirebilir hale getirmek, kamera özelliklerini ayarlamak gibi işlevler basit komutlarla gerçekleştirilebilmektedir. Burada yazılım geliştiriciler sadece içeriğe odaklanmaktadırlar. Asıl zorluk ise görüntü ağacı sisteminin doğru anlaşılması ve Java 3D komutlarının bir işi yaptırabilmek için birbirleri ile çelişmeyecek şekilde ayarlanmasıdır. Her komutun birçok parametresinin olması dikkatli olmayı gerektirmektedir.

Modellenecek nesnelerdeki çeşitlilik, yazılımların içermesi gereken iki boyutlu uygulama geliştirme arayüzlerinde, modellemenin altyapısının oturtulmasını ve daha karmaşık görüntüleme ve dönüşüm işlemlerini gerektirmektedir. Model3B yazılımlarına eğrisel nesneler de katıldığından modelleme için yazılan kod sayıları Java 2D ve Java 3D kullanımlarında büyük farklılıklar göstermektedir. Java 3D kullanıcılar için özellikle hazır olanakları ve görüntü ağacı sayesinde modellemeyi kolaylaştırarak, içerik ve senaryoya yoğunlaşılmasını sağlamaktadır.

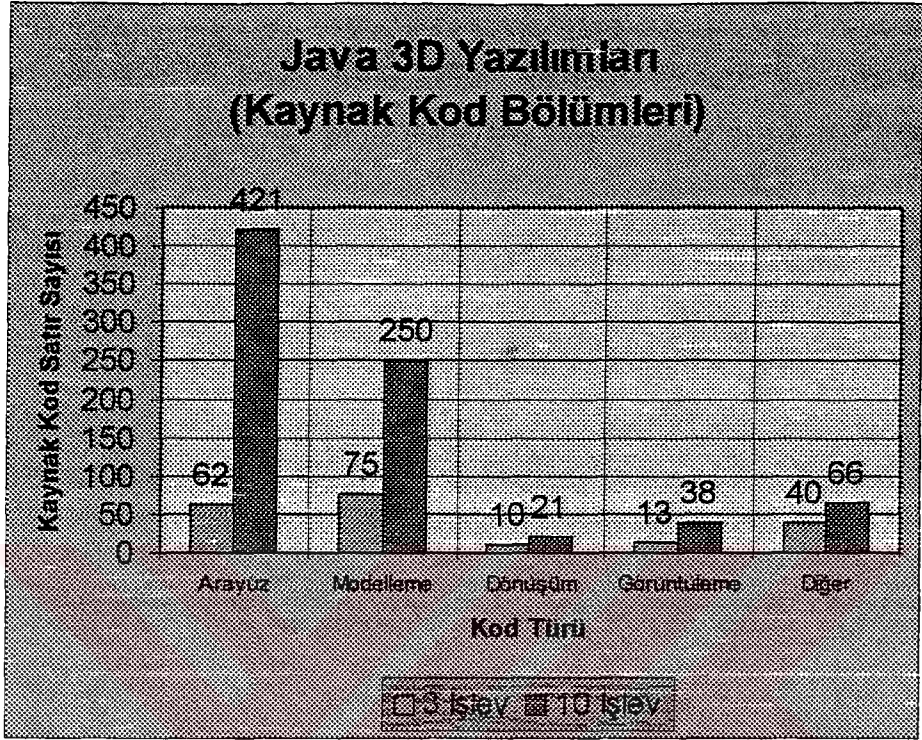
Uygulama geliştirme arayüzlerinin günümüzde geldiği noktada, çok işlevli ve gerçekçi yazılımlara doğru gidildikçe, iki ve üç boyutlu uygulama geliştirme arayüzlerinin üç boyutlu grafik yazılımları geliştirmedeki performans farkları kapanmaktadır. Bu nedenle, arayüzlerin kullanımının kolaylaştırılması ve ileri düzeyde gerçekçiliği sağlayıcı düzenlemeler yapılması gerekmektedir.

9.4.3 Genel Karşılaştırma

İşlev sayısı arttığında, bu yazılımların bölümlerine göre satır sayılarının dikey karşılaştırılmaları, Java 2D için Şekil 9.15'te, Java 3D için Şekil 9.16'da gösterilmektedir.

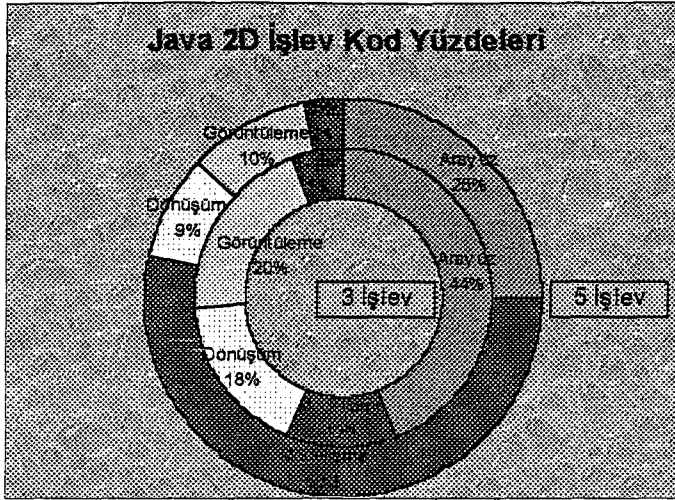


Şekil 9.15 : Java 2D yardımı ile geliştirilen Kup3B (3 işlevli) ve Model3B (5 işlevli) yazılımlarının bölümlerine göre kod satır sayıları.

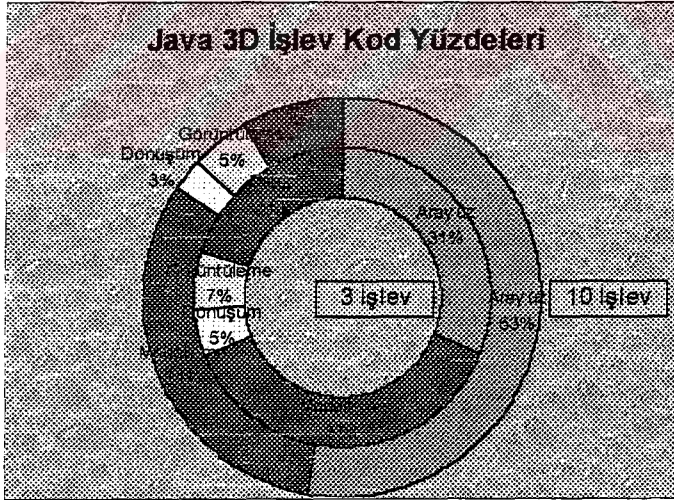


Şekil 9.16 : Java 3D yardımı ile geliştirilen Kup3B (3 işlevli) ve Model3B (10 işlevli) yazılımlarının bölümlerine göre kod satır sayıları.

İşlev sayısı arttığında, bu yazılımların bölümlerindeki satır sayısı yüzdelерinin dikey karşılaştırılmaları ise, Java 2D için Şekil 9.17’de, Java 3D için Şekil 9.18’de gösterilmektedir.

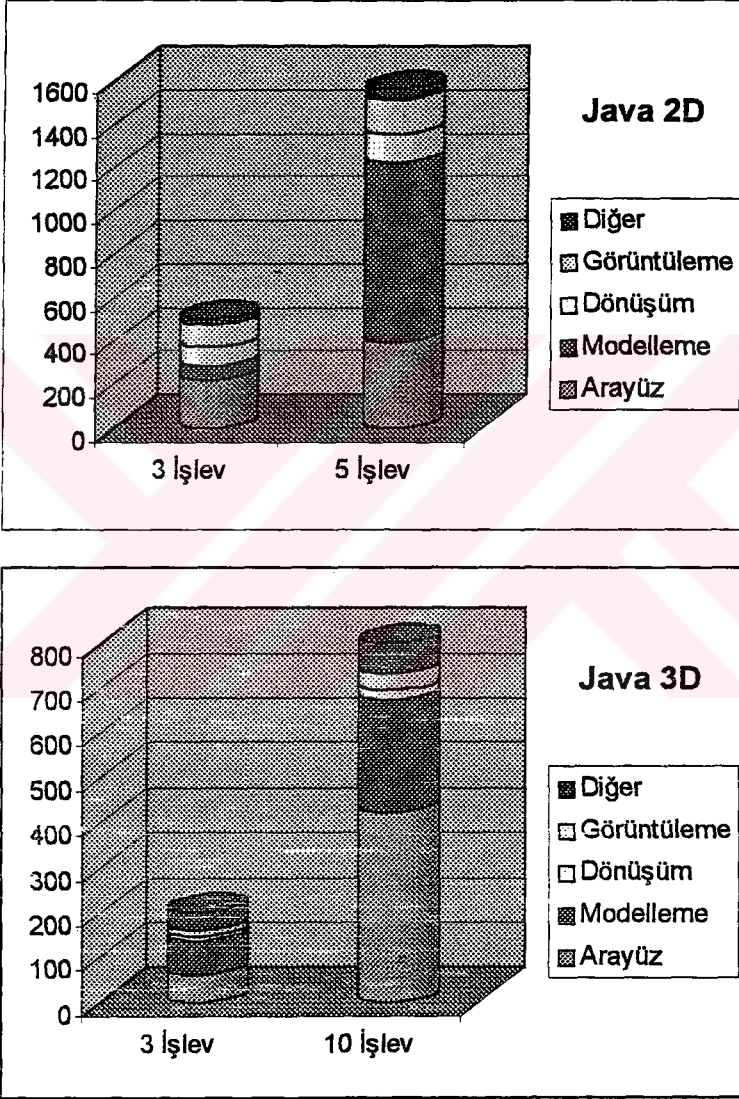


Şekil 9.17 : Java 2D yardımı ile geliştirilen Kup3B (3 işlevli) ve Model3B (5 işlevli) yazılımları arasında bölümlerine göre kod satır sayısı yüzdelere.



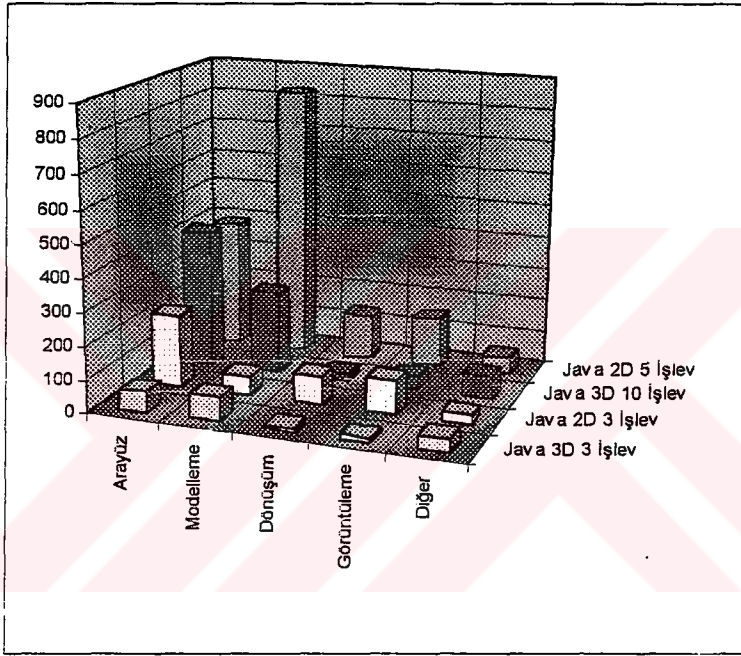
Şekil 9.18 : Java 3D yardımı ile geliştirilen Kup3B (3 işlevli) ve Model3B (10 işlevli) yazılımları arasında bölümlerine göre kod satır sayısı yüzdelere.

İşlev sayısı arttığında, bu yazılımların bölümlerindeki satır sayılarının karşılaştırılmaları, Şekil 9.19'da (Java 2D ve Java 3D için) yığılmış sütun grafiği üzerinde gösterilmektedir.



Şekil 9.19 : Yığılmış sütun grafiği. Yukarıda Java 2D ve altta Java 3D'ye ilişkin grafik yer almaktadır.

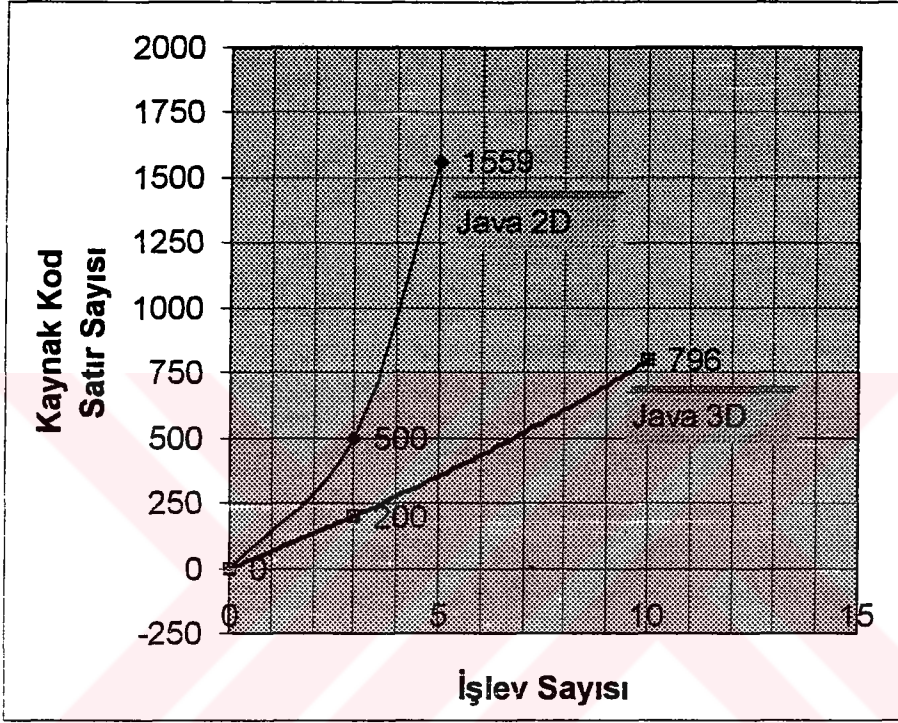
Şekil 9.20’de tüm yazılımların aynı grafik üzerinde karşılaştırılması görülmektedir. Java 2D’de işlev sayısı ve bir anlamda istenen gerçekçilik arttıkça ve karmaşık nesneleri modellemek gerektiğinde, modelleme için yazılması gereken kod miktarının çok hızlı arttığı daha açık olarak görülmektedir.



Şekil 9.20 : Tüm yazılımların tüm bölümlerinin üç boyutlu sütun grafiğinde karşılaştırılması.

İşlev sayısının artışına göre geliştirilen üç boyutlu grafik yazılımının kaynak kod satır sayısının artışı Şekil 9.21’de gösterilmektedir. Java 3D’nin sunduğu gerçekçilik oranına yaklaştıkça, Java 2D ile geliştirilmesi gereken kod miktarı daha fazla bir artış

gösterecektir. Belirli bir düzeyden sonra eklenecek olan her bir işlem yazılımın geliştirilmesini daha da zorlaştırmaktadır.



Şekil 9.21 : İşlev sayısının artışına göre kaynak kod satır sayısı değişimi

Bunların dışında üç boyutlu bir uygulama geliştirme arayüzü yardımı ile geliştirilen programın etkileşim özellikleri daha standart olup, daha niteliklidir. Büyük kuruluşların ortaklaşa geliştirdikleri bir üç boyutlu uygulama geliştirme arayüzünün, belirli bir uygulama için bir şirket veya kişiler tarafından geliştirilecek olan bir arayüzden daha iyi olacağı kesindir.

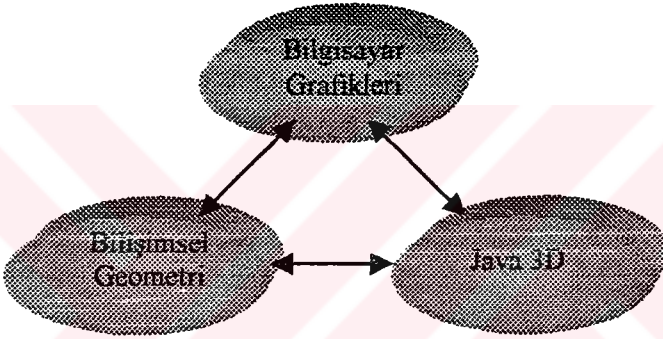
9.4.4 Yazılım Ölçümü

Yazılım ölçümü, yazılımın kalitesini anlamak ve yazılım geliştiricilerin verimliliğini belirlemek gibi amaçlarla yapılır. Yazılım ölçütleri fiziksel ölçütler gibi iki çeşittir : Doğrudan ölçüler ve dolaylı ölçüler. Boyut uyarlı ölçütler, yazılım mühendisliği çıktısının ve kalitesinin doğrudan ölçülmesinde kullanılır. Fonksiyon uyarlı ölçütler, dolaylı ölçütlerdir. İnsan uyarlı ölçütler, yazılım geliştiren kişilerle ilgilidir. Yazılım mühendisliği işlemindeki doğrudan ölçüler, maliyet ve harcanan çabadır. Ürünün doğrudan ölçüleri arasında, üretilen kaynak kodun satır sayısı, işletim hızı, belirli bir süre içinde tespit edilen hata sayısı sayılabilir. Ürünün dolaylı ölçüleri, fonksiyonelliği, kaliteyi, karmaşıklığı, etkinliği, güvenilirliği ve diğer özellikleri içerir. Verimlilik ölçütleri, yazılım mühendisliği işleminin çıktısına odaklanır, kalite ölçütleri, müşteri gereksinimlerinin (yazılımın kullanılabilirliği) uygunluğunun göstergesidir. Teknik ölçütler ise yazılımın mantıksal karmaşıklığı gibi yazılımın karakterine odaklanır.

10. SONUÇLAR

Bilgisayar grafikleri alanındaki bu tez çalışmasında aşağıda belirtilen üç konu ve birbirleri ile etkileşimleri üzerinde ağırlıklı olarak araştırma yapılmıştır (Şekil 10.1) :

- Bilgisayar Grafikleri
- Bilişimsel Geometri
- Grafik Uygulama Geliştirme Arayüzleri (özellikle Java 3D)



Şekil 10.1 : Tez çalışması kapsamındaki konular

Bilgisayar grafiklerindeki geometrik problemlerin çözümünde kullanılan yöntemler üzerinde durulmuş, grafik yazılımları geliştirmeyi sağlayan popüler araçlar kullanılarak gerçekleştirilen ve geometrik hesaplamalar içeren dört grafik yazılımı (Kup3B Java 2D sürümü, Kup3B Java 3D sürümü, Model3B Java 2D sürümü ve Model3B Java 3D sürümü) üzerinde çalışmalar yapılarak sonuçlara ulaşılmıştır. Önceki bölümlerde yer alan sonuçlara da bu bölümde referans verilmiştir. Tüm yazılımlara "bornova.ege.edu.tr/~ugur" adresinden ulaşılabilmekte ve yazılımlar etkileşimli olarak internet üzerinden çalıştırılabilmektedir

10.1 Bilgisayar Grafikleri, Bilişimsel Geometri ve Aralarındaki İlişki

Tez çalışması kapsamında yapılan araştırmalardan elde edilen sonuçlara bağlı olarak, üç boyutlu bilgisayar grafiklerinin, bilgisayar yazılımlarında ve gerçek yaşamda kendisini daha fazla hissettireceği belirlenmiştir. Aynen iki boyutlu grafiklerin günümüzdeki durumu gibi, üç boyutlu grafikler de, web sayfalarıyla, sunum, kelime işlemci gibi değişik türlerde yazılımların doğal bir parçası haline gelecektir. Örnek olarak, üç boyutlu geometrik modeller ve sahneler içeren etkileşimli sunular hazırlanması, sunu yazılımlarının sıradan işlevleri arasında yer alacaktır. (Bkz. Bölüm 3.3)

Tez çalışmasında yapılan araştırmalara göre, bilişimsel geometri alanı önemini giderek artırmaktadır ve genişlemektedir. Bilişimsel geometrinin gelecekte göstermesi beklenen gelişim Bölüm 2.4'te anlatılmıştır.

Bilgisayar grafiklerinin, bilişimsel geometri tekniklerinin en önemli uygulama alanı olduğu söylenmektedir. Bilgisayar grafikleri ile bilişimsel geometri arasındaki karşılıklı yarar esasına dayanan ilişki Bölüm 2.3.1'de anlatılmıştır.

10.1.1 Geometrik problemlerin çözümünde kullanılan algoritmik yöntemler

Geometrik problemlerin çözümünde kullanılan algoritmik yöntemler Bölüm 4'te örnekler verilerek anlatılmıştır. Bir geometrik problemin çözümünde, problemin şartlarına uyan değişik birçok algoritma geliştirilebilir. Bu algoritmaların etkinlikleri, zaman ve yer karmaşıklıkları da farklılıklar gösterebilmektedir. Tez projesinde, seçilen bir problem üzerinde (dışbükey kabukların bulunması), bu yöntemlere

uyarak çözümü gerçekleştiren farklı algoritmalar, zaman karmaşıklıkları yönünden karşılaştırılmışlardır.

Bu yöntem desenlerini bilmek, algoritmaların anlaşılmasında, açıklanmasında ve yeni algoritmaların tasarımında yararlı olacaktır. Daha önce benzeri ile karşılaşılan bir algoritmanın geliştirilmesi daha kolaydır. Her bir yöntem belirli durumlarda kullanılmaktadır : Birer birer ekleme yöntemleri, genelde girdinin zamanın akışına göre gelebildiği durumlarda kullanılan yararlı bir yöntemdir. Tüm girdinin baştan belirli olmasının gerekmemesinden dolayı çevrimiçidir. Birer birer seçme yöntemleri ise, tüm girdilerin baştan belirli ve hazır olduğu durumlarda etkin sonuçlar vermektedir. Bu nedenle çevrimdışı olarak çalışırlar. Düzlem tarama yöntemleri, düzlemde geometrik nesneler içeren problemlerin çözümünde etkindirler. Böl ve çöz yaklaşımı ise, küçük problemlerin çözümünün kolay olması gerçeğine dayanarak problemleri daha küçük parçalarına bölerek her bir parçayı çözer ve çözümleri, orijinal probleme çözüm olacak biçimde birleştirir. Uzaysal bölümlleme algoritmaları da uzaysal bir bölgeyi küçük parçalarına ayrıştırarak, problemleri daha küçük ve daha basit alt problemlerine indirgeyerek çözülmesini sağlarlar.

Etkin algoritmaların geliştirilmesinde kullanılan veri yapılarının da önemi büyüktür. Üç boyutlu nesneler bilgisayarda, köşe, kenar ve yüzey bilgileri ile tanımlanırlar. Bu tür listelerin tutulmasında veri tiplerinin ve bileşik veri tiplerinin yanında, bağlı liste, yığıt, kuyruk ve ağaç gibi veri yapıları da kullanılmaktadır. Bu tür genel veri yapılarının dışında, iki ve üç boyutlu geometrik nesneler üzerinde işlemler yapmak için nokta, köşe, çokgen gibi sınıflar da oluşturulmalıdır.

10.1.2 Bilgisayar grafiklerindeki geometrik problemler

Bilgisayar grafiklerindeki geometrik problemler Bölüm 6'da anlatılmıştır. Üç boyutlu nesnelerin ekranda görüntülenmesi için izlenen

işlem sırası içinde, geometrik hesaplamalar ve çözülmüş olan birçok geometrik problem yer almaktadır. Bunlardan en önemlileri şu şekilde özetlenebilir:

Dönüşüm işlemleri : Nesneler ve kamera için ölçeklendirme, döndürme ve taşıma gibi birçok dönüşüm işlemi, matrisler yardımıyla kolaylıkla gerçekleştirilmektedir.

İzdüşüm işlemleri : Üç boyutlu nesnelerin, iki boyutlu görüntü yüzeylerinde görüntülenmek üzere izdüşümlerinin alınması ile z koordinatlarının ortadan kaldırılması için geometrik birtakım işlemler gerçekleştirilmektedir. İzdüşüm işlemleri, basit üçgenlerden yararlanarak oluşturulan formüller veya matris işlemleri yardımı ile gerçekleştirilmektedir.

Görünmeyen yüzeylerin ortadan kaldırılması işlemleri : Bir sahnedeki görünmeyen yüzeylerin belirlenerek ortadan kaldırılması ile, görünen yüzeyler belirlenir. Bakış noktasından görünmeyen yüzeylerin görüntülenmemesi gerekmektedir. Nesnelerin arka yüzeyleri ve diğer nesnelerin arkasında kalan nesne parçaları görüntülenmemelidir.

Gerçek koordinatlardan araç koordinatlarına dönüşüm işlemleri: İki boyutlu görüntü yüzeylerine aktarılmış ve görünecek olan nesneler için bir çeşit ölçeklendirme yapılarak, gerçek dünyadaki pencerenin ekran penceresine eşlenmesi sağlanır.

Tez projesinin uygulama kısmının bir parçası olarak, üç boyutlu nesnelerin ekranda görüntülenebilmesi için izlenen işlem sırasının Java programlama dilinde gerçekleştirimi yapılmıştır. Bu gerçekleştirim sırasında, özellikle iki boyutlu şekillerin ve grafiklerin çizdirilmesi ve renkler için, “Java awt” ve Java 2D API’den de destek alınmıştır. Java 3D API kullanılarak geliştirilen yazılımlarda ise, üç boyutlu geometrik nesnelerin sisteme eklenmesi ve görüntülenmesi, uygulama geliştirme

arayüzünün otomatik bir parçası olduğundan, ayrıca bu işlem sırasının uygulanmasına gerek kalmamıştır.

Bilgisayar grafiklerinde üzerinde çalışılan geometrik problemler Bölüm 6.3'te anlatılmıştır.

10.2 Grafik Uygulama Geliştirme Arayüzleri

Bölüm 7.5'te üç boyutlu grafik yazılımları geliştirmede 2B ve 3B API'lerin karşılaştırılması yapılmıştır. Üç boyutlu nesnelerin 2B API'ler yardımıyla ekran üzerinde görüntülenebilmesi için, karmaşık bir işlem sırasının izlenmesi gerekmektedir. Bu sürecin zorluğu, işlemleri zorlaştırıp kaliteyi düşürmekte, yazılım maliyetlerini artırmakta ve yazılımların piyasaya sürümünü geciktirmektedir. Bu tür programları yazmak, iyi bir matematik ve programlama altyapısı da gerektirmektedir. Bu nedenlerle, yazılımlarında veya programlarında üç boyutlu bilgisayar grafiklerini kullanmak isteyen profesyonel ve amatör kişi veya şirketler için bu yolu izlemek, çok uygun olmamaktadır. Üç boyutlu grafik ve animasyonlar içeren yazılımlar geliştirmek için kullanılan diğer bir yol üç boyutlu uygulama geliştirme arayüzlerinden yararlanmaktır. Alt düzeyli 3B API'ler hız sağlarken, yüksek düzeyliler yazılım geliştiricilerin sadece yazılımın içeriğine odaklanmalarını sağlar (Brown ve Petersen, 1998) (Day, 1998) (Sowizral ve ark., 1998) (Sun Microsystems, 1999). Nesnelerin temsili ve görüntülenmesi gibi konularda, analiz, tasarım ve kodlama yapılmasına gerek kalmaz. Yüksek düzeyli 3B API'lerin önemi, yeni geliştirilmekte olan yazılımlarda kullanım oranları artmıştır ve geliştirilecek olan yazılımlarda, daha yoğun ve etkin olarak kullanılacakları tahmin edilmektedir. Tez kapsamında geliştirilen Kup3B ve Model3B yazılımlarının yazılım geliştirme süreçlerinden alınan sonuçlara bakılarak yapılan karşılaştırmalar ve elde edilen grafikler Bölüm 9.4'te sunulmuştur. Şirketler ve yazılım geliştiriciler, üç boyutlu

grafik yazılımları geliştirmede 3B API'lerin yazılım geliştirme sürecini hızlandırıp kolaylaştırdığını ifade etmektedirler. Bu çalışma da, “ne kadar?” sorusunun yanıtını aramak açısından değerlidir. Tez kapsamında yapılan araştırmalarda, benzeri bir çalışmaya rastlanmamıştır.

10.2.1 Yüksek Düzeyli 3B API'lerin Özellikleri ve Bilgisayar Grafikleri Alanına Etkileri

Yüksek düzeyli üç boyutlu grafik uygulama geliştirme arayüzleri ve özellikleri Bölüm 7.4'te anlatılmıştır. Tez kapsamında geliştirilen yazılımlardan elde edilen tecrübelerle göre, 3B API'lerdeki gelişmelerin bilgisayar grafikleri alanına ve grafik içeren yazılımların geliştirilmesine etkileri şu şekilde özetlenebilir:

- 3B API'lerdeki gelişmeler, üç boyutlu grafik ve animasyonlardaki ve bunları içeren yazılımlardaki kalite ve performansı artırmaktadır.
- 3B API'lerdeki gelişmeler, kolaylıkla genişletilebilen yazılımlar üretilmesini sağlamaktadır. Grafik yazılımlarının yeni sürümlerinin geliştirilmesini kolaylaştırmaktadır. 3B API'ler, yazılımlardaki yeniden kullanılabilirliği (software reuse) artırmaktadır.
- Yazılan bir grafik programı, her türlü donanım ve yazılım ortamında çalışabilmeli hedefine yaklaşmıştır.
- 3B API'lerin nesneye yönelik arayüzü ve yüksek düzeyli modeli, yazılım geliştiricilerin sadece yazılımdaki içeriğe ve üç boyutlu nesneler üzerine yoğunlaşabilmelerini sağlar; görüntüleme, görüntüleme işlemi ve alt düzeyli programlama yükünü hafifletir. Sonuçta, üç boyutlu nesnelerin tasarımında ve üç boyutlu görüntüleme kodunun yazılmasında daha az zaman ve çaba harcanır; daha az hata ortaya çıkar. Üç boyutlu bilgisayar grafikleri içeren yazılımlarda, yazılım geliştirme işlemi kolaylaşır ve maliyet düşer.

- 3B API'ler, üç boyutlu yazı tipleri, dokular, biçimler gibi konularda geniş kütüphaneler oluşturulmasını kolaylaştırarak bilgisayar grafiklerindeki çeşitliliği artırmaktadır. Grafiklerde kullanılan vektör matematiği konusunda destek vermektedirler. Çok geniş grafik özellikleri arasından seçim yapmayı sağlamaktadırlar.
- Sayısı giderek artan üç boyutlu yükleyicilerden alınan grafiklerin, programlama dilleriyle kullanılabilmesini sağlamaktadırlar.

10.2.2 Yüksek Düzeyli 3B API'lerin Beklenen Genel Etkileri

Üç boyutlu API'lerin- gelişim süreci boyunca şu gelişmelerin yaşanması beklenebilir :

- Üç boyutlu bilgisayar grafikleri içeren yazılımları geliştirmek isteyen kişilerin, çok iyi matematik altyapısına sahip olması ve uzun programlar yazması gerekmeyecektir. Deneyimsiz programcılardan profesyonellere kadar her düzeydeki insan, üç boyutlu sanal dünyalar oluşturup, bunlar üzerinde işlemler yapabilecektir. Bunun için sadece kişisel bilgisayara sahip olmak yetecektir. Üç boyutlu bilgisayar grafikleri içeren yazılım kodlarının anlaşılması kolaylaşacaktır. Sonuçta grafik yazılımları geliştiricilerinin sayısı artacaktır.
- Üç boyutlu bilgisayar grafikleri, bilgisayar yazılımlarında ve gerçek hayatta ağırlığını daha fazla hissettirecektir.
- Daha nitelikli ve ileri grafiksel içerikli projeler üretilebilecektir.
- Bilgisayar grafikleri ile ilgili derslerde de üç boyutlu görüntüleme adımlarındaki alt düzey programlama yükü hafifleyeceğinden, daha ileri konularda daha ileri örnekler verilebilecek ve öğrenciler daha ileri projeler yapabileceklerdir. Yüksek düzeyli 3B API modellerinin kullanımı, derslerdeki anlaşılabilirliği ve performansı artıracaktır.

- Grafik yazılımlarındaki kalitenin artışı, yazılım geliştirmenin kolaylaşması ve sürecin hızlanması sonucunda, grafik yazılımlarının maliyetleri düşecektir. Şirketler arasında oluşacak rekabet, niteliğin daha da artmasını sağlayacaktır. Böylece örnek olarak eğitim yazılımlarında görsel öğelere daha çok yer verilecektir. Bu da konuların anlaşılabilirliğini artıracaktır.

Bütün bu gelişmelerin sonucunda 3B API'ler gelecekte, günümüzde olduğundan daha fazla önem taşıyacaktır. Teorik araştırmalar ile diğer uçtaki kaliteli ürün ve yazılım geliştirme arasında bağlantılar kuran bir platform oluşturmak önemli bir konudur. 3B API'lerdeki gelişim, akademi, araştırma ve pratik hayattaki gelişmeleri tetikleyen bir faktör olacaktır. Bu gelişmeler bilgisayar grafiklerinde ve bağlantılı alanlarda da kendini hissettirecektir.

10.2.3 Yüksek Düzeyli 3B API'lerin Kullanımı

3B API'lerin kullanımında dikkat edilmesi gereken konular şu şekilde özetlenebilir :

- 3B API'ler ile grafiksel içerikli bir yazılım projesine başlamadan önce, yazılım geliştiricilerin, arayüzün olanaklarını ve kısıtlamalarını çok iyi bilmeleri gerekmektedir. Hangi işlemler ve özellikler, arayüz tarafından desteklenmektedir ve hangi işlemler programlanmalıdır. Bunun için arayüzlerin tanıtıldığı ayrıntılı başvuru ve kullanıcı belgeleri ile hazır örneklerin, internet üzerinden ve/veya eğitim CD'lerinden elde edilmesi gerekir. Ayrıca üç boyutlu grafik programlamanın zorluğundan kaynaklanan nedenlerle, arayüzü geliştiren çalışma gruplarının ve uzmanların hazırladığı derslere katılmak, kitapları okumak da yararlı olacaktır.

- Grafiksel içerikli bir yazılım projesinin işletileceği ortamların bilinmesi de önem taşımaktadır. Geliştirilecek yazılımın hedeflediği şirket veya kişilerin sahip olduğu veya olabileceği donanım ve yazılım ortamına göre arayüz seçilmelidir. Yazılım projesinden beklenen hız ve kalite de arayüzün seçiminde büyük önem taşımaktadır. Ayrıca, arayüzün sürümlerinin ve sürümleri arasındaki farklılıkların da bilinmesi gereklidir.
- Uzman olmayan programcılardan, yüksek düzeyli 3B API'ler yardımıyla grafiksel içerikli büyük yazılımlar geliştirmeleri beklenmemelidir. Bilim adamları, öğretmenler, şirket yöneticileri gibi uzman olmayan programcılar, birkaç sene öncesine kadar pek olası görünmemesine rağmen, biraz eğitimle API'leri kullanarak basit programlar geliştirebilirler. Daha ileri programlar geliştirebilmek için, çalışma yapılan alanın dışında, ileri bilgisayar grafikleri bilgisi, ileri yazılım bilgisi (yazılım mühendisliği, programlama, algoritma, veri yapıları) ve ileri matematik bilgisi gerekmektedir.

10.2.4 Yüksek Düzeyli 3B API'lerin Eksikleri

Yüksek düzeyli grafik API'leri, yazılım geliştiriciler için birçok işlemi kolaylaştırmıştır. Ancak tez projesi kapsamında yapılan inceleme ve geliştirilen yazılımlardan elde edilen sonuçlara göre, yüksek düzeyli 3B API'ler, gelişim süreçlerinin başındadırlar ve aşağıdaki iyileştirmelerin yapılması gerektiği belirlenmiştir :

- **Gerçekçilik Artırılmalıdır** : 3B yüksek düzeyli API'ler, çok gerçekçi görüntüler üretebilecek yeteneklere sahip olmalıdır. Bu amaçla, ışın izleme ve ışınma algoritmalarının da bu sistemlerin birer parçası olması gerekmektedir. Gerçek dünyada algıladığımız renkler, nesnenin fiziksel özellikleri ile ışık kaynaklarının özellikleri, nesnelerin ışık kaynaklarına, diğer nesnelere, bakış noktasına ve

açısına göre konumlarının bir bileşimidir. Java 3D, gerçek dünyadaki fiziksel kurallara yaklaşım yapan bir aydınlatma modeli kullanır. Karmaşık fiziksel olaylar modellenmemektedir. “Ambient Light” (ortamda her yerde olan ışık), “Directional Light” (güneş gibi çok uzak kaynaktan gelen yeri olmayan tek yönlü ışık), “Point Light” (mum gibi etkisi uzaklaştıkça azalan noktasal ışık) ve “Spot Light” (belirli bir yerde odaklanan ışık) gibi ışık kaynaklarını desteklemektedir. 3B grafik API’lerinin sanal gerçeklik ve e-ticaret gibi bir çok alanda yazılım geliştiren şirket veya kişiler tarafından, gönül rahatlığı ile kullanılabilmesi için, gerçekçilik konusunda geliştirilmeleri gerekmektedir. İş dünyasından gelen yoğun baskılar sonucunda, 3B API’lerdeki gerçekçilik özellikleri artırılacaktır. Örnek olarak otomobil şirketleri, yeni çıkan modellerinin üç boyutlu görüntülerini, tanıtım için internet üzerine yerleştirmektedirler. Kullanıcılar, ürünleri incelerken döndürebilmekte, büyütüp küçültebilmekte ve içine girip deneme sürüşü yapabilmektedirler. Ancak ışık yansımaları ve kırılmaları, gölgeler, camdan ortamın görüntüsünün yansması gibi etkiler de gerçekçiliği artıracaktır. Bu, şirketlerin ürün tanıtımlarını iyi yapmalarını sağlayacaktır. Bu tür karmaşık fiziksel özelliklerin eklenmesi, yüksek düzeyli 3B API’lerin kullanımını daha da artırıp yaygınlaştıracaktır.

- **Olanaklar Genişletilmelidir :** Üç boyutlu uygulama geliştirme arayüzleri ile, her istenen şey yapılamamaktadır. Standart özellikleri dışına çıkmak, ayrı zorluklar getirmektedir. Molekülleri görüntüleyen bilimsel bir yazılımla bir oyun programının, internette ürün tanıtımında kullanılan bir grafik programı ile, genel amaçlı bir üç boyutlu çizim veya boyama programının gereksinimleri birbirinden farklıdır. Bu şekilde düşünüldüğünde, farklı türlerde yazılım geliştirmek isteyenlerin gereksinimlerini karşılayabilecek bir API’nin,

çok geniş özelliklere sahip olması gerektiği sonucuna varılacaktır. API özelliklerinin, eldeki yapıyı ve kullanımı karmaşıklaştırmadan genişletilmesi gerekmektedir. Çünkü özellikler ve parametreler çoğaldıkça, uygulama geliştirme arayüzünü belli bir alana yönelik olarak öğrenmek de zorlaşmaktadır.

- **Hız Artırılmalıdır :** Üç boyutlu yüksek düzeyli uygulama geliştirme arayüzleri, özellikle animasyon üretiminde daha hızlı sonuçlar üretmek durumundadırlar yani hızlandırılmalıdırlar. Bunun için, donanım ve yazılım konularında yapılan çalışmalar hızlandırılmalıdır.
- **Zorluklar Azaltılmalıdır :** Arayüzlerin büyük ölçekli yazılım geliştirmedeki zorlukları da azaltılmalıdır. 3B API'ler ile programlama, bir programlama dilinin yapısını bilmekten daha fazlasını gerektirmektedir. Animasyon yapmak için, fare ve klavye etkileşimini doğru olarak vermek için, görüntü ağacı gibi sistem yapısını ve konuyu tümüyle kavramış olmak gerekmektedir. Tez çalışmaları sonucunda, birçok kişinin Java 3D kullandığı, ancak kullanımını zor bulduğu belirlenmiştir. İnternet üzerinden elde edilebilecek yazılım örneklerinin çoğaltılması, yeni kitap, belge ve yayınların yazılması, arayüzlerin yapılarında da bazı kolaylaştırmalar yapılması, hata ayıklama araçları geliştirilmesi, zorlukları azaltmada bir derece etkili olacaktır.
- **Hata, Belirsizlik ve Uyuşmazlıklar Giderilmelidir :** 3B API'ler, gelişme sürecinde olup yeterince oturmadıklarından doğru yazılan programlarda belirsizlikler, hatalar veya beklenmeyen durumlar ortaya çıkabilmektedir. Belirsizliklerin ve arayüz sürümleri arasındaki uyuşmazlıkların giderilmesi gerekmektedir. Web tarayıcıları ile tümüyle bütünleştirilmeleri de yararlı olacaktır.

11. İLERİ ÇALIŞMALAR VE ÖNERİLER

11.1 Doktora Sonrası Yapılacak Çalışmalar

Bilgisayar grafiklerinin günümüzde geldiği noktada, doktora sonrası yapılması planlanan tez kapsamı dışındaki çalışmalar, teorik ve pratik alanda olmak üzere iki kısımda incelenebilir :

Teorik alanda yapılacaklar,

- Bilgisayar grafiklerindeki üzerinde çalışılan geometrik problemlerden (önceki bölümlerde açıklanan) birkaçı üzerinde çalışılarak etkin çözümler geliştirilmesi,
- Değişik bilişimsel geometri problemlerinin çözümünde kullanılabilecek genel amaçlı bir java paketi geliştirilmesi.

Pratik alanda yapılacaklar,

- Yazılımlarda nesnelere animasyon ve biçim değiştirme özellikleri verilmesi ve iki ve üç boyutlu uygulama geliştirme arayüzlerinin bu yönleri ile de karşılaştırılması.
- Yazılımlara, üç boyutlu dosya biçimlerini tanıma dışında kendi modelleme parçalarının da eklenmesi : Şu an sistemde kullanılmakta olan temel şekiller ve üç boyutlu yazılar dışında, üç boyutlu nesnelerin etkileşimli olarak oluşturulabilmesini sağlayan kolay kullanılır bir modelleme arayüzü daha etkili olacaktır.
- Yazılımlara, mühendislik analizi bölümlerinin kısmen eklenmesi.
- Yazılımların, e-ticaret gibi özel alanlarda kullanılabilecek şekilde ticari paket durumuna getirilmesi.
- Java 2D API kullanılarak geliştirilen yazılımda, ışın izleme, ışıma gibi gerçekçi yöntemlerin tam gerçekleştiriminin yapılması (dünyada

büyük grafik araştırma grupları tarafından yürütülmekte olan ışın izleme ve ışınma projeleri gibi : POV-RAY, HELIOS).

- Java 3D API kullanılarak geliştirilen yazılımın, daha çok üç boyutlu grafik dosya biçimini tanıtır hale getirilmesi. Grafik arayüzlerindeki gelişmelere bağlı olarak, dışarıdan alınan nesnelerdeki geçirgenlik, doku, renk, mercek etkisi, kırılma, gölgelendirme gibi özelliklerinin de alınabilmesi.

11.2 Yakın Konularda Çalışma Yapacaklara Öneriler

Tez çalışması sürecinde yapılan araştırmalar sırasında elde edilen deneyimlere göre, yakın konularda çalışmalar yapmak isteyen araştırmacılar için belirlenen bazı güncel konular aşağıda belirtilmiştir :

- Işın izleme ve ışınma yöntemlerinin paralel bilgisayar sistemlerinde etkin olarak gerçekleştirimi.
- İnternet üzerine gerçekçi üç boyutlu içerik eklemeyi kolaylaştıran yazılım geliştirilmesi.
- Yeni arayüzler yardımı ile bilgisayar destekli tasarım yazılımı geliştirilmesi ve gerekli performans ölçümü ve karşılaştırmaların yapılması.
- Java 2D tabanlı gerçekçi bir üç boyutlu arayüz tasarımı.
- Yeni yaygınlaşmakta olan diğer yüksek düzeyli grafik uygulama geliştirme arayüzleri ve OpenGL gibi alt düzeyli API'lerin karşılaştırılmalarının da yapılması. Bu amaçla, internet üzerinden işletilebilen gerçekçi yazılımlar geliştirilmesi.

11.3 Araştırmalara Yeni Başlayacaklara Öneriler

- İnternet üzerine üç boyutlu içerik ekleme konusuyla bağlantılı konularda araştırma yapmak isteyenlerin belirli aralıklarla internet üzerinden büyük ölçüde tarama yapması yararlı olmaktadır. Popülerliğini artıran bir konu olmasından dolayı birkaç hafta içerisinde bile birçok yeni yayın, makale, rapor, çalışma yayınlandığı gözlenmekte ve birçok yeni yazılımın ortaya çıktığı görülmektedir. Bilgi tazelemesi bu tür konularda önemlidir. Kaynaklar dizini sonunda bu adreslerden bazıları verilmiştir.
- Periyodik dergiler dışında, internet üzerinden abone olunabilen dergiler de bilgisayar grafikleri alanında popüler konularda yapılmakta olan çalışmaların izlenmesi açısından önemlidir. Ayrıca Her yıl veya iki yılda bir düzenlenen “WSCG”, “VR”, “I3D”, “Solid Modeling”, “SIGGRAPH”, “Eurographics”, “Pacific Graphics”, “Visualization”, “Volume Graphics”, “Graphics Interface”, “Spring Conference on Computer Graphics” gibi konferansların da takip edilmesi yararlı olacaktır.
- Java 3D konusunda çalışmak isteyenlerin, “www.javasoft.com” adresindeki tartışma listesine üye olmaları, birçok problemlerini çözmelerini sağlayacaktır. Ayrıca bu tartışma listesinden, geçmiş aylara ilişkin soru ve yanıtlarına da etkin bir arayüz yardımı ile ulaşılabilir.
- Bilgisayar grafikleri, bilişimsel geometri, Java 3D ve ışın izleme gibi tez kapsamında yer alan konularda çalışmaya başlayanların tez metninde adı geçen kaynakları da en başta okumaları önerilmektedir.

KAYNAKLAR DİZİNİ

- Anand, V.B.**, 1993, Computer Graphics and Geometric Modeling for Engineers, John Wiley and Sons, Inc., USA, 407 p.
- Ashdown, I.**, 1994, Radiosity : A Programmer's Perspective, John Wiley and Sons, Inc., USA, 496 p.
- Bouvier, D.J.**, 1999, Java 3D API Tutorial Updated – New Includes Version 1.2 Features, Sun Microsystems, USA.
- Brown, K. and Petersen, D.**, 1998, Ready-to-Run Java 3D with plug-and-play code, John Wiley and Sons, Inc., 400 p.
- Burger, P. and Gillies, D.**, 1989, Interactive Computer Graphics (Functional, Procedural and Device Level Methods), Addison Wesley, Great Britain, 504 p.
- Chalmers, A. and Reinhard, E.**, 1998, Parallel and Distributed Photo-Realistic Rendering, Course Notes for SIGGRAPH 98, USA.
- Chazelle, B., et all**, 1996, Application Challenges to Computational Geometry, CG Impact Task Force Report, 48 p.
- Chen, J.**, 1996, Computational Geometry : Methods and Applications, Computer Science Department, Texas, A&M University, 221 p.
- Cohen, M.F. and Greenberg, D.P.**, 1985, The hemi-cube: A Radiosity Solution For Complex Environments, *ACM Computer Graphics*, 19(3) 31-40 pp.
- Day, B.**, Aralık 1998, 3D Graphics Programming in Java : Part 1, Java 3D, Javaworld.
- Day, B.**, Ocak 1999, 3D Graphics Programming in Java : Part 2, Advanced Java 3D, Javaworld.

KAYNAKLAR DİZİNİ (devam)

- Deitel, H.M. and Deitel, P.J.**, 1997, Java How to Program, Prentice-Hall, USA, 1050 p.
- Deitel, H.M. and Deitel, P.J.**, 1999, Java How to Program, Third Edition, Prentice-Hall, USA, 1355 p.
- Foley, J.D., Dam, A., Feiner, S.K. ve Hughes, J.F.**, 1996, Computer Graphics Principles and Practice 2nd edition in C, Addison Wesley, 1175 p.
- Fox, D. and Waite, M.**, 1984, Computer Animation Primer, McGraw-Hill, USA, 501 p.
- Giambruno, M.**, 1997, 3D Graphics and Animation, New Riders Publishing, USA, 541 p.
- Goodman, J.E. and O'Rourke, J.**, 1997, Handbook of Discrete and Computational Geometry, CRC Press, 991p.
- Greiner, G., Heidrich, W. and Slusallek, P.**, 1993, Blockwise Refinement – A New Method For Solving The Radiosity Problem, Proc. Fourth Eurographics Workshop on Rendering, Paris, France, 233-245 pp.
- Hearn, D. and Baker, M.P.**, 1997, Computer Graphics C Version, Prentice-Hall, USA, 652 p.
- Jansen, F.W. and Reinhard, E.**, 1998, Data Locality In Parallel Rendering, *Second Eurographics Workshop on Parallel Graphics and Visualization*, Rennes, 1-15 pp.
- Knudsen, J.**, 1999, Java 2D Graphics, O'Reilly, USA, 339 p.
- Langsam, Y., Augenstein, M.J., Tenenbaum, A.**, 1996, Data Structures Using C and C++, 2nd Edition, Prentice Hall, USA, 672 p.

KAYNAKLAR DİZİNİ (devam)

- Laszlo, M.J.**, 1996, Computational Geometry and Computer Graphics In C++, Prentice-Hall, 266 p.
- Main, M. and Savitch W.**, 1995, Data Structures and Other Objects, The Benjamin/Cummings Publishing Company, Inc., 813 p.
- Mäntylä, M.**, 1988, An Introduction to Solid Modeling, Computer Science Press, Inc., USA, 401p.
- Mount, D.**, 1997, Computer Graphics : Lecture Notes, University of Maryland, USA.
- O'Rourke, J.**, 1998, Computational Geometry In C, Second Edition, Cambridge University Press, 376 p.
- Pfleeger, S.L.**, 1998, Software Engineering (Theory and Practice), Prentice-Hall, 576 p.
- Reinhard, E. and Jansen, F.W.**, 1995, Hybrid Scheduling for Efficient Ray Tracing of Complex Images, *International Workshop on High Performance Computing for Computer Graphics and Visualization*, London, UK.
- Reinhard, E. and Jansen, F.W.**, 1995, Scheduling Issues in Parallel Rendering, *Proceedings of The First Annual Conference of the Advanced School for Computing and Imaging*, Netherlands, 268-277 pp.
- Reinhard, E.**, 1996, Hybrid Scheduling for Parallel Ray Tracing, Delft University of Technology, Final Work for TWAIO course.
- Reinhard, E. and Jansen, F.W.**, 1997, Rendering Large Scenes Using Parallel Ray Tracing, *Parallel Computing*, 873-886 pp.

KAYNAKLAR DİZİNİ (devam)

- Reinhard, E., Chalmers, E. and Jansen, F.W.,** 1999, Hybrid Scheduling for Parallel Rendering Using Coherent Ray Tasks, 1999 IEEE Parallel Visualization and Graphics Symposium, ACM SIGGRAPH, 21-28 pp.
- Sahni, S.,** 1998, Data Structures, Algorithms and Applications in C++, McGraw-Hill, 824 p.
- Sommerville, I.,** 2000, Software Engineering, 6th Edition, Addison-Wesley, USA, 693 p.
- Sowizral, H.A. and Nadeau, D.R.,** 1999, Introduction to Programming with Java 3D, San Diego Supercomputer Center, University of California at San Diego, SIGGRAPH99 Course Notes, 603 p.
- Sun Microsystems, Inc.,** 29 Temmuz 1999, Java 3D API Collateral, The Fourth Generation of 3D Graphics APIs Has Arrived – A New Generation of 3D API Emerges, www.javasoft.com/products/java-media/3D/collateral.
- Sun Microsystems, Inc.,** Java 3D 1.2 API Specification, www.javasoft.com/products/java-media/3D/index.html
- Uğur, A.,** 1996, Üç Boyutlu Çizim ve Animasyon, Yüksek Lisans Tezi, İzmir, 106 s.
- Uğur, A., Kaşlı, A. ve Adaher, O.,** 1999, Effects of 3D APIs on Computer Graphics, 1063-1065, *Proceedings of The Fourteenth International Symposium on Computer and Information Sciences*, ISCIS XIV, Kuşadası, 1088 p.
- Watt, A.,** 2000, 3D Computer Graphics, Third Edition, Addison-Wesley, 570 p.

KAYNAKLAR DİZİNİ (devam)

Weiss, M.A., 1996, Algorithms, Data Structures and Problem Solving with C++, Addison-Wesley, 820 p.

Weiss, M.A., 1999, Data Structures and Algorithm Analysis In Java, Addison Wesley, 542 p.

Woo, M., Neider, J., Davis, T. and Shreiner, D., 1999, OpenGL Programming Guide, Third Edition, Addison Wesley, 730 p.

WEB ADRESLERİ

www.3dlinks.com, Ultimate 3D Links

www.brilliantdigital.com, Brilliant Digital Entertainment

www.cycore.com, Cycore

www.eyematic.com, Eyematic

www.javasoft.com, Sun

www.macromedia.com, Macromedia

www.pennnet.com, Computer Graphics World

www.pulse3d.com, Pulse3D

www.richfx.com, RichFX

www.shockwave.com, Shockwave

www.viewpoint.com, ViewPoint

www.cgri.cs.mcgill.ca/~godfried/teaching/cg-web.html,
Computational Geometry

www.scs.carleton.ca/~csgs/resources/cg.html, Computational
Geometry Resources

ÖZGEÇMİŞ

Adı Soyadı : Aybars UĞUR
Doğum Tarihi : 1972
Doğum Yeri : İzmir
Medeni Hali : Bekar
Uyruğu : T.C.
Görevi : Araştırma Görevlisi
 Ege Üniversitesi
 Bilgisayar Mühendisliği Bölümü

Eğitim

Yüksek Lisans : Ege Üniversitesi
 Fen Bilimleri Enstitüsü
 Bilgisayar Mühendisliği Anabilim Dalı, 1996

Lisans : Ege Üniversitesi
 Bilgisayar Mühendisliği Bölümü, 1993

Lise : İzmir İnönü Lisesi, 1989 (Lise Birinciliği)

Araştırma Alanları : Bilgisayar Grafikleri, Yazılım Mühendisliği,
 Bilişimsel Geometri, Bilgisayar Destekli Tasarım

Yabancı Dil : İngilizce