ETKİLEŞİMLİ SÜRÜCÜ EĞİTİMİ İÇİN KURAL TABANLI BİR PLATFORM GELİŞTİRİLMESİ

2012
DOKTORA TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

İsmail KURNAZ

ETKİLEŞİMLİ SÜRÜCÜ EĞİTİMİ İÇİN KURAL TABANLI BİR PLATFORM GELİŞTİRİLMESİ

İsmail KURNAZ

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalında
Doktora Tezi
Olarak Hazırlanmıştır

KARABÜK
Ocak 2012

İsmail KURNAZ tarafından hazırlanan "ETKİLEŞİMLİ SÜRÜCÜ EĞİTİMİ İÇİN KURAL TABANLI BİR PLATFORM GELİŞTİRİLMESİ" başlıklı bu tezin Doktora Tezi olarak uygun olduğunu onaylarım.

Prof. Dr. Abdullah ÇAVUŞOĞLU .....
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Doktora tezi olarak kabul edilmiştir. 12/01/2012

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Prof. Dr. Abdullah ÇAVUŞOĞLU (KBÜ)	.....
Üye : Doç. Dr. H. Haldun GÖKTAŞ (YBÜ)	.....
Üye : Yrd. Doç. Dr. İ. Rakıp KARAS (KBÜ)	.....
Üye : Yrd. Doç. Dr. Baha ŞEN (KBÜ)	.....
Üye : Yrd. Doç. Dr. Hüseyin DEMİREL (KBÜ)	.....

...../...../2012

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu tez ile, Doktora derecesini onamıştır.

Doç. Dr. Nizamettin KAHRAMAN .....
Fen Bilimleri Enstitüsü Müdürü

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

İsmail KURNAZ

ÖZET

Doktora Tezi

ETKİLEŞİMLİ SÜRÜCÜ EĞİTİMİ İÇİN KURAL TABANLI BİR PLATFORM GELİŞTİRİLMESİ

İsmail KURNAZ

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Prof. Dr. Abdullah ÇAVUŞOĞLU
Ocak 2012, 103 sayfa

Bu çalışmada, sürücü adaylarına güvenli sürüş becerilerini kazanmaları için eğitim ve test imkânları sağlayan 3 boyutlu grafik ortamında çalışan bir trafik simülasyon sistemi (TSS) geliştirilmiştir. Hiyerarşik eş zamanlı durum makineleri kullanılarak trafik akışını sağlayan otonom taşıtlar mikro simülasyon modeli ile geliştirilmiştir. Otonom taşıtlar şerit/taşıt takibi, trafik kurallarına uyum, yavaşlama, hızlanma, kavşak geçişleri gibi güvenli sürüş becerilerinin yanında trafikte normal, agresif ve kural dışı davranacak sürücü davranış modellerine sahiptirler. TSS oryantasyon, eğitim ve test olmak üzere üç farklı şekilde çalışmaktadır. Oryantasyon esnasında aday sürücünün, aracına alışması ve trafik akışı olmayan şehir ortamında trafik kurallarını öğrenmesi hedeflenmiştir. Eğitim esnasında sürücü uymadığı kurallarla ilgili olarak sesli ve yazılı mesaj ile uyarılır. Trafikte uyulması gereken kurallar sistemde bir XML dosyasında tutulmaktadır ve

bu veri kullanılarak sürücü davranışı izlenmektedir. Test esnasında ise sürücüye dakika olarak belirlenen zaman süresince, trafik kurallarına uyumu kontrol edilen bir değerlendirme yapılır. Test sürüşü sonunda sürücüye hatalarını gösteren bir rapor sunulmaktadır. Bu raporlar daha sonra yapılabilecek analiz ve değerlendirme işlemleri için sürücü adına göre test sonuçları veritabanına kaydedilebilmektedir. Sürücü adayı, araç sürme işlevini direksiyon ve pedal donanımlarını kullanarak, üç monitörden oluşan bir görüntü platformu kullanarak gerçekleştirmektedir. TSS kullanıcı ara yüzü ile sürücü adayına araç tercih etme, trafikte yer alacak diğer araçların sayısını belirleme, trafik ışıklarının zaman ayarlaması ve hava koşullarını belirleme işlemleri yapılabilmektedir.

Anahtar Sözcükler : Trafik simülatörü, Mikro Simülasyon, Hiyerarşik eş Zamanlı Durum Makineleri, Etmen Modelleme.

Bilim Kodu **:** 902.1.014

ABSTRACT

Ph.D. Thesis

DEVELOPING A RULE-BASED PLATFORM FOR THE INTERACTIVE DRIVER EDUCATION

İsmail KURNAZ

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

Thesis Advisor:
Prof. Dr. Abdullah ÇAVUŞOĞLU
January 2012, 103 pages

In this study, a Traffic Simulation System (TSS) to provide driver candidates opportunities for training and testing using 3D graphical environment to acquire safe driving skills has been developed. Autonomous vehicles that allow the flow in a micro-simulation traffic model were developed using hierarchical concurrent state machines. Simulated vehicles have autonomous driver characteristics such as lane/vehicle tracking, compliance with traffic rules, deceleration, acceleration, intersection crossings along with driver behaviours disregarding the traffic rules such as normal, aggressive etc. TSS works in three different modes; orientation, education and testing. During the orientation phase, the candidate driver is directed to become familiar with the vehicle and learn traffic rules in city conditions. During the training, the driver is warned both orally and in written format in the case of problematic driving. The rules base is kept in XML file in the system and the data is used to follow the driver's behaviours. During the testing phase, an evaluation mechanism to

check the adaptation of the driver to the traffic rules for the determined time interval is carried out. At the end of the test drive a report showing the drivers mistakes is produced. These reports and test results can be recorded to the database with driver names for further analysis and evaluation processes. Driver candidates perform the driving by means of a steering wheel and pedals with a screen platform consisted of three monitors. The TSS user interface provides the user to select vehicles, determining the types of other vehicles in the traffic, timing adjustments for traffic lights and determining whether conditions.

Key Words : Traffic Simulator, Micro Simulation, Hierarchical Concurrent State Machines, Agent Modelling.

Science Code : 902.1.014

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Prof. Dr. Abdullah ÇAVUŞOĞLU başta olmak üzere hocalarım Doç. Dr. H. Haldun GÖKTAŞ ve Yrd. Doç. Dr. İsmail Rakıp KARAS hocalarıma teşekkürlerimi sunarım.

Sevgili aileme manevi hiçbir yardımı esirgemeden yanımda oldukları için tüm kalbimle teşekkür ederim.

Bu doktora tez çalışması Karabük Üniversitesi tarafından Bilimsel Araştırma Projeleri (BAP) kapsamında, 10D4570501 numaralı proje olarak desteklenmiştir.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

Bm	: Bakış mesafesi
DD	: Yavaşlama mesafesi
FD	: Takip mesafesi
k	: Direksiyon döndürme katsayısı
m	: Metre
km	: Kilometre
km/sa	: Bir saatte gidilen km
s	: Saniye
Sm	: Sapma mesafesi
SD	: Güvenlik mesafesi
Ş	: Şerit durumu
VL	: Araç uzunluğu
Y	: Hareket yönü
°	: Derece
θ	: Sapma açısı

KISALTMALAR

2B : İki Boyutlu
3B : Üç Boyutlu
API : Application Program Interface
AR-GE : Araştırma Geliştirme
BRep : Boundary Representation
CBS : Coğrafi Bilgi Sistemleri
CFM : Constraint Force Mixing
CSG : Constructive Solid Geometry
CR : Crystal Reports
DOS : Disk Operating System
ERP : Error Reduction Parameter
GML : Geography Markup Language
HEZDM : Hiyerarşik Eş Zamanlı Durum Makineleri
JR : Joystick Wrappers
LGPL : Lesser General Public License
LoD : Level of Detail
MEB : Milli Eğitim Bakanlığı
MS VS : Microsoft Visual Studio
MOVES : Institute at the Naval Postgraduate School in Monterey
OpenAL : Open Audio Language
ODE : Open Dynamics Engine
ODBC : Open Database Connectivity
OGC : Open Geospatial Consortium
OSG : Open Scene Graph
PLIB : Portable Game Library
PSTN : Public Switched Telephone Network
SDM : Sonlu durum makineleri
SQL : Structured Query Language
XML : Extended Markup Language
YKG : Yapısal Katı Geometrisi
USB : Universal Serial Bus

BÖLÜM 1

GİRİŞ

Günümüzde, gelişmiş ve kalabalık nüfusa sahip şehirlerin önemli problemlerinden birisi de trafik sorunudur. Trafik sistemini oluşturan öğeler düşünüldüğünde sürücüler bu sistemin en önemli öğelerinden biri olduğu söylenebilir. Bu nedenle sürücü kalitesini arttırmaya yönelik gerçekleştirilecek eğitim faaliyetleri, trafik sorununun çözülmesi ve trafik güvenliğinin iyileştirilmesine yönelik katkılar sağlayabilir.

Trafikte yer alan bütün sürücüler için güvenli araç sürme becerisi bir süreç sonunda kazanılan bir davranıştır. Güvenli araç sürme becerisinin temelinde trafik kurallarına uyma alışkanlığının kazanılması yer alır. Gerçek bir ortamda yapılacak sürücü eğitim faaliyetlerinin pahalı ve riskli olduğu düşünülürse, gerçekçi bir simülasyon ortamında sürücü eğitiminin verilmesi daha iyi sonuçlar verebilir.

İçerisinde araçların, yayaların, trafik işaretlerinin bulunduğu trafik simülasyon çalışmaları 1950'li yıllardan bu yana AR-GE çalışmaları için uygulama alanı olmuştur [1-2] ve son yıllarda eğitim amaçlı bir çok araç sürüş simülatörü geliştirilmiştir [3].

Trafik kazaları, can ve maddi kayıpların çok fazla olduğu kazaların başında gelirler. Yapılan araştırmalar incelendiğinde, trafik kazalarının birinci dereceden sorumlusu olarak sürücü davranışları ve hatalarının sebep olduğu işaret edilmektedir [4,5,6]. Bununla birlikte dünyadaki birçok ülkede yapılan araştırmalar değerlendirildiğinde, ölümlü veya yaralanmalı kazalara karışma oranı, genç sürücülerin diğer sürücülerden daha fazla olduğu tespit edilmiştir [7]. Benzer şekilde, sürücü belgesi aldıktan sonra trafiğe çıkan sürücülerin kaza yapma oranı deneyimli sürücülere nazaran daha

yüksektir [8]. Bunun yanında trafik kazalarının yaklaşık % 80'e varan büyük bir kısmı şehir içi trafiğinde meydana gelmektedir [9].

Bu nedenle acemi olarak isimlendirilecek yeni sürücü belgesi almış veya genç sürücülerin şehir içi trafiğinde güvenli sürüş becerileri ile donatılması trafik sorunlarının azaltılmasına yönelik çözümler sağlayabilir. Bu maksatla geliştirilecek simülasyon sistemleri, sürücü adaylarına gerçek bir trafik ortamında sahip olmaları gereken becerilerin simülatör ortamında canlandırılması ile sürücülerin bu durumları kontrollü ortamlarda yaşayıp, tecrübe etmeleri ve kuralları öğrenip bu kurallara uyma alışkanlıkları kazanmalarını sağlayabilirler [10].

Bu çalışmanın amacı, sürücü adaylarına gerçekçi bir trafik ortamında trafik eğitimi vererek güvenli sürüş becerileri kazanmalarına yardımcı olacak bir trafik eğitim simülatörü yazılımı geliştirmektir.

Tez çalışması altı bölümden oluşmaktadır. Birinci bölüm "Giriş" olup burada çalışmanın gerekliliği ve amacı verilmiştir. İkinci bölümde, simülasyon sistemleri ve gerçekleştirme metotları hakkında yapılan bilimsel çalışmalar ile ilgili bilgi verilmiştir. Üçüncü bölümde trafik simülasyon sistemi geliştirilirken kullanılan kütüphaneler hakkında bilgi verilmiştir.

Dördüncü bölümde geliştirilen yazılımda kullanılan veri kaynaklarının görsel modellerden temin edilirken kullanılan uygunlaştırma yöntemleri anlatılmıştır.

Beşinci bölüm geliştirilen yazılımı içermektedir. Bu bölümde simülasyon sahnesinin oluşturulması, taşıt işlemleri, kullanıcı giriş-çıkış donanımlarının kontrolü, otonom hareket eden vekil taşıt davranışları ve rapor işlemleri için yazılan kaynak kodlarının algoritmaları ve açıklamaları yer almaktadır.

Çalışmanın altıncı bölümünde, yapılan deneysel çalışmalar anlatılarak, elde edilen bulgular değerlendirilmektedir. Yedinci ve son bölümde ise çalışma sonunda elde edilen sonuçlar anlatılmıştır.

BÖLÜM 2

SİMÜLASYON SİSTEMLERİ

Karşılıklı etkileşimli sürücü eğitimi için kural tabanlı bir platform geliştirilmesi çalışması, gerçekte bir simülasyon yazılımı geliştirme sürecidir.

Simülasyon sistemi geliştirirken kullanılacak modeller, içerdikleri ayrıntı seviyesine göre dört farklı sınıfa ayrılırlar [11, 12]:

1. Makro Simülasyon modelleri,
2. Mikro Simülasyon modelleri,
3. Meso Simülasyon modelleri,
4. Nano Simülasyon modelleridir.

Makro simülasyon modelinde, sistemi oluşturan öğeler bir bütün olarak modellenir. Öğelerin kendilerine has özelliklerinin modellenmesi yapılmaz. Bunun yerine genel olarak bütün öğeler için yapılandırılmış özellikler kullanılır. Örneğin bir trafik simülasyonu gerçekleştiriliyorsa, trafik akışını sağlayacak taşıtlar ayrı ayrı modellenmez, farklı hızları yoktur. Bu taşıtlar ortalama bir hızda, çoğu zaman akışkanlar dinamiği kullanılarak yapılandırılmış matematiksel bir modele bağlı kalarak trafik akışı sağlarlar. Makro simülasyon modeli, ideal koşullardaki sistemlerin incelenmesi için faydalı olabilir, fakat çok sayıda değişken parametreye sahip trafik simülatörleri için bu modelleme türü yetersiz kalabilmektedir.

Makro simülasyon modelinden kaynaklanan eksiklikleri gidermek için mikro simülasyon modeli kullanılabilir. Mikro simülasyon modelinde, simülasyon sisteminde yer alan öğelerin bir bütün yerine, tek tek modellenmesi gerçekleştirilmektedir.

Trafik ortamı gibi taşıt, yaya, trafik kuralları, vb bir çok değişkenin ve bu değişkenlere ait bir çok farklı özelliğin bulunduğu bir sistemin benzetimi için mikro simülasyon modeli kullanılabilir. Trafikte her biri farklı davranış karakteristiklerine sahip ve birbirleri ile etkileşim içinde bulunan öğeler bulunmaktadır. Örneğin kimi sürücüler trafik kurallarına uygun hareket ederken, kimileri ise bu kuralların bazılarına uyarlar. Mikro simülasyon modeli ile bu tip farklı davranış gösteren sürücü tipleri modellenebilir.

Mikro simülasyon modellemenin odak noktası, sistemi oluşturan öğeleri belirlemek ve bu öğelerin alt sistemler ile temsil edilmesini gerçekleştirmektir. Öğelerin özellikleri ve birbirleri ile etkileşimlerini kapsayan fonksiyonlar, alt sistemlerde tanımlanır. Bir alt sistemin de farklı alt sistemleri olabilir. Ana sistemin davranışı, sahip olduğu alt sistemlerin davranışlarının ve etkileşimlerinin sonucu ortaya çıkmaktadır [13, 14].

Trafik sistemi mikro simülasyon yöntemi kullanılarak modellenirken sistem içinde yer alan her bir taşıt, yayalar, trafik kuralları, kara yolları, hava koşulları gibi öğeler birer alt sistem olarak düşünülebilir. Bu alt sistemler içinde yer alan trafik kuralları da trafik işaretleri, trafik ışıkları gibi alt sistemlerden oluşabilir. Sistemde yer alan her bir trafik ışığının durumları, yanış periyotları, görünümleri gibi özelliklerin modellenmesi gerekir. Bununla birlikte bir taşıtın trafik ışığı ile etkileşimleri ve bu etkileşim sonucu uygulaması gereken durma, yavaşlama veya geçiş gibi davranışları da modellenmelidir.

Başka bir simülasyon modeli olan meso simülasyon modeli, makro ve mikro simülasyon modellerinin karma kullanımı ile oluşturulmuş bir yöntemdir. Bu modelde, simülasyonda yer alan kimi alt sistemler makro simülasyon yöntemi ile modellenirken, kimi alt sistemler ise mikro simülasyon yöntemi ile modellenir. Meso simülasyon kullanılarak trafik simülasyonu gerçekleştirilmesi, örneğin, trafikte yer alan taşıtlar tip, hız, motor kuvveti, büyüklük gibi nitelikler açısından ayrı ayrı modellenirken, belli bir karayolu kesitinde yer alan tüm taşıtların aynı taşıt olması ve diğer simülasyon öğeleri ile etkileşiminde basit davranışlar sergileyecek şekilde modellenmesi, biçiminde olur.

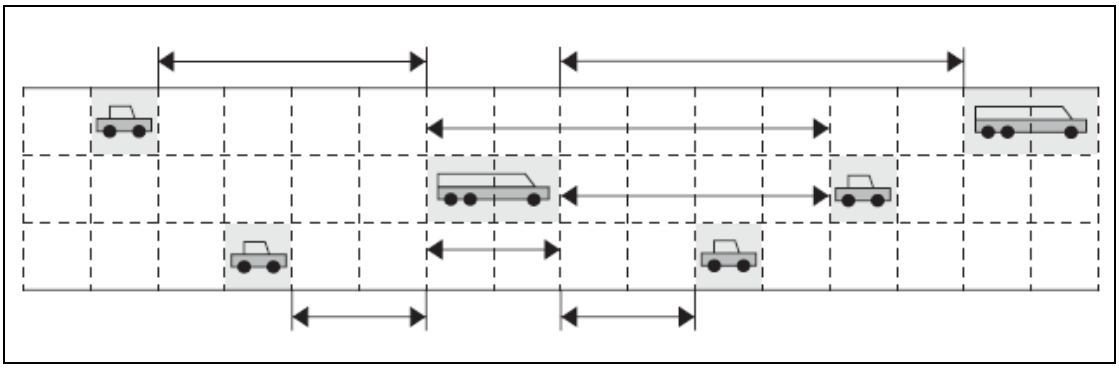
Nano simülasyon modeli, yukarıda anlatılan simülasyon modelleri içerisinde en ayrıntılı olanıdır. Nano simülasyon modeli, temel olarak mikro simülasyon modeline benzer, fakat nano simülasyon modelinde alt sistemler tanımlanırken ayrıntı düzeyi derinleşir. Örneğin, nano simülasyon yöntemi kullanılarak bir taşıt modellenirken mikro simülasyon modelinde tanımlanan taşıtın tipi, hızı, büyüklüğü gibi niteliklerinin yanında kullandığı lastiklerin aşınması, fren mesafesi, sürtünme kuvveti vb. niteliklerinde modellenmesi gerekir.

2.1. TRAFİK AKIŞI MODELLEME TEKNİKLERİ

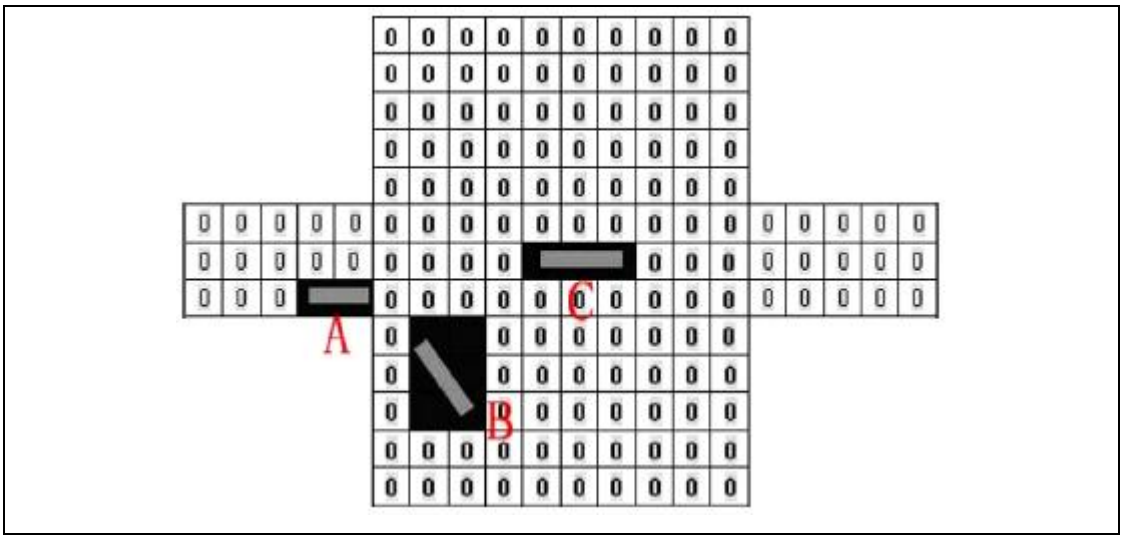
Simülasyon yazılımlarında trafik akışının veya taşıtların ilerlemesinin nasıl olacağı önemli bir problemdir. Bu problemin çözümü için hücresel otomat, uzman sistemler, sonlu durum makineleri, hiyerarşik eş zamanlı durum makineleri, bulanık mantık, etmen tabanlı modelleme yöntemleri kullanılmaktadır.

2.1.1. Hücresel Otomat (Cellular Automata) Model teorisi

Hücresel otomat modelinde öncelikle düzenli hücreler oluşturulur. Bu hücrelerin durumları (1 veya 0) olarak belirlenir. Her bir hücrenin bulunduğu yere göre komşulukları belirlenir. Her bir hücreye bir başlangıç değeri atanır. Belirli zaman aralıklarında hücrelerin başlangıç değerlerinin ne olacağı belirlenir. Hücrenin değerine göre komşu hücrelerin değerleri bazı kurallara göre ayarlanabilir. Hücrelerin boyutları yapılacak işe göre ayarlanır. Trafik simülasyonları için 7,5 m boyutlarında hücreler kullanılır. Zaman değişimi ikincil değişkendir. Örneğin saniyede 4 hücre geçen bir aracın ortalama gerçek hızı (4*7,5*60*60) 108 km/sa olur.



Şekil 2.1. Hücresel otomat modelinde taşıtların yerleştirilmesi [15].



Şekil 2.2. Hücresel otomat modelinde hücrelerin anlamlandırılması [16].

2.1.2. Uzman Sistemler

Uzman sistemler, belirli bir alanda uzmanlaşmış bir kişinin bilgi birikimi ve tecrübelerini kullanarak, sadece o alan ile ilgili bir probleme çözümler getirebilecek bilgisayar programlarıdır [17].

Bir uzman sistemde uzman kişiyi, sistemin veri tabanı temsil eder. Veri tabanında yer alan şart ifadeleri ile durum – işlem çiftleri oluşturulur. Sistem ilgili durumla karşılaştığında yapılması gereken işlem veri tabanından bulunarak yürütülür. Bu yüzden uzman sistemlerde veri tabanında yer alacak durum – işlem çiftlerinin bütün alternatifleri ile birlikte yer alması gerekir.

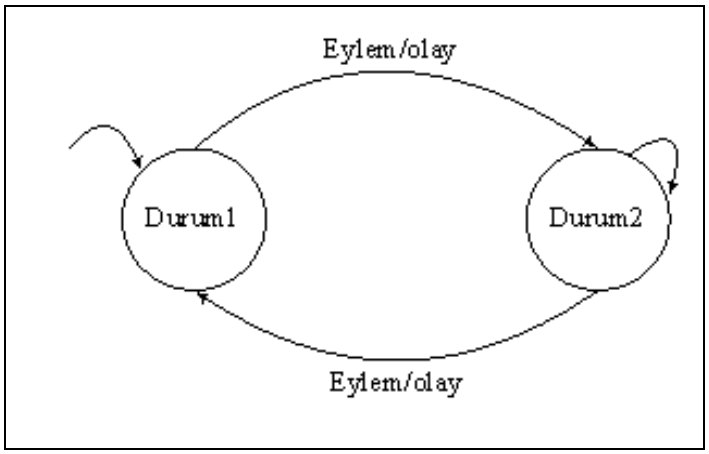
Günümüzde uzman sistem yöntemi kullanılarak geliştirilen birçok başarılı uygulama olmasına rağmen özellikle karmaşık modelleme gereksinimi duyan sistemlerde uzman sistem kullanmanın bazı dezavantajları vardır. Bu dezavantajlar uzman alan bilgisinin, durum – işlem kural yapısına dönüştürme zorluğu ve çok büyük boyutlu kural kümesi ihtiyacı olarak sıralanabilir.

Trafik sistemleri düşünüldüğünde uzman sistemler, trafiğin izlenmesi ve kontrolü ile karayollarının analizi, planlanması ve yönetiminde kullanılmaktadır [18]. Trafik simülasyon sistemi kullanılarak elde edilen verilerin incelenmesinde ve sonuçların analizinde uzman bir kişiye ihtiyaç duyulacaktır. Bu türden analizlerin otomatik olarak gerçekleştirilmesi için uzman sistemlerden faydalanılabilir.

2.1.3. Sonlu Durum Makineleri (Finite State Machines)

Sonlu durum makineleri (SDM), davranış modellemede kullanılan bir tekniktir. Sınırlı sayıda durum ve durumlar arası geçiş gerektiren eylem ve olaylar bir SDM'yi oluşturur. Bir SDM sınırlı sayıdaki durumlar arasında, gerçekleşen eylemlere göre durum değişikliği yapan mantıksal bir mekanizmadır. Bir durumdan diğerine geçiş olabilmesi için önceden belirlenen olayın veya koşulun gerçekleşmesi gerekir [19].

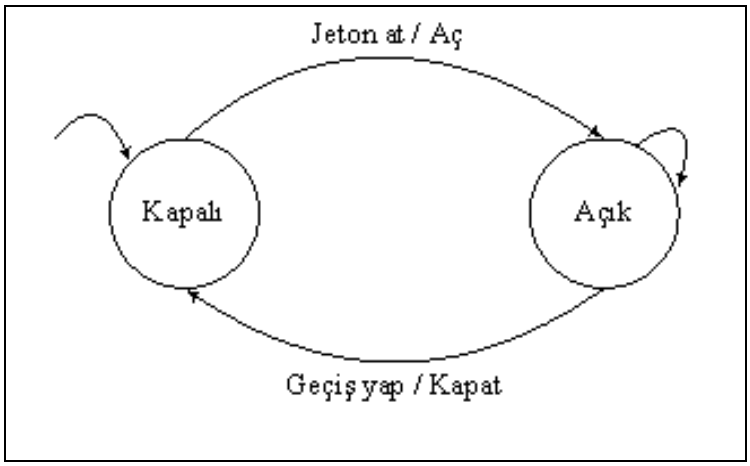
SDM'nin temeli durum geçiş diyagramına (state transition diagram) dayanır [20, 21]. Şekil 2.3'te basit bir SDM'nin durum geçiş diyagramı gösterilmiştir. Burada bulunan SDM'nin başlangıç durumu Durum1'dir ve bir eylem/olay oluşana kadar bu durumunu muhafaza eder. Eylem/olay oluştuğunda, SDM Durum2'ye geçer ve yeni bir olay oluşana kadar bu durumda kalır. SDM'nin durum değiştirmesi için bulunduğu duruma ait geçiş şartının gerçekleşmesi gerekir.



Şekil 2.3. Tipik bir SDM durum geçiş diyagramı.

Durum, her hangi bir anda sistemi tanımlayan değişkenler kümesi olarak tanımlanabilir. Boşluktan başlayan çizgi başlangıç durumunu; durumu veya durumları birbirine bağlayan çizgi geçişi gösterir. Bir durum mevcut durumunu koruma ve durum değiştirme olmak üzere iki tip eylem gerçekleştirebilir.

Günlük hayatta SDM'nin kullanıldığı en basit örnek bilet, jeton veya benzeri bir kart sistemi ile çalışan turnikelerdir. Bir turnike başlangıçta kapalıdır. Turnikeye jeton atıldığında geçiş şartı gerçekleşir ve turnike açılır. Örneğin bu durum bir otoyola giriş kontrolünde ise taşıt turnikeden geçer. Taşıt turnikeden geçtiğinde geçiş şartı sağlanır ve turnike tekrar kapanır. Turnikeye jeton atılana kadar turnike kapalıdır. Bu durum Şekil 2.4'de yer alan durum geçiş diyagramında gösterilmiştir.



Şekil 2.4. Turnike SDM'nin durum geçiş diyagramı.

SDM daha çok ardışık (yani herhangi bir anda sadece tek bir durum makinesi ile tanımlanabilen) davranışlar sergileyen sistemlerin modellenmesinde kullanılmaktadır. Trafik simülasyonu modellerinde genellikle trafik ışıkları SDM ile modellenmektedir. Ancak, karmaşık ve eş zamanlı etkileşimlerin yer aldığı sistemlerin SDM kullanılarak modellenmesi bazı kısıtlamalara neden olur. Bu kısıtlamalar SDM'nin mantıksal mekanizmasının temelini teşkil eden durum geçiş diyagramının yapısından kaynaklanmaktadır. Bu kısıtlamalar;

1. Modeli oluşturan durumlar arasında herhangi bir gruplandırma ya da hiyerarşik yapılanma gerçekleştirilememektedir. Bu sebeple, davranış modelinin belirli kısımlarının diğer kısımlardan izole edilmesi ve parça parça geliştirilmesi mümkün olmamaktadır [21].
2. Eş zamanlı etkileşimler dikkate alındığında, sistem doğrusal olarak büyürken, sistemi tanımlamakta kullanılan durumların sayısı da genellikle geometrik olarak büyümektedir. Bu sebeple, çok sayıda durum ve bu durumlar arasındaki bağlantılar, oldukça fazla karmaşık ve anlaşılması zor bir diyagram oluşturabilmektedir [22].

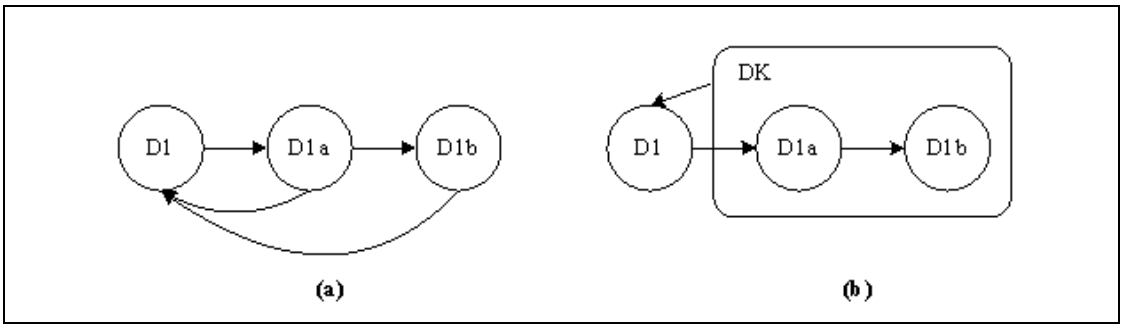
Bu kısıtlamalar simülasyon sisteminin SDM ile modellenmesini güçleştirir. Bunun yanında sistemin bazı bölümlerinde SDM kullanılabilir. Örneğin trafik ışıklarının durum değiştirme geçişlerinin ayarlanmasında SDM'nin kullanılması tercih edilebilir.

2.1.4. Hiyerarşik Eşzamanlı Durum Makineleri (Hierarchical Concurrent State Machines)

SDM modelinin, sistemi oluşturan durumlar arasında herhangi bir gruplandırma ya da hiyerarşik yapılanma gerçekleştirilememe kısıtlamasını gidermek ve böylece karmaşık sistemleri modellemek için Hiyerarşik Eş Zamanlı Durum Makineleri (HEZDM) kullanılmaktadır.

HEZDM, durum grafikleri ile kontrol edilir. Durum grafikleri; sistemi oluşturan durumların, üst durum ve alt durum olarak hiyerarşik bir düzende gruplandırılması

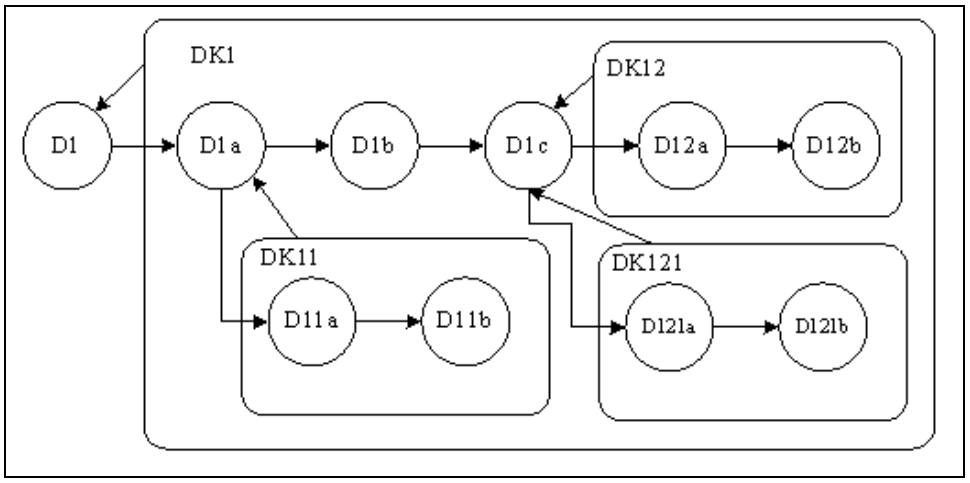
ve kümelenmesi işlemlerini gerçekleştirebilir. Bununla birlikte durumlar arasındaki tekrarlanan geçişler, mantıksal operatör kullanarak (ÖZEL VEYA - XOR) önlenebilmektedir. Gruplanmış durumların meydana getirdiği üst durum, durum kümesi olarak adlandırılabilir. Şekil 2.5. (a) ve (b)'de durumlar arasında tekrarlanan geçişlerin, durum geçiş diyagramı ve durum grafiği ile gösterimi yer almaktadır. Eğer sistem ya da nesne, DK durum kümesinde ise DK kümesinin içinde yer alan alt durumlardan sadece birinde demektir [21].



Şekil 2.5. Durumlar arasındaki tekrarlanan geçişlerin, a) durum geçiş diyagramı ve b) durum grafiği ile gösterimi.

Durum geçişlerini teorik olarak gösteren durum geçiş diyagramlarının gerçekleşmesi için SDM modeli kullanılır. Bu duruma benzer bir şekilde durum grafiklerinin gerçekleşmesi için de HEZDM modelinden faydalanılır [21].

Bir HEZDM, giriş ve çıkış geçişleri, hiyerarşik olarak kümelenmiş SDM ve eylem fonksiyonlarından oluşan bir sistemdir. Şekil 2.6'da durum grafiklerinin gösterildiği basit bir HEZDM gösterilmiştir. Bu sistem ile 10 farklı durum, 13 geçiş sinyali ile kontrol edilmektedir. Örneğin DK1 durum kümesinde yer alan D1a durumundan DK11 durum kümesine geçişi sağlayacak eylem yapıldığında, HEZDM D11a ve D11b durumlarına geçmiş olur. Böylece aynı eylem ile kontrol edilen farklı durumlar tek bir geçiş fonksiyonu ile kontrol edilmiş olur.



Şekil 2.6. Basit bir HEZDM.

Simülasyon sistemleri birbirleri ile ilişkili olan ve olmayan birçok alt sistemden oluşmaktadır. Örneğin bir trafik simülatörü içinde taşıtlar ayrı bir alt sistem, trafik kuralları ayrı bir alt sistemdir ve buna benzer çok sayıda alt sistem vardır. Birbirinden bağımsız farklı alt sistemler aynı eylem ile tetiklenebilirler. Örneğin trafikte art arda giden iki taşıt için arkadan gelen taşıt önden giden taşıtı geçmek için takip mesafesini korumalı, geçiş şeridinin uygunluğunu kontrol etmeli ve hızını ayarlamalıdır. Bu üç durum, taşıt geçme eylemine bağlı olarak gerçekleştirilir. Bu türden hiyerarşik eylemlerin yazılımsal olarak gerçekleştirilebilmesi için HEZDM kullanılabilir.

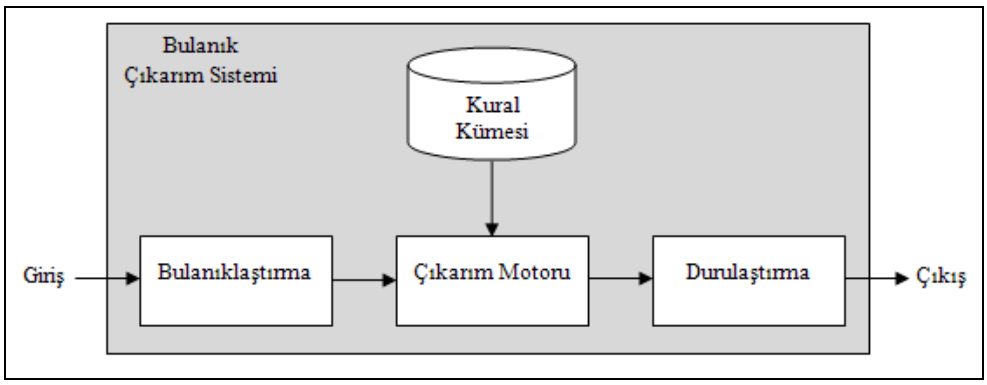
HEZDM, mikro simülasyon ve nano simülasyon uygulamaları için sürücü davranışlarının modellenmesinde kullanılmaktadır [22, 23].

2.1.5. Bulanık Mantık

Bulanık mantık, klasik mantığın yanında, "mutlak doğru" ile "mutlak yanlış" arasında kalan değerlerin de kullanıldığı, kısmi doğruluk kuramıdır [24, 25].

Matematik modellerinin çıkarılması zor olan ve uzman bilgisi kullanılarak modelleme ihtiyacı duyulan karmaşık sistemlerin modellenmesi ve denetlenmesinde bulanık mantıktan faydalanılır. Günümüzde bulanık mantık modeli, kontrol, yönetim ve karar destek gibi çeşitli amaçlara yönelik olarak endüstride uygulanmaktadır.

Bulanık mantık tabanlı sistemler, aldığı giriş verilerini bulanık mantık operasyonları ile işleyerek bir çıkış elde ederler. Bulanık mantık operasyonları bulanıklaştırma (fuzzification), çıkarım motoru (inference engine) ve durulaştırma (defuzzication) aşamalarında gerçekleştirilir. Şekil 2.7'de bulanık mantık modelinde, bir çıkarım sistemi gösterilmiştir.



Şekil 2.7. Bulanık mantık tabanlı sistem blok diyagramı.

Bulanıklaştırma aşamasında, girişteki sayısal değerler üyelik fonksiyonları kullanılarak sembolik değerlere atanır. Bu değerler çıkarım motoru ile kural kümesinde karşılaştırılır ve izlenecek yola karar verilir. Sembolik değerlerin sayısal değerlere atanması için bulanık kesin dönüşüm fonksiyonları kullanılarak durulaştırma işlemleri gerçekleştirilir [18].

Bulanık tabanlı yaklaşım ile kurulan bir sistemin giriş, çıkış ve denetim (çıkarım) tepkisi değerlerinin, bir uzman tarafından yorumlanmasına ihtiyaç vardır. Denetim altındaki sistemin matematik modeline ihtiyaç yoktur.

Bulanık mantık kuramı, kesin ve açık olmayan verilerin kolay, anlaşılır bir şekilde modellenmesine imkân tanır. Bir sürücü içinde bulunduğu trafik ortamını anlamak ve aracına yaptıracağı manevralara karar vermek zorundadır. Trafik simülatöründe, otonom hareket eden taşıtların çevresini algılaması (önünde giden aracın uzaklığı, hızı; hava ve yol koşulları, trafik kuralları vb.) kesin olmayan gözlemlere dayanmaktadır. Otonom hareket etmesi gereken bir vekil taşıt, trafik simülasyonunda kesin olmayan bu algılara dayanarak mantık yürütmekte ve karar vermektedirler [26, 27]. Bununla birlikte bu vekil taşıt, vermiş olduğu kararları uygulama sürecinde, yine

kesin olmayan refleksleri vasıtasıyla uygulamaktadır. Ancak, bulanık mantık kuramı trafik simülasyon modellerinde, özellikle nano simülasyon modelinde kullanımı çok tercih edilmemiştir [28, 29].

2.1.6. Etmen Tabanlı Modelleme

Son yıllarda bilgisayar bilimleri alanında çalışanlar, etmenler ve çok etmenli sistemler konusuna yoğun ilgi göstermektedirler. Yapay zekâ bilimiyle uğraşan bilim adamları tarafından sorulan "Zekâ nedir?" sorusuna benzer şekilde, bilim adamları "Etmen nedir?" sorusunu sormakta ve cevap aramaktadır [30].

Bilgisayar biliminde kullanılan etmen, algılayıcıları yardımıyla içinde bulunduğu çevreyi algılayan ve eylemcileri yardımıyla çevreyi ve ileride çevreye yönelik algılarını etkileyen eylemde bulunan, nesne veya programlar olarak tanımlanabilir [31, 32].

Etmenler, gerçek hayatta ya da yapay bir çevre (artificial environment) içerisinde kullanılmak üzere oluşturulabilirler. Gerçek hayat için tasarlanan etmenler donanım ve yazılım mimarilerinin kullanılmasını gerektirirken, yapay çevre için oluşturulan etmenler de sadece yazılım mimarisi kullanılır.

Etmenlerin temel özellikleri şöyle sıralanabilir [29, 33];

1. Otonomi: Etmenler herhangi bir dış faktörün etkisi olmadan, kendi kontrol mekanizmalarına göre davranırlar.
2. Tepkisel davranma: Etmenler çevresinde olan olayları algılayarak, bu olaylara uygun ve zamanında tepki gösterirler.
3. Öngörülü davranma: Etmenler, çevrelerindeki değişimlere tepki göstermenin yanı sıra, herhangi bir olay olmadan, hedefleri doğrultusunda davranış sergileyebilirler.
4. Sosyal olma: Etmenler, kendilerine verilen görevleri başarmak için insanlarla veya diğer etmenlerle iletişim kurabilirler, yardımlaşabilirler.

Bazı etmenler, sahip oldukları özelliklere bağlı olarak zeki davranışlar sergilerler. Bu türden etmenler zeki etmen terimi ile ifade edilirler. Bazı etmenler ise belirgin bir biçimde otonom davranışlar sergilerler. Bu türden etmenler ise otonom etmen terimi ile ifade edilirler [30].

Etmenler karar verme mekanizmalarına göre kasıtlı – planlı ve tepkisel – plansız etmenler olarak sınıflandırılabilir. Kasıtlı – planlı etmenler, tepkisel – plansız etmenlere göre öngörülü davranma kabiliyetleri daha yüksektir, fakat tepki hızları daha yavaş ve sembolik çevre modeline bağımlılıkları daha fazladır. Bu sebeple kasıtlı – planlı etmenler;

1. Yavaş tepki verirler,
2. Yüksek seviyede zeki davranış sergilerler,
3. Yüksek işlem kapasitesi gerektirirler.

Tepkisel – plansız etmenler ise;

1. Gerçek zamanlı ve hızlı tepki verirler,
2. Düşük seviyede zeki davranış sergilerler,
3. Basit işlem kapasitesi gerektirirler [34].

Etmen tabanlı modelleme tekniği kullanılarak geliştirilen uygulamalarda tamamen kasıtlı, planlı ya da tamamen tepkisel, plansız etmenlerin kullanımı yerine melez etmenler tercih edilir. Melez etmenler, kullanım amacına yönelik olarak her iki karar verme mekanizmasının birtakım özelliklerini taşıyan etmenlerdir [30, 33].

Etmenler program içerisinde nesne yapısında oluşturulurlar. Etmenin davranışları, özellikleri nesnenin üyelik fonksiyonları ve değişkenler ile ifade edilirler. Fakat etmenler kendilerine has bir otonomiye sahip olduklarından dolayı; bir nesne, *public* olarak tanımlanmış bir üyelik fonksiyonuna erişilmesine izin verirken, etmen, duruma göre erişimi engelleyebilmektedir [35].

Etmenler, endüstriyel uygulamalar (üretim, hava trafik kontrol, süreç kontrolü vb.), ticari uygulamalar (bilişim, iş yönetimi vb.), medikal uygulamalar (hasta takip, bilgi değişimi vb.), eğitim uygulamaları (sivil ve askeri) ve bilgisayar oyunları alanlarında yaygın olarak kullanılmaktadır.

Şehir içi trafiği, farklı birçok alt sisteme ve bu alt sistemler arasındaki etkileşimlere sahip karmaşık bir sistemdir. Etmen tabanlı modelleme tekniği, şehir içi trafiğinin benzetimi için geliştirilen simülasyon sistemlerinde kullanabilecek uygun ve verimli bir çözüm yaklaşımıdır [32].

Trafik simülasyon modellerini incelendiğinde, trafik ortamında otonom ve zeki davranış sergileyen unsurların etmen tabanlı modelleme tekniği kullanılarak hazırlanabilir. Trafik sistemi simülasyonu gerçekleştiren birçok uygulamada (sistemde yer alan taşıtların, sürücülerin, trafik ışıklarının, yaya davranışlarının ve bu öğelerin birbirleri ile etkileşimlerinin modellenmesinde) etmenlerin kullanıldığı görülmektedir [36].

2.2. ŞEHİR BİLGİSİ

Trafik simülasyon sistemleri oluşturulurken şehir verisi için coğrafi bilgi sistemleri ve üç boyutlu görsel modeller kullanılmaktadır. Coğrafi bilgi sistemleri yükseklik bilgileri, yol, bina gibi nesnelerin koordinat değerlerini ve boyut bilgilerini kapsar. Fakat görsel modelleri yoktur ve bu veriler görselleştirme için doğrudan kullanılmaz. Coğrafi bilgi sistemlerini görselleştirmek için çözünürlük değerlerine bağlı olarak uygunlaştırılması gerekir. Bu yüzden görsellik için üç boyutlu dosyaların kullanılması daha iyi bir seçim olabilir. Bununla birlikte simülasyon sistemi için gerekli yol, şerit, trafik işaretleri gibi verilerin de bu modele eklenmesi gerekir.

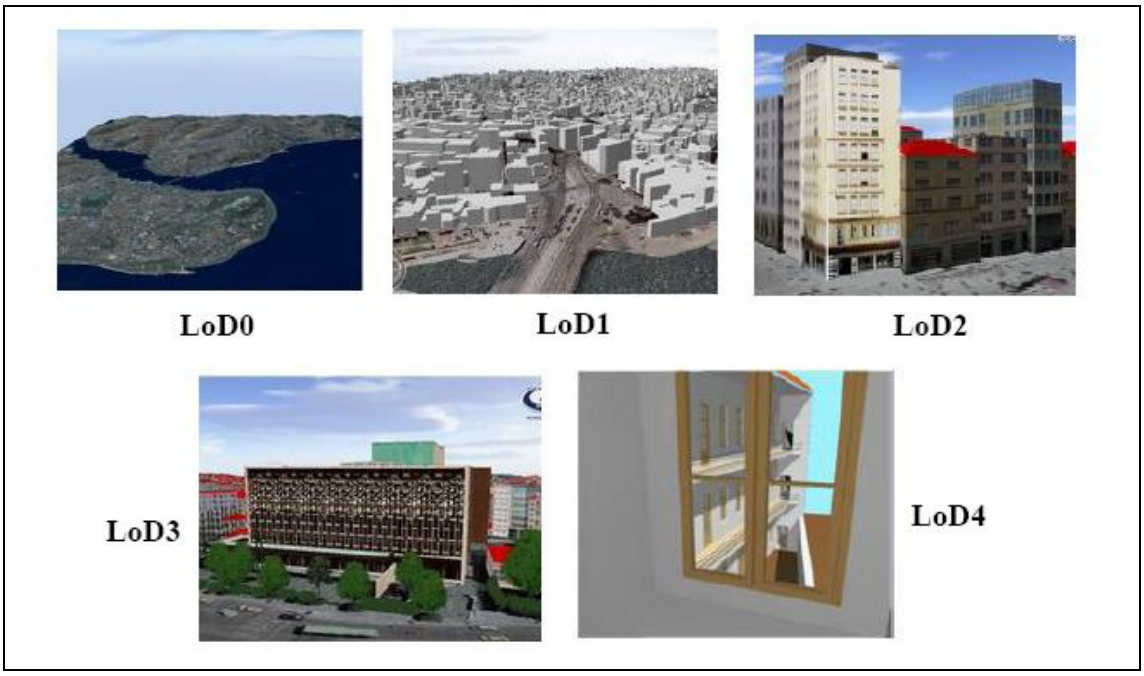
2.2.1. Coğrafi Bilgi Sistemi Verileri

Coğrafi bilgi sistemi, konumu belirlenmiş verilerin kapsanması, yönetimi, işlenmesi, analiz edilmesi, modellenmesi ve görüntülenebilmesi işlemlerini kapsayan yöntemler sistemidir. Coğrafi bilgi sistemleri verilerini oluşturmak için ESRI, MapInfo gibi

firmaların yanı sıra Open Geospatial Consortium (OGC) gibi ortak bir platform da vardır. OGC konsorsiyumu şehir bilgisi verilerini ortak bir mimaride birleştirmek için CityGML olarak isimlendirilen bir ortak işaretleme dili geliştirmiştir.

2.2.1.1. CityGML

CityGML, 3B kent modellerine GML dilini kullanarak veri depolama, veri dönüşümü ve veri değişimi için XML tabanlı ortamların oluşturulmasını sağlar. CityGML'de LoD0, LoD1, LoD2, LoD3 ve LoD4 adı verilen beş ayrıntı düzeyi tanımlanmıştır. Bunlardan LoD0 ayrıntının en az olduğu düzeydir ve yalnız 3B arazi modelini içerir. Model uydu görüntüsü gibi ekstra veriler ile desteklenebilir. Bu düzeyde arazi modeli 3B olmasına rağmen, kent modeli 3B değildir. Çünkü bu düzeyde binalar 3B gösterilmemektedir. Bir üst düzey olan LoD1 ayrıntı düzeyi basit kent modelleme işlemlerinde en çok kullanılan düzeydir. Bu ayrıntı düzeyinde binalar dikdörtgen prizmalar ile çatılar da düz olarak gösterilir. LoD2 ayrıntı düzeyinde ise bina çatı tipleri, bina cephelerinin fotoğrafları ve basit bitki modelleri eklenerek model zenginleştirilir. LoD3 ayrıntı düzeyinde binaların balkonları, duvar ayrıntıları gösterilir. Yüksek çözünürlüklü fotoğraflar bu ayrıntı düzeyinde yapıların dış yüzeylerine yerleştirilebilmektedir. Ayrıca ayrıntılı bitki modelleri ve taşınabilir nesneler LoD3 modellerinde gösterilir. LoD3 ayrıntı düzeyindeki yapılara, odalar, merdivenler, iç duvarlar, mobilyalar gibi bina içinde bulunan nesnelerin eklenmesi ile LoD4 ayrıntı düzeyine ulaşılır [1, 2].



Şekil 2.8. OGC tarafından 3B kent modelleri için belirlenmiş ayrıntı düzeyleri.

2.2.1.2. MapInfo Professional

MapInfo firması tarafından piyasaya sürülmüş, üç boyutlu şehir bilgisi oluşturan bir yazılımdır. MapInfo verilerini tablolarda tutar. Bu yazılımın kullandığı tablolar ayrı ayrı dosyalarda tutulurlar. Bunlar:

1. *.tab: Tablonun yapısını tutan küçük boyutlu bir dosyadır.
2. *.dat: Veri içerir.
3. *.map: Grafik nesnelerini tanımlar.
4. *.id: veriler ile nesneler arasında bağlantı kuran karşılıklı bir referans dosyasıdır.
5. *.ind: Harita objelerini sorgulamak için indeks verisini tutar.

2.2.1.3. ArcGIS

ArcGIS ESRI firması tarafından hazırlanmış, entegre bir coğrafi bilgi sistemi (CBS) yazılımıdır. CBS, yazılım bileşenlerinin ortak kütüphanesi ArcObjects üzerine

kurulmuş bir sistemdir. Shp uzantılı tek bir dosyada nesne, koordinat, desen vb. bilgileri tutulabilir.

2.2.2. 3B Modeller

Yaygın olarak kullanılan 3B dosya biçimleri, *.flt, *.3ds, *.x, *.dxf, *.wrl olarak sayılabilir[37]. Birçok yazılım farklı dosya biçimlerinin birbirlerine dönüştürülmesini gerçekleştirebilir.

1. Flt Model: Multigen Creator yazılımının çıktısıdır. Simülasyon sistemleri için ortam oluşturmak maksadıyla kullanılır. Bu modelde oluşturulan bütün gruplar tek bir ana düğüme bağlanırlar. Gruplar arasındaki ilişkiler bir graf üzerinde belirlenir. Bir model nesne düğümleri, yüz düğümleri, poligonlar, çizgiler ve köşeler ile ifade edilir. Bu veriler OpenGL gibi grafik kütüphaneleri tarafından görüntülenebilmesi için aynı modeldeki hiyerarşik yapı tekrar oluşturulur.
2. 3DS Model: Autodesk firması tarafından oluşturulmuştur. Bu biçim 3 bölümden oluşur: dosya başlığı, materyal listesi ve alt model listesi. Alt model listesi bütün nesneleri kapsar ve bu nesnelerin dönüşüm matrisleri, köşe koordinatları, üçgen yüzleri, normal vektörleri, materyal listeleri ve desen koordinatları yer alır.
3. X Model: Autodesk firması tarafından oluşturulmuştur ve çalıştırılabilir sistem dosyası olarak biçimlendirilmiştir. Her bir dosya veri bloklarından oluşur ve her blokta dosya başlığı ve model bilgisi bulunur. Başlık, blok tipini ve model bilgisinin uzunluğunu tutar. Blok 3DS modelinin asıl birimidir ve renk, desen, köşeler ve diğer özel veri bilgisi burada tutulur. Bu yüzden biçime bağlı olarak okunabilir ve blok mimarisi kullanıcı tarafından yeniden organize edilebilir.
4. WRL Model: Özellikle web sayfaları için tasarlanmıştır. Ekran grafı, ekran düğümü-alan ve model mimarisi olmak üzere üç önemli kavram ile anlatılır. Düğüm 60 farklı tip içerebilir, fakat bir nesne sadece şekil, görünüş, materyal ve geometri tiplerine ihtiyaç duyar. Her düğüm alan serilerini içerir ve bu seriler düğüm için özel fonksiyon değerlerini saklar. Düğümler ekran grafını oluşturur. Nesneler sınırlara bağlı kalmak koşulu ile üçgen yüzlerle oluşturulur. Ayrıntıya göre üçgen sayısı ve büyüklüğü çok fazla sayıda ve düzensiz biçimde olabilir.

BÖLÜM 3

KÜTÜPHANELER

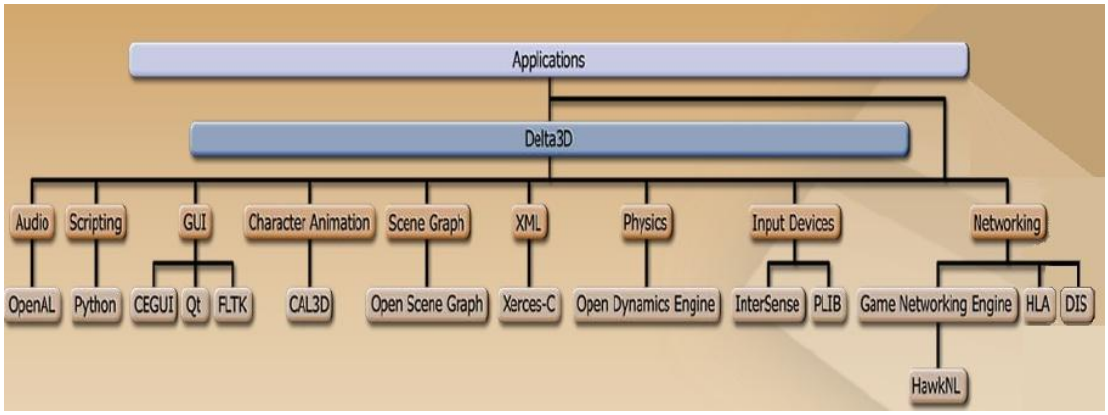
Sürücü adaylarının simülasyon ortamında sürücü becerilerini geliştirmeleri için, gerçek taşıtta bulunan direksiyon, pedallar, korna, sinyal kolları, vites gibi donanımlara benzer donanımlar kullanarak, gerçeğe yakın bir trafik ortamında taşıt sürme faaliyetlerini gerçekleştirmeleri yararlı olabilir.

Sürücünün yaptığı test sürüş performanslarının veri tabanına kaydedilmesi, farklı zamanlarda gerçekleştirilen sürücü performanslarını karşılaştırma imkânı sunar. Sürücü ve eğitimci bu performansları kapsayan raporlardan, öğrenilen veya öğrenilmesi tamamlanamayan sürücü davranışlarını kolaylıkla görebilir.

TSS yazılımında, görüntüleme, fiziksel işlemler ve ses işlemleri için Delta3d, giriş donanımlarından gelen sinyalleri algılamak için PLIB (Portable Game Library), veri tabanı işlemleri için OleDb, raporlama işlemleri için Crystal Reports kütüphanelerinden faydalanılmıştır.

3.1. DELTA3D

Trafik simülasyon yazılımı geliştirilirken grafik ortamı hazırlanması için MOVES (Institute at the Naval Postgraduate School in Monterey) tarafından geliştirilen açık kaynak kodlu Delta3d kütüphanesinden faydalanılmıştır. Delta3d, oyun programları, simülasyon sistemleri geliştirmek için hazırlanmış bir kütüphane yazılımıdır. İçerisinde Open Scene Graph, Open Dynamics Engine, OpenAL gibi birçok grafik kütüphanesini barındırır. Bu kütüphanelere erişmek için Delta3d içerisinde birçok çalışma alanı (namespace) yer almaktadır. Delta3d 2.4.0 kütüphanesinde yer alan çalışma alanları dtCore, dtABC, dtHLAG, dtTerrain, dtDAL, dtGUI, dtGame, dtUtil, dtAudio, dtNet, dtPhyton olarak sıralanabilir.



Şekil 3.1. Delta3D'nin kullandığı açık kaynak kodlu kütüphaneler [38].

dtCore çalışma alanı, temel ortak fonksiyonları kapsar. Giriş aygıtları (klavye, fare, oyun kumandaları), hareket modelleri (Fly, UFO, Walk, Orbit, First Person), çevre koşulları (bulutlar, mevsimler, gün ışığı), desteklenen dosya biçimleri (.3dc, .3ds, .ac, .dw, .flt, .geo, .ive, .logo, .lwo, .lws, .md2, .obj, .osg, .tgz, .x, .zip, .bmp, .dds, .gif, .jpg, .pic, .png, .pnm, .rgb, .tga, .tiff, .txp, .wav), kamera kontrolleri, fizik katmanı gibi bileşenler dtCore tarafından yönetilebilir.

dtABC çalışma alanı, yüksek seviyede uygulama bileşenlerini içerir. Bazı uygulamaları geliştirmede kullanışlıdır. Uygulama şablonları, hava durumu değişiklikleri için ara yüz parametreleri, FLTK pencere birleştirici bileşenleri dtABC tarafından yönetilebilir.

dtHLAGM çalışma alanı, Yüksek Seviyeli Mimarideki (High Level Architecture) simülasyon sistemlerinin haberleşmesi için kullanılan ara yüzdür. Sistemler, dâhili komponentler çalıştırma alt yapısı (Run-Rime Infrastructure -RTI) kullanılarak birleştirilirler.

dtTerrain çalışma alanı, arazi yüklemek, görüntülemek ve düzenlemek için geliştirilmiştir. GEOTIFF, DTED haritalarını destekler.

dtDAL çalışma alanı, aktör oluşturmak, oluşturulan aktörlere erişmek ve aktörleri işlemek için geliştirilmiştir. Aktörler ses, arazi, karakter, desen gibi tiplerde olabilir. Oluşturulan aktörler, aktör kütüphanesine kayıt edilebilir.

dtGUI çalışma alanı, Crazzy Eddie GUI (CEGUI) grafik ara yüz kütüphanesi kullanarak, grafik ara yüzü oluşturmak için kullanılabilir.

dtGame çalışma alanı, karmaşık oyunlar ve eğitim uygulamaları hazırlamak için komple bir mimari sağlar. Oyun Yöneticisi (Game Manager), yerel veya istemci-sunucu ortamlarında aktörler ve komponentlerin haberleşebileceği yüksek seviyeli bir alt yapıdır.

dtUtil çalışma alanı, Delta3d'nin sürekli kullandığı temel nesneleri kapsar.

dtAudio çalışma alanı, simülasyonlarda ses çaldırmak için kullanılır. 2B/3B sesler çaldırılabilir. Ses çaldırma kontrolleri (oynat, durdur, duraklat gibi) yapılabilir. Ses donanımının sahip olduğu özellikler kullanılabilir.

dtNet çalışma alanı, çok kullanıcılı haberleşme desteği sağlar. Sunucu-istemci mimarisini destekler. Güvenilir ve güvenilir olmayan paket transferi yapılabilir.

dtPhyton çalışma alanı, Phyton script dilini kullanarak simülasyon hazırlamak maksadıyla hazırlanmıştır.

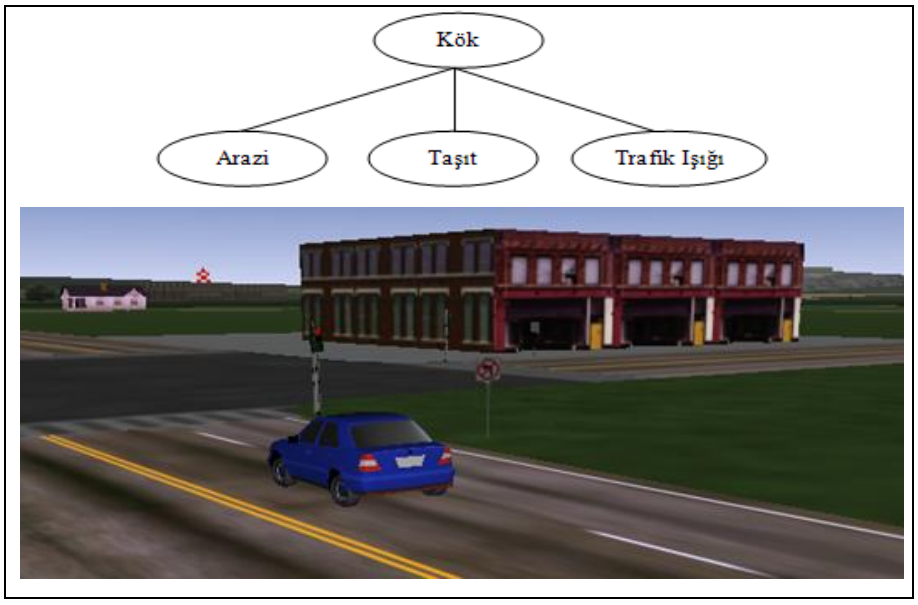
3.1.1. Open Scene Graph

Delta3d görüntüleme (render) işlemi için Open Scene Graph (OSG) kütüphanesini kullanmaktadır. OSG açık kaynak kodlu yüksek performanslı üç boyutlu bir grafik kütüphanesidir. C++ ve OpenGL kullanılarak yazılmıştır. OSG görsel simülasyon, oyun, sanal geçeklik ve modelleme uygulamalarında kullanılır. OSG sadece kameranın görüntülendiği alanın oluşturulması (frustum culling), görünmeyen alanların gerçek zamanlı olarak belirlenerek görüntüleme dışında bırakılması (occlusion culling), belirlenen piksel boyutlarındaki görüntülerin kırpılarak

görüntülemesi (small-feature culling), çekirdek ekran sahnesindeki görüntü listeleri, LOD düğümleri gibi büyük miktarda performans artışı sağlayan birçok konsepti destekler [39, 40].

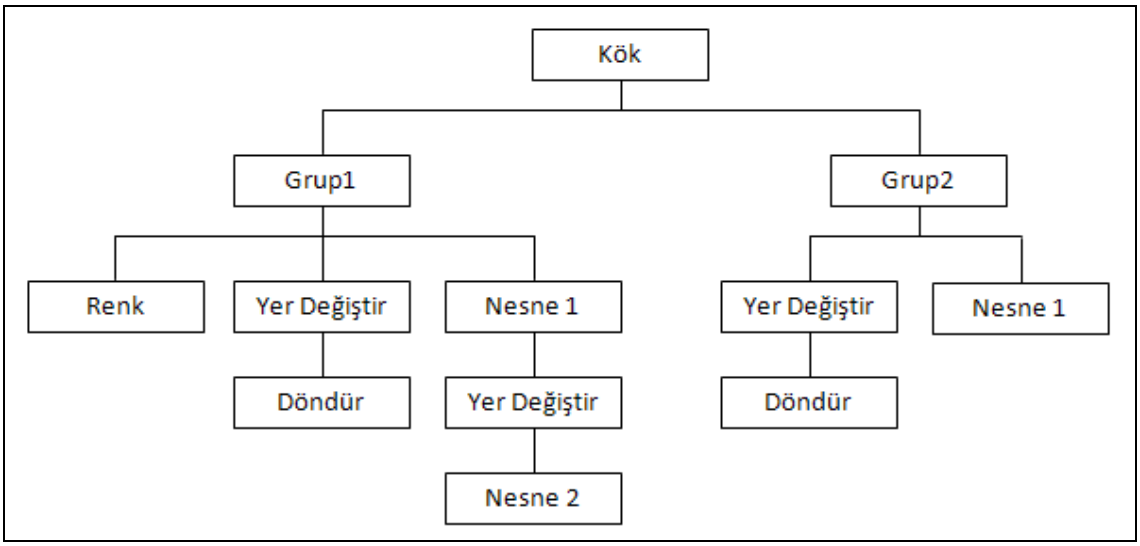
3.1.1.1. Sahne Grafı

Sahne grafı, bilgisayar grafiklerinde bir sahnede görüntülenen elemanların organize edilmesi için kullanılan bir veri yapısıdır. Sahne grafı, ağaç veri yapısına benzer bir şekilde elemanları, mantıksal ve mekânsal olarak organize eder. Sahne düğümü birçok çocuk düğümüne, benzer şekilde çocuk düğümlerde özelliklerin, dönüşümlerin ve nesnelerin ifade edildiği birçok düğüme sahiptirler.



Şekil 3.2. Basit bir sahne grafı.

Sahne grafının en üstünde bir kök düğüm (root node) vardır. Kök düğümün altında görüntülenecek elemanların organize edildiği grup düğümler (group node) yer alır. Grup düğümlerin altında yer alan yaprak düğümler (leaf node) sahnelenecek nesneleri ve bu nesnelerin özelliklerini içerirler. Yaprak düğümlerin çocuk düğümleri yoktur. Şekil 3.2'de bir sahne görüntüsü ve bu görüntüyü oluşturan sahne grafı gösterilmiştir.



Şekil 3.3. Örnek bir sahne graf ağacı.

Sahne grafında yer alan düğümlerin işlenme yönü soldan sağa ve yukarıdan aşağı doğrudur. Şekil 3.3'de gösterilen sahne graf ağacında iki grup düğüm yer almaktadır. Grup1 düğümünde yer alan Nesne1 düğümüne bir renk atanarak, daha sonra yer değiştirme ve döndürme dönüşümlerine uğratılmıştır. Nesne2, Nesne1'in bütün özelliklerine sahip olmakla beraber, bunların yanında bir defa daha yer değiştirmeye uğratılmıştır. Grup2'de yer alan Nesne3 ise yer değiştirme ve döndürme dönüşümlerine uğratılmıştır. Üst düğümlerde olan herhangi bir değişiklik miras kalıtımı ile alt düğümlere yayılır. Bu davranış ile farklı düğümlerden oluşmuş pek çok nesne aynı özelliklerden yararlandırılabilir. Örneğin bir şasi nesnesi ve dört tekerlek nesnesi bir grup düğümüne bağlanarak bir taşıt modeli oluşturulabilir. Bu modelde grup düğümüne yer değiştirme ve ya döndürme gibi dönüşümler uygulanarak taşıt hareket ettirilmiş olur.

Sahne grafı, düşük seviyede görüntüleme işlemi geçekleştiren OpenGL, DirectX gibi API (Application Programming Interface) uygulamalarını kullanarak bilgisayar grafiklerinin işlevselliğini arttırır. Farklı mesafelerdeki nesne görüntülerinin ayrıntı düzeyleri (LoD) belirlenebilir. Ayrıca görüntülenmeyecek sahnelerin yüklenmesi önlenerek donanımların çalışması optimize edilir. Farklı biçimlerdeki 3B dosyalar sahne grafları ile işlenebilir. Bir kez belleğe yüklenen dinamik 3B veri, sahne grafiği veri yapısı ile kolayca yönetilebilir. Bununla birlikte sahne grafı, bir dosya

biçiminden diğerine dönüştürmek için etkin bir araçtır. Acrobat 3D, Adobe Illustrator, AutoCAD, CorelDRAW, OSG, VRML97 ve X3D gibi platformlar 3B uygulama geliştirmek için sahne grafı kullanırlar.

3.1.1.2. OSG düğümleri

OSG, yaprak düğümlerin geometrik verilerini saklamak ve gerektiğinde çağırmak için osg::Geode sınıfını kullanır. Geode, "geometry node" ifadesinin kısaltmasıdır. Bir Geode görüntüsü oluşturmak için ilgili nesnenin addDrawable() metodu ile uygulamaya eklenmesi gerekir.

```
osg::ref_ptr<osg::Geode> geode = new osg::Geode;
geode->addDrawable( geom.get() );
```

OSG grup düğümleri için osg::Group sınıfını kullanır. Grup düğümleri çok sayıda çocuk düğüme sahip olabilirler. Bu düğümler OSG'nin kalbidirler. Çünkü geliştirilen uygulamada yer alan elemanlar sahne grafı üzerine gruplar kullanılarak yerleştirilebilir. Grup özelliklerinin kendine bağlı ebeveyn veya çocuk düğümlerine miras olarak aktarılması, sahne grafının yönetilmesinde kolaylık sağlar. Grup düğümler dönüşüm düğümü, LoD düğüm ve anahtar düğüm olarak kullanılırlar.

osg::Transform sınıfı, grup düğümlerine ait geometrik verilere dönüşüm işlemlerinin (yer değiştirme, döndürme vb.), uygulanması için hazırlanmıştır. Dönüşüm işlemi grupla birlikte grubun bütün çocuk düğümlerine de uygulanır. OSG dönüşüm işlemleri için osg::MatrixTransform veya osg::PositionAttitudeTransform metotlarını kullanır.

```
osg::ref_ptr<osg::MatrixTransform> mt = new osg::MatrixTransform;
mt->setReferenceFrame( osg::Transform::ABSOLUTE_RF );
```

Nesnelerin farklı mesafelerdeki ayrıntı düzeylerinin görüntülenmesini sağlamak amacıyla osg::LOD düğümleri kullanılır. Her bir çocuk düğümün görünebilirliği minimum ve maksimum aralık değerleri verilerek belirlenir. Belirtilen değerler içerisinde ilgili çocuk düğüm, ebeveyn düğüme eklenerek görüntülenmesi sağlanır.

```
osg::ref_ptr<osg::Geode> geode;
// ...
osg::ref_ptr<osg::LOD> lod = new osg::LOD;
// Düğümü 0.f <= aralık  < 1000.f değerinde göster
lod->addChild( geode.get(), 0.f, 1000.f );
```

Sahne grafı üzerinde bir grup düğümünde yer alan çocuk düğümler görüntülenirken, seçme işlemi yapılarak (bazı çocuk düğümlerini göster, bazılarını gösterme gibi) sahne grafı oluşturmak için osg::Switch sınıfındaki anahtar düğümler kullanılmalıdır. Anahtar düğümler 3B bir modelin farklı görüntüleri için kullanılırlar. Örneğin bir savaş oyununda yer alan bir aracın sağlam hali ve bir bomba ile vurulmuş hasarlı hali aynı modelde bir grup düğüme atanıp, aracın isabet alma durumuna göre görüntülenecek çocuk düğümler anahtarlanabilir.

```
osg::ref_ptr<osg::Group> group0, group1;
// ...
// Bir anahtar düğüm oluştur ve iki grup düğüm ekle
osg::ref_ptr<osg::Switch> switch = new osg::Switch;
// İlk grup düğümü göster
switch->addChild( group0.get(), true );
// ikinci grup düğümü gösterme
switch->addChild( group1.get(), false );
```

3.1.1.3. OSG kütüphaneleri

3B grafik uygulamaları geliştirirken kullanılan OSG yazılımının temelinde 3 tane kütüphane yer almaktadır.

1. osg kütüphanesi, sahne grafı üzerinde yer alan düğümlerin oluşturulmasını sağlayan sınıfları kapsar. Bununu yanında vektör ve matris matematiği, geometrisi ve görüntülenecek ve yönetilecek durum özeliklerine ait sınıfları da içerir.
2. osgUtil kütüphanesi, sahne grafının işletilmesi, içeriğinin görüntülenmesi, iyileştirilmesi ve istatistik hesaplarının yapılmasını sağlayan sınıf ve fonksiyonlara sahiptir. Bununla birlikte delaunay üçgenlemesi, üçgen stripifikasyonu ve desen işlemlerini yürütecek sınıfları vardır.
3. osgDB kütüphanesi, 3B modellerin saklandığı veritabanlarının gösterilmesi için gerekli sınıfları ve fonksiyonları içerir. Birçok 2B/3B grafik dosyaları

(COLLADA, LightWave (.lwo), Alias Wavefront (.obj), OpenFlight (.flt), TerraPage (.txp), Carbon Graphics GEO (.geo), 3D Studio MAX (.3ds), Peformer (.pfb), AutoCAd (.dxf), Quake Character Models (.md2), Direct X (.x), Inventor Ascii 2.0 (.iv), VRML 1.0 (.wrl), Designer Workshop (.dw), AC3D (.ac), resim formatları (.rgb), (.gif), (.jpg), (.png), .(tiff), (.pic), (.bmp), (.dds), (.tga) ve quicktime) için giriş çıkış işlemlerini gerçekleştirebilecek bir kayıtlığa sahiptir. osgDB, büyük çaplı verilerin aktarılması ve yüklenmesi işlemlerini destekler.

3.1.2. Open Dynamics Engine

Delta3d fizik işlemlerini Open Dynamics Engine (ODE) kütüphanesi ile gerçekleştirir. ODE katı cisim dinamiklerinin benzetiminde yüksek performans sağlayan bir kütüphanedir. Platformdan bağımsız çalışan C/C++ kullanılarak kolayca erişilebilir. Birçok bilgisayar oyununda, 3B yazarlık araçlarında ve simülasyon araçlarında kullanılmaktadır. ODE ile katı cisim dinamiklerindeki motor, çarpışma, eklemler (joints) gibi birçok fiziksel durum modellenebilir. Özellikle araçların, sanal ortam nesnelerinin ve sanal öğelerin simüle edilmesinde kullanışlıdır [41].

Genel olarak bir simülasyon işlemi şöyle yürütülür [42]:

1. Hareketliler için bir ortam oluştur,
2. Cisimler oluştur,
3. Bütün cisimlerin başlangıç durumlarını ayarla,
4. Eklemleri oluştur,
5. Eklemleri cisimlerle birleştir,
6. Bütün eklemlerin parametrelerini ayarla,
7. Çarpışma uzayını oluştur,
8. Temas edilen eklemler için grup oluştur,
9. Simülasyon döngüsü;
    a. Cisimlere güç uygula,
    b. Eklem parametrelerini uygunlaştır,
    c. Çarpışma testi yap,

d. Her çarpışma noktası için bir temas eklemi oluştur ve gruba ekle,
e. Simülasyonu ilerlet,
f. Temas grubundaki bütün eklemleri kaldır,

10. Hareketlileri ve çarpışma uzayını yok et.

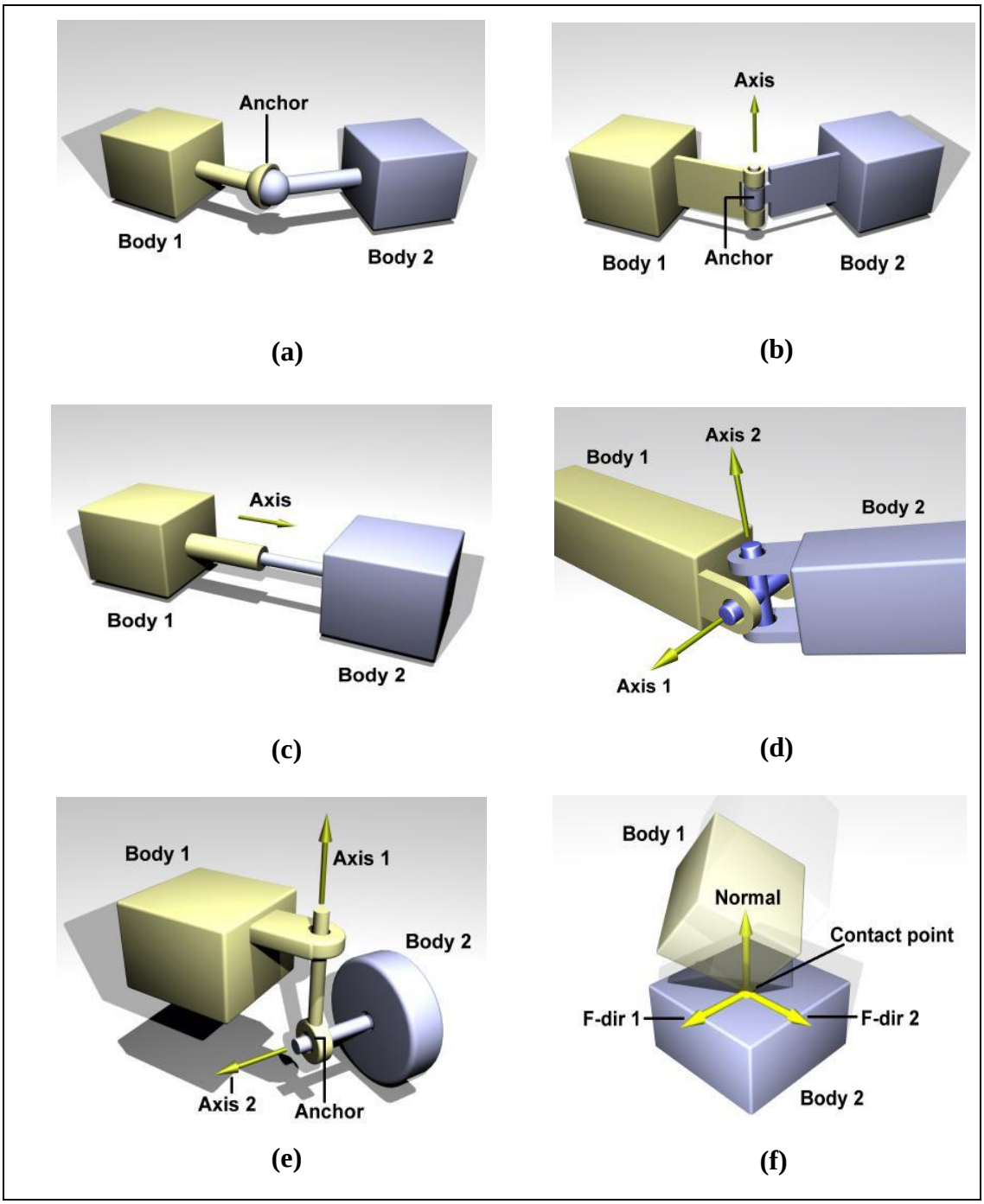
Simülasyon döngüsünden anlaşılacağı üzere simülasyon sistemlerinde fizik işlemleri önemli bir yer tutmaktadır.

3.1.2.1. Eklemler (joints)

Taşıtlar gövde ve tekerlerden oluşmaktadır. Simülasyon sisteminde bir taşıtı oluşturmak için gövde ile her bir tekerin birbirlerine eklemlenmesi gerekir. Bu yüzden öncelikle taşıtın gövde ve tekerlek 3B modelleri ayrı ayrı nesnelere yüklenirler. Taşıtın gövde ve teker verisini içeren nesnelerin birbirlerine eklemlenmesi gerekir. Bu amaçla bir eklem grubu oluşturulur. Şekil 3.4'den de görüleceği üzere ODE kütüphanesi farklı tiplerde eklemler kullanmaktadır. ODE, bu eklemleri şöyle listelemektedir (Şekil 3.4'de belirtilen numaralarına göre):

1. Toplu soket eklem (ball and socket joint),
2. Menteşe eklem (hinge joint),
3. Kaydırıcı eklem (slider joint),
4. Evrensel eklem (universal joint),
5. Menteşe2 eklem (Hinge2 joint),
6. Kontak eklem (contact joint).

Hareket ve manevra kabiliyetleri olan bir cisim için kullanılması gereken eklem tipi Menteşe2 eklemidir.



Şekil 3.4. Eklem tipleri [42].

Menteşe2 eklemde farklı noktalara yerleştirilmiş iki menteşe bağlantısı yer alır. Şekil 3.4. (e)'de Eksen 1 (Axis 1) bağlantısı taşıtın direksiyon manevralarını, Eksen 2 (Axis 2) bağlantısı ise tekerin dönmesini kontrol eder. Çapa (anchor), bir taşıt için gövde ve tekerlerin birleştirileceği vektörel büyüklük içeren birleştirme noktasıdır. Taşıt dinamiğinde bulunan motor gücü, tork değeri, süspansiyon verisi, hata azaltma

parametresi (Error Reduction Parameter - ERP) ve karıştırma kısıtlama kuvveti (Constraint Force Mixing - CFM) değerleri eklem grubuna parametre değeri olarak girilebilir. Böylece taşıtın tekerlerine güç uygulanarak taşıt hareketi sağlanır. Simülasyonda taşıt ilerlerken verdiği tepkiler, gerçek hayatta bir taşıtın hareket ederken göstereceği tepkilere benzerdir.

3.1.3. OpenAL

OpenAL, çok kanallı, üç boyutlu konumsal ses efektlerini çalıştırmak için kullanılan bir ses API'sidir. *Creative* tarafından geliştirilmesine devam edilen OpenAL, yeni nesil Windows oyunları için donanım hızlandırmalı ses desteği sağlamaktadır. OpenAL Linux, MacOS 8/9, Windows ve BeOS işletim sistemleri tarafından da desteklenmektedir.[43]

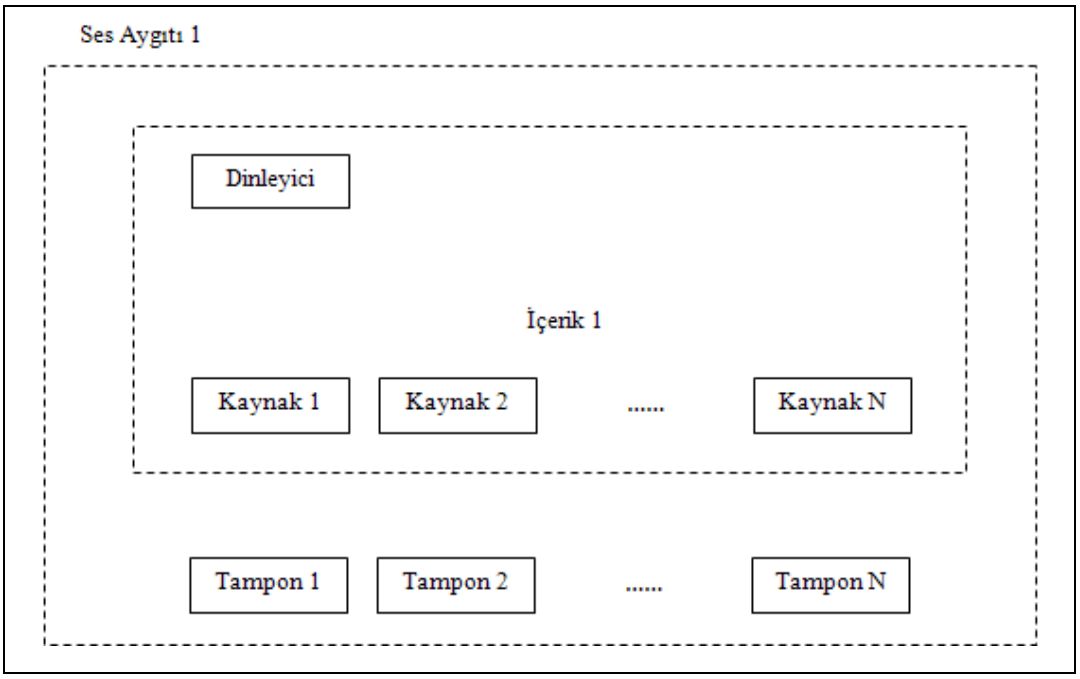
OpenAL bir komut kümesidir. Programcı bu komut kümesi ile 2B/3B özellikte, birden çok ses kaynağını ve bir dinleyiciyi kontrol edebilir. OpenAL'in birleştirilmiş komutları kullanılarak ses kaynaklarının ve çıkış tamponuna aktarılan sesin kontrolü gerçekleştirilir. Ses arabirimlerini kontrol ederken dışarıdan kullanılan OpenAL komutları önemsenmeyecek değerde de olsa, bir ses gecikmesine sebep olur.

OpenAL komutlarını kullanan bir program, hoparlör veya kulaklık gibi bir donanıma aktarılacak çıkış için kullanılacak ses aygıtının çağrılması ile başlar.

```
// Ses aygıtının başlatılması
Device = alcOpenDevice(NULL);
```

Ses aygıtı ile iletişim kurulduktan sonra yapılması planlanan işlemler OpenAL komutları kullanılarak ses aygıtına yüklenir. Bu işlemlere, bir ses dosyasını oynatmak, ses dosyasını defalarca oynatmak, ses aygıtlarında aynı anda birden fazla ses dosyası oynatmak, ses aygıtı kullanarak kayıt işlemi yapmak örnek olarak verilebilir.

OpenAL kaynaklar (sources), tamponlar (buffers) ve bir dinleyici (listener) olmak üzere 3 temel bileşen veya nesneden oluşur. Şekil 5.1'de bir ses aygıtının bileşenleri gösterilmektedir. Her bir nesne diğerlerinden bağımsız olarak değiştirilebilir ve bu değişiklik diğer nesnelerin özelliklerini etkilemez. Uygulama, işlemlerin etkisini kaldıran modları ayarlayabilir. OpenAL yordam çağrıları ile modlar ayarlanır, nesneler özelleştirilir.



Şekil 3.5. Ses aygıtı bileşenleri.

OpenAL, her bir olay için tek bir dinleyici nesnesi kullanır ve bir ismi yoktur. Dinleyici nesnesinde ses çıkışını etkileyen birçok özellik tanımlanır. Dinleyiciyi kontrol ederek geliştirilen uygulama kullanıcının sanal dünyadaki deneyimlerini kontrol eder. Dinleyici nesnesine aktarılan pozisyon, hız ve kazanç parametreleri ile çıkış akışı etkilenir. 3B hale getirmek için kullanılan metot donanım bağımlıdır.

OpenAL her bir olay için birden fazla kaynak kullanabilir. Kaynaklar pozisyon, hız ve tampon gibi verileri belirler. Bir kaynağın özellikleri kontrol edilerek bir uygulama tampondan, kaynağa aktarılan veri çerçevesinde değiştirilebilir ve parametrik bir hale getirilebilir. Bununla birlikte aktif kullanılan tampondaki içerik ses verisi üzerinde oynat, duraklat, durdur gibi işlemler gerçekleştirilebilir.

```
// Kaynakları oluştur
alGenSources(1,source);
```

OpenAL, bir ses aygıtı için bütün kaynaklar tarafından paylaşılan birden fazla tampon kullanmayı destekler. Kullanılacak ses verisi bir tampon alanında saklanır. Uygulamalar ile tampon alanına ses verisi yüklenebilir, istenebilir ve boşaltılabilir. Ses verisi tamponda farklı biçimlerde saklanır. Sıkıştırılmış veya sıkıştırılmamış biçimlerdeki ses verisi ile ilgili olası işlemler harici bir vasıta kullanılmadan tampon tarafından yürütülebilir. Kaynak ve dinleyici nesnelerinin aksine tampon alanları paylaşılabilir. Tamponlar kaynaklara göre ayrılırlar. Bir tampon alanı birden fazla kaynak tarafından kullanılabilir. Böylece sürücüler ve donanım tarafından gerçekleştirilen depolama ve yürütme işlemleri iyileştirilir.

```
// Tamponları oluştur
alGenBuffers(NUM_BUFFERS, g_Buffers);
```

OpenAL ile program geliştirilirken bellek kullanımı dinamik olarak yürütülür. Bu yüzden uygulamalarda ses API'sinin kullanımı tamamlandığında mutlaka belleğin boşaltılması gerekir.

3.2. PLIB

PLIB, açık kaynak kodlu taşınabilir bir oyun geliştirme platformudur. 1997 yılında Steve Baker tarafından yazılmış ve LGPL lisansına sahiptir. PLIB sahip olduğu fonksiyonlar ile kullanıcılarına ses efektleri hazırlama, 3B grafiksel işlemler, pencere işlemleri, yazı tipi oluşturma, grafik ara yüzü oluşturma, 3B matematiksel işlemleri gerçekleştirme imkânları sağlamaktadır. Platformdan bağımsız olarak çalıştırılabilmektedir. PLIB içerisinde yer alan oyun kumanda kütüphanesi, eksen ve buton fonksiyonlarına sahip birçok oyun kumanda donanımına destek vermektedir [44].

PLIB, giriş aygıtlarında meydana gelen olayları takip etmek için Joystick Wrappers (JS) komponentini kullanır. JS, kullanılan oyun kumanda sayısında herhangi bir kısıtlaması olmayan, taşınabilir bir ara yüzdür. Bu işlem, işletim sistemi API'leri

kullanılarak gerçekleştirilmektedir. Geliştirilen bir uygulamaya JS kütüphanesi eklenerek, çok sayıda giriş birimi aynı uygulamada kullanılabilir.

Oyun kumanda aygıtları ile eksen (axis) ve buton (button) giriş olayları gerçekleştirilebilir. Eksen girişleri direksiyon çevrilmesi, gaz pedalına basılması gibi analog veri değerlerini içerir. Buton girişleri ise ilgili butonların durumlarını içeren sayısal (0,1) verilerdir. JS kullanarak oyun kumanda veya herhangi bir giriş biriminde gerçekleşen anlık olaylar kolaylıkla takip edilebilir.

3.3. OLEDB

Veritabanları, diğer uygulamalar ile standartlaştırılmış yapılar kullanarak iletişime geçerler. Bu yapılar sayesinde platformdan bağımsız olarak istenilen veritabanı kolaylıkla kullanılabilir. OleDb, veritabanında tutulan bilgilere erişmek için kullanılan ODBC (Open Database Connectivity - Açık Veritabanı Bağlantısı) standartlarında bir kütüphanedir. Microsoft firması tarafından geliştirilmiştir.

OleDb ile veritabanına istemci olarak erişim sağlanır. OleDb, ODBC'nin yaptığı gibi ilişkisel veritabanları için standart bir ara yüz sağlamanın yanı sıra, ilişkisel olmayan veri kaynaklarına da bağlantı kurulmasını sağlayabilecek özelliklere sahiptir.

Bir uygulamada OleDb sağlayıcısı kullanarak veritabanı işlemlerini gerçekleştirebilmek için öncelikle uygulamaya ilgili kütüphane dosyalarının dâhil edilmesi gerekir. Bu aşamadan sonra veritabanı ile ilgili işlemler için aşağıdaki algoritma yürütülür:

1. Bağlantı oluştur ve aç,
2. Sorgu oluştur,
3. Sorgu sonuçlarını veri kümesine aktar,
4. Veri Kümesini göster,
5. Bağlantıyı kapat.

OleDb kütüphanesinde veritabanı ile bağlantı kurmak ve işlemler tamamlandıktan sonra veritabanı bağlantısını kapatmak için OleDbConnection nesnesi kullanılır. OleDbConnection nesnesi kurulurken, bağlantı kurulacak veritabanı sağlayıcısı ve kaynak dosya konumu parametre değerleri ile yapılandırılır.

```
// Bağlantı nesnesi kuruluyor
OleDbConnection^ vt_baglan= gcnew OleDbConnection("Provider=
                            Microsoft. Jet.OLEDB.4.0;DATA
                            Source=ornek.accdb");
// Bağlantıyı Aç
vt_baglan->Open();
// ……….……
// işlemler
// ……….……
// Bağlantıyı kapat
vt_baglan->Close();
```

Veritabanı ile bağlantı açıldıktan sonra kullanıcı ilgili veritabanında geliştirdiği uygulama için planladığı işlemleri yapabilir. Veritabanında yapılacak işlemler tablo oluşturmak, veri eklemek, veri aramak, veri güncellemek, veri silmek, tablo silmek, veritabanı silmektir. Bütün veritabanı yönetim sistemlerinde bu tip işlemleri gerçekleştirmek için SQL (Structured Query Language – Yapılandırılmış Sorgulama Dili) isminde ortak bir sorgu dili kullanılmaktadır. SQL'in kendine özgü bir sözdizimi vardır ve SQL ile yalnızca veri tabanı üzerinde işlem yapılabilir.

OleDb kütüphanesinde SQL sorgusu oluşturmak ve ilgili sorguyu çalıştırmak için OleDbCommand nesnesi kullanılır. OleDbCommand nesnesi kurulurken, kullanılacak bağlantı nesnesi ve çalıştırılacak sorgu cümlesi değerleri belirlenebilir veya bu değerler ilgili özelliklere değer atayarak da sonradan gerçekleştirilebilir. OleDbConnection nesnesinin Connection özelliği bağlantı nesnesini, CommandText özelliği sorgu cümlesini atamak için kullanılır. Hazırlanan komutun çalıştırılması için ExecuteNonQuery() metodu kullanılır. Bu metot, özellikleri belirlenen SQL komutunu bağlantı kurulan veritabanında çalıştırır.

```
// Komut nesnesini oluştur
OleDbCommand sql_komut;
// Komut özelliklerini belirle
sql_komut.Connection=vt_baglan;
//tablo1 tablosunun ad, hobi alanlarına İsmail, Spor verileri ekle
//sorgusunu oluştur
```

```
sql_komut.CommandText="INSERT INTO tablo1(ad,hobi) VALUES
                              ('İsmail','Spor')";
// Sorguyu çalıştır
sql_komut.ExecuteNonQuery();
```

Veritabanında ekleme, silme, güncelleme gibi işlemler için hazırlanmış sorguların çalıştırılmasında ExecuteNonQuery() metodu kullanılır ve ilgili değişiklikler veritabanına doğrudan uygulanır. Veritabanında bulunan verileri veya özellikleri belirlenmiş verileri görüntülemek için SQL sorgusundan elde edilen sonuçların veri kümesine aktarılması gerekir. OleDb kütüphanesinde bu tip sorgulardan elde edilen sonuçları tutmak için kullanılan nesnelerden biri OleDbDataReader nesnesidir. OleDbDataReader nesnesini doldurmak için komut, ExecuteReader() metodu kullanılarak çalıştırılmalıdır. İşlenen tablodan elde edilen sonuçlar, sorguda istenen alan değişkenlerini içeren yapıdaki bir diziye aktarılır. Dizideki eleman sayısı, veritabanında sorgudan etkilenen toplam satır sayısı kadardır. Basit bir döngü kurularak elde edilen verilerin içerikleri görüntülenebilir.

```
// Komut nesnesini oluştur
OleDbCommand^ sql_komut=gcnew OleDbCommand();
// Komut özelliklerini belirle
sql_komut->Connection=vt_baglan;
// tablo1 tablosundaki bütün alanları getirecek sorguyu oluştur
sql_komut->CommandText="SELECT * FROM tablo1";
// Sonuçların aktarılacağı nesneyi oluştur
OleDbDataReader ^okunan_veri;
// Sorguyu çalıştır ve veri kümesine aktar
okunan_veri=sql_komut->ExecuteReader();
// Aktarılan değerleri getir
while(okunan_veri->Read())
{
   // Sonuç dizisi satırları yazdırılıyor
   System::Console::WriteLine(okunan_veri["ad"]);
   System::Console::WriteLine(okunan_veri["hobi"]);
}
```

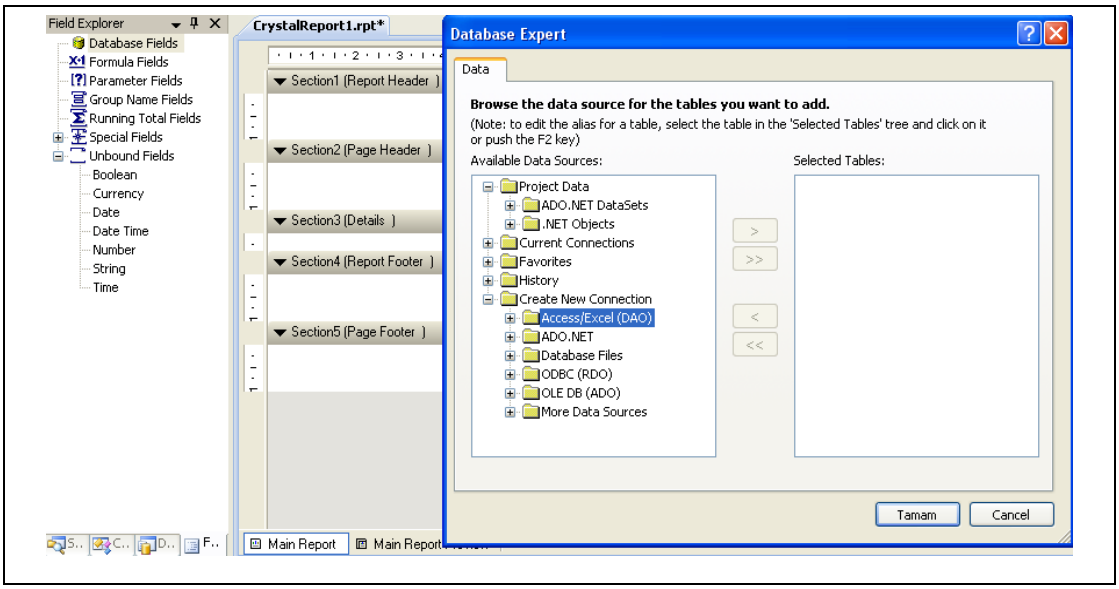
Sorgudan elde edilen sonuçları özel nesnelere veya raporlara aktarmak için, sorgu sonuçları ve bu özel nesneler arasında adaptasyonu sağlayacak OleDbDataAdapter nesnesi kullanılır. Bu nesne parametre veya özellik olarak belirtilen bağlantı ve sorgu ifadelerinden elde ettiği sonuçları veri tablosuna aktarabilir. Oluşturulan bu veri tablosu, özel nesnelerin veya raporların veri kaynağı olarak gösterildiğinde, veritabanı sorgusu sonucu elde edilen veri kümesi, ilgili nesnelere aktarılmış olur.

```
// Veri adaptörünü parametre değerleri ile oluştur
OleDbDataAdapter^ veri_adap = gcnew OleDbDataAdapter(”SELECT * FROM
                              tablo1”, vt_baglan);
// Veri kümesi oluştur
DataSet veri_kume = gcnew DataSet();
// Veri kümesini doldur
veri_adap->Fill(veri_kume,”sonuc”);
// Rapor belgesini oluştur ve veri kümesini rapora yükle
ReportDocument ^Rapor = gcnew ReportDocument();
Rapor->Database->Tables[0]->SetDataSource(vei_kume->Tables[0]);
```

3.4. CRYSTAL REPORTS

Crystal Reports (CR), büyük çaplı veri kaynaklarını anlaşılabilir ve görsel biçimde raporlamak için kullanılan bir uygulamadır. Birçok yazılım geliştirme platformunda olduğu gibi MSVS yazılımının 2003’ten, 2008’e kadar olan sürümlerinde bir raporlama aracı olarak kullanılmıştır.

CR, SQL, Oracle, MySql, Access, Excel, Word, Exchange Server, text dosyaları vb. gibi bilgi kaynaklarına veya veri tabanlarına bağlantı kurarak, esnek raporlama teknikleri ile ileri düzeyde raporlama yapabilir. CR, kullanıcılara sunduğu grafik ara yüzü ile raporda kullanılacak veri kaynaklarının ve rapor tasarımının hazırlanmalarını kolaylaştırır. Şekil 3.6’da MSVS 2008’de kullanılan CR grafik ara yüzü gösterilmiştir.



Şekil 3.6. MS VS 2008 CR ara yüzü

CR grafik ara yüzü "Database Expert" penceresi kullanılarak, raporlanması istenen veritabanı ile bağlantı kurulup, raporlanacak tablolar ve alanlar seçilir. Tasarım amaçlarına göre raporda formül, gruplama, tarih bilgisi gibi veriler gösterilmek isteniyorsa, ilgili alanlara Field Explorer paneli kullanılarak erişilebilir. Rapor çıktısı beş bölümden oluşmaktadır.

1. Rapor başlığı (Report Header): Raporda gösterilmesi planlanan başlık bilgisidir.
2. Sayfa başlığı (Page Header): Veri kaynağından getirilecek alanları ifade edecek başlık bilgisidir.
3. Ayrıntılar (Details): Veri kaynağından getirilecek alanlardır.
4. Rapor altbilgisi (Report Footer): Raporda gösterilmesi planlanan alt bilgi değeridir.
5. Sayfa alt bilgisi (Page Footer): Sayfa alt bilgisi değeridir.

Ayrıntılar bölümü dışındaki diğer bölümlerde gösterilecek veri Text Object nesnesi eklenerek yazılabileceği gibi veritabanı uzmanı (Database Expert) yardımı ile belirlenen alanlar ile de doldurulabilir. Ayrıntılar bölümünde ise hedeflenen veri alanları belirlenerek rapor üzerine konumlandırılır.

Etkileşimli CR raporları hazırlanırken grafik ara yüzü kullanmak, bazı durumlarda yetersiz kalır. Bu gibi durumlarda raporlanacak veri kümesi, rapor belgesi gibi CR uygulamasının ihtiyaç duyduğu bilgiler, kaynak kodlar yazılarak oluşturulur. Raporda gösterilecek verilerin veritabanından alınıp, rapora aktarılması için bir veri adaptörüne ihtiyaç duyulur. Bu amaçla OleDb kütüphanesinde yer alan OleDbDataAdapter kullanılabilir. Bu adaptör vasıtasıyla veri tabanından elde edilen veri kümesi rapor göstericinin veri kaynağı yapılarak istenen özelliklerde etkileşimli raporlar hazırlanabilir.

```
// Veri adaptörünü parametre değerleri ile oluştur
OleDbDataAdapter^ veri_adap = gcnew OleDbDataAdapter(”SELECT * FROM
                              tablo1”, vt_baglan);
// Veri kümesi oluştur
DataSet veri_kume = gcnew DataSet();
// Veri kümesini doldur
veri_adap->Fill(veri_kume,”sonuc”);
// Rapor belgesi oluştur
ReportDocument ^rapor = gcnew ReportDocument();
// Rapor belgesini yükle
rapor->Load("rapor.rpt");
// Veri kümesini rapora yükle
Rapor->Database->Tables[0]->SetDataSource(vei_kume->Tables[0]);
// Rapor göstericiyi oluştur
CrystalReportViewer rapor_goster = gcnew CrystalReportViewer();
// Rapor göstericiye rapor belgesini aktar
Rapor_goster->ReportSource = RepDoc;
```

BÖLÜM 4

SİMÜLASYON VERİSİNİ OLUŞTURMA

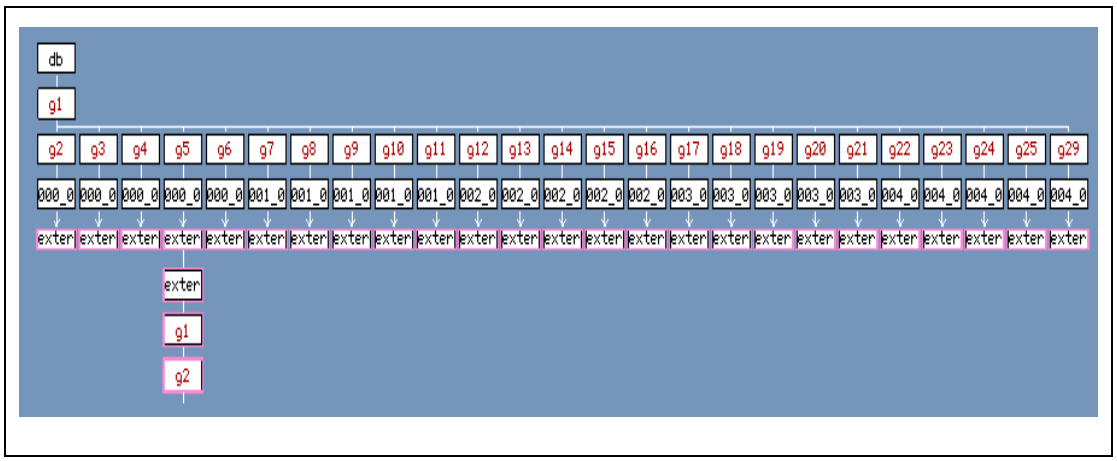
Trafik simülasyon sisteminin uygulanacağı ortamda yer alan görsel elemanlar bina, araç, yol, trafik işareti, trafik ışığı vb. olarak sıralanabilir. Ortamın görsel olmasının yanında işlenebilir bir özelliğe sahip olması da gereklidir. Özellikle trafik akışını sağlayacak hareketlilerin gerçeğe yakın bir biçimde simüle edilmeleri için ortam görsel elemanlarının verilerine herhangi bir enterpolasyona gerek duyulmadan erişilebilinmelidir. CBS verileri kullanılarak oluşturulan verilerin dezavantajı düşük yoğunluktaki haritalardan elde edilen verilerin ayrıntılandırılabilmesi için belli oranlarda enterpolasyona uğratılması gerekliliğidir. 3B görsel modellerin kullanılması bu dezavantajı giderir.

Trafik simülasyon sisteminde trafik akışı otonom çalışan taşıt hareketlerinden elde edilecektir. Otonom çalışan taşıtlar etmen tabanlı olarak modellenebilir. Sistemde kullanılan etmenler, konumlarına göre içinde bulunduğu ortamı algılayan ve eylemcileri yardımıyla ortamı ve ileride ortama yönelik algılarını etkileyen eylemde bulunan program parçaları olarak tanımlanabilir [31, 32].

Otonom olarak hareket eden etmenlerin, şehir üzerinde yol bilgisini bilerek hareket etmesi zorunludur. Çünkü şehir üzerinde bina, yeşil alan, dere, kara yolu gibi birçok öğe vardır. Etmenlerin hareket kabiliyetlerini sadece yol üzerinde gerçekleştirmeleri beklenir. Otonom taşıtlar davranış özelliklerine göre trafikte ilerlerken karşılaştıkları trafik kurallarına da uymak durumundadırlar.

4.1. YOL BİLGİSİ VERİ KAYNAĞININ OLUŞTURULMASI

Simülasyon sisteminde etmen taşıtların hareketlerinin yönlendirilmesi için gerekli ortam bilgisi 3B Open Flight (FLT) modelinde örnek bir uygulamadan temin edilmiştir. Şekil 4.1'de bu örnek uygulamanın graf görünümü gösterilmiştir. Presagis Creator yazılımının çıktısı olan FLT modeli, simülasyon sistemlerine ortam hazırlamak maksadıyla geliştirilen üç boyutlu görsel bir modeldir. Bu modelde oluşturulan grafik nesneleri, bir grafın düğümleri gibi birbirlerine bağlıdırlar. Bu yapıları sayesinde FLT modeller OpenGL, DirectX gibi grafik kütüphaneleri tarafından işlenebilirler [37].

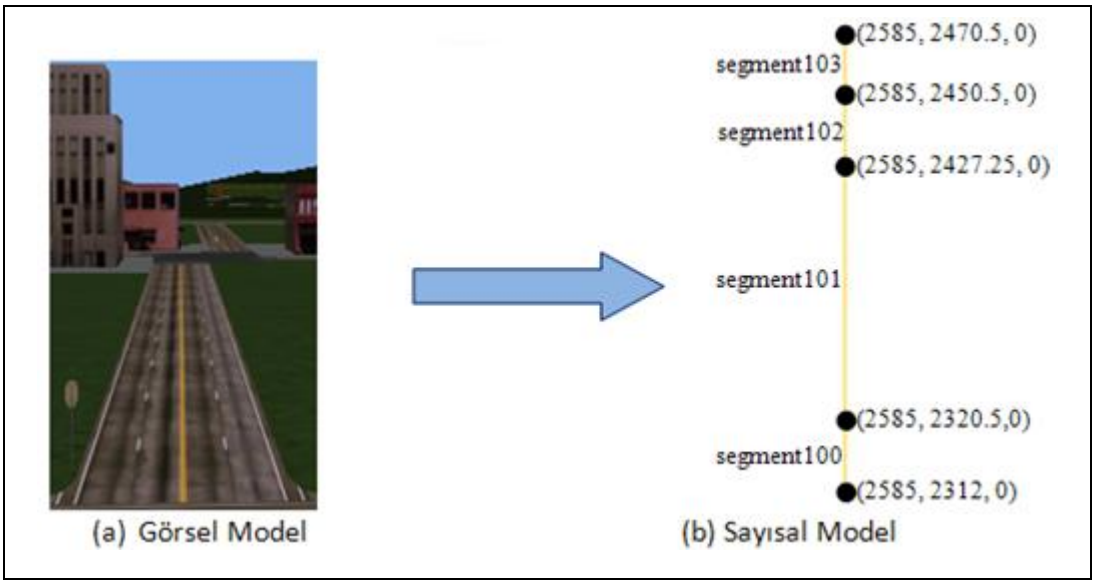


Şekil 4.1. "Town.flt" dosyasının graf görünümü.

Görsel model, etmen hareketlerinin organize edilmesi için gerekli yol, şerit, trafik işaretleri, trafik ışıkları gibi sayısal verilere sahiptir. Fakat bu verilerin yazılımda kullanılabilmesi için uygunlaştırılmaya ihtiyaçları vardır. Bu amaçla, 3B şehir modeli dosyası Creator yazılımı ile analiz edilerek şehir üzerindeki yol bilgileri kurallı bir biçimde bir XML dosyasında yapılandırılmıştır. Gerçekte, şehir modelinde yer alan bütün yolların sayısal bir haritası çıkarılarak yol bilgisi dosyasına kaydedilmiştir.

Yol bilgisi dosyasında yollar segmentlerden oluşmaktadır ve kavşaklar ile birbirlerine bağlanmaktadırlar. Sayısal yol haritası üzerinde segment bilgileri, segment-segment ve segment-kavşak bağlantı noktaları ve kavşak bilgileri tutulmaktadır. Bir yol, uzunluğuna göre farklı sayıda segmentten oluşmaktadır.

Sayısal harita üzerinde her bir segmentin başlangıç ve bitiş noktaları, genişlikleri, yükseklikleri, yol üzerindeki şerit sınır değerleri, kavşak bilgileri, bağlantı noktaları gibi bilgiler bulunmaktadır. Şekil 4.2'de roads.xml dosyası oluşturulurken YOL10 isimli yolun, görsel modelden sayısal modele geçiş işleminin bir bölümü yer almaktadır. Sayısal model üzerinde, yola ait dört segmentin başlangıç ve bitiş koordinat noktaları yer almaktadır.

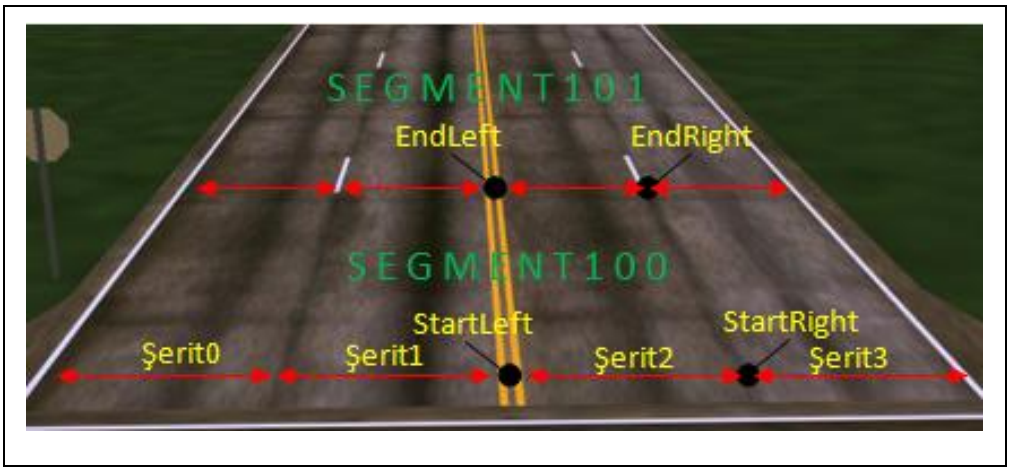


Şekil 4.2. roads.xml dosyasının üretilmesi.

Simülasyon sisteminde yer alan etmen taşıtlar şerit takibi, trafik kurallarına uyum, kavşak geçişleri gibi davranışları otonom bir biçimde gerçekleştirirler. Etmenlerin bu davranışları yürütebilmeleri için bulundukları konuma hâkim olmaları gerekir. Örneğin şerit takibi davranışı için etmen bulunduğu yolun şeridinde, diğer şeritleri ihlal etmeden ve yoldan çıkmadan hareket etmelidir. Bunun için bulunduğu şeridin başlangıç ve bitiş noktaları dışına çıkmaması gereklidir. Yol bilgisi dosyasında bu amaçla tutulan şerit sınır değerlerine ait parametreler Şekil 4.3'de gösterilmiştir. Bu parametre değerleri belirlenirken aşağıdaki algoritma yürütülmüştür.

1. Bulunulan segmentin orta noktasının başlangıç ve bitiş koordinatlarını çıkar (StartPoint, EndPoint).

2. Şeritlerin, segment başlangıç noktasındaki, sağdan ve soldan sınır değerlerinin, segment başlangıç noktasına olan uzaklıklarını hesapla (StartRight, StartLeft, StartUp, StartDown).
3. Şeritlerin, segment bitiş noktasındaki, sağdan ve soldan sınır değerlerinin, segment bitiş noktasına olan uzaklıklarını hesapla (EndRight, EndLeft, EndUp, EndDown).



Şekil 4.3. Şerit Sınır parametreleri.

Örneğin SEGMENT100'ün, Şerit2'sinde ilerleyen bir etmen şerit takibi davranışını, yukarıda belirtilen algoritmadaki sınır değerleri içinde kalarak gerçekleştirir. Otonom hareket eden etmen taşıtın şerit takibinde olduğu gibi diğer davranışlarını da gerçekleştirebilmesi için, bulunduğu segmentin eksen bilgisi (axis), kavşağa yakınlık durumu (TL), düz veya eğri olması (curve) ve eksen geometrisi (type); bulunduğu şeridin adı (LID), yönü (Direction), kullanabilirliği (valid) ve şeride girebilecek taşıt tipleri (VehicleType) gibi parametrelerin ilgili segment özelliklerinde bulunması gerekir. Şekil 4.4'de roads.xml dosyasında bulunan örnek bir segment verisi gösterilmiştir.

```
<Segment sName="segment100" sid="100" axis="1" TL="3" curve="0" type="1">
      <StartPoint SX="2585.00" SY="2315.50" SZ="0.000"/>
      <EndPoint EX="2585.00" EY="2320.50" EZ="0.000"/>
      <Widths SRight="7.5" SLeft="7.5" ERight="7.5" ELeft="7.5" />
      <Heights SUp="0.00" SDown="0.00" EUp="0.00" EDown="0.00" />
      <Lanes>
        <Lane LID="0" Valid="1" Direction="0" VehicleType="1" StartRight="-
3.75" StartLeft="-7.5" EndRight="-3.75" EndLeft="-7.5" StartUp="0.00"
StartDown="0.00" EndUp="0.00" EndDown="0.00"></Lane>
        <Lane LID="1" Valid="1" Direction="0" VehicleType="1"
StartRight="0.00" StartLeft="-3.75" EndRight="0.00" EndLeft="-3.75"
StartUp="0.00" StartDown="0.00" EndUp="0.00" EndDown="0.00"></Lane>
        <Lane LID="2" Valid="1" Direction="1" VehicleType="1"
StartRight="3.75" StartLeft="0.00" EndRight="3.75" EndLeft="0.00"
StartUp="0.00" StartDown="0.00" EndUp="0.00" EndDown="0.00"></Lane>
        <Lane LID="3" Valid="1" Direction="1" VehicleType="1"
StartRight="7.50" StartLeft="3.75" EndRight="7.50" EndLeft="3.75"
StartUp="0.00" StartDown="0.00" EndUp="0.00" EndDown="0.00"></Lane>
      </Lanes>
    </Segment>
```

Şekil 4.4. Örnek bir yol segment verisi.

Yol bilgisi dosyasında segment bilgilerinin yanı sıra segmentlerin bağlantı bilgileri de tutulmaktadır. Etmen taşıt ilerlerken segment değiştirdiğinde ulaştığı yeni yer ya bir segmenttir, ya da bir kavşaktır. Bu durumda etmen taşıtın, yeni konumuna göre davranışlarını belirlemesi gerekir. Bu sebeple bir segmente ait şeritlerin bütün bağlantı alternatiflerinin yol bilgisi dosyasına eklenmesi gerekir. Şekil 4.5'de yol bilgisi dosyasında yer alan segment-segment (a) ve segment-kavşak (b) bağlantı verisi gösterilmiştir.

```
<SegmentConnection BeginOf="segment100" EndOf="segment101">
      <LaneConnection FromSegment="segment101" FromLane="0"
ToSegment="segment100" ToLane="0"></LaneConnection>
      <LaneConnection FromSegment="segment101" FromLane="1"
ToSegment="segment100" ToLane="1"></LaneConnection>
      <LaneConnection FromSegment="segment100" FromLane="2"
ToSegment="segment101" ToLane="2"></LaneConnection>
      <LaneConnection FromSegment="segment100" FromLane="3"
ToSegment="segment101" ToLane="3"></LaneConnection>
    </SegmentConnection>
```

(a)

```
<Intersection>
    <Segment Name="segment103" />
    <Segment Name="segment110" />
    <Segment Name="segment083" />
    <Segment Name="segment090" />
    <IntConnection FromSegment="segment103" FromLane="2"
ToSegment="segment090" ToLane="1"></IntConnection>
    <IntConnection FromSegment="segment103" FromLane="3"
ToSegment="segment090" ToLane="0"></IntConnection>
    <IntConnection FromSegment="segment103" FromLane="2"
ToSegment="segment110" ToLane="2"></IntConnection>
    <IntConnection FromSegment="segment103" FromLane="3"
ToSegment="segment110" ToLane="2"></IntConnection>
    <IntConnection FromSegment="segment110" FromLane="0"
ToSegment="segment083" ToLane="3"></IntConnection>
    <IntConnection FromSegment="segment110" FromLane="1"
ToSegment="segment083" ToLane="2"></IntConnection>
    <IntConnection FromSegment="segment110" FromLane="0"
ToSegment="segment103" ToLane="1"></IntConnection>
    <IntConnection FromSegment="segment110" FromLane="1"
ToSegment="segment103" ToLane="1"></IntConnection>
    <IntConnection FromSegment="segment083" FromLane="0"
ToSegment="segment103" ToLane="0"></IntConnection>
    <IntConnection FromSegment="segment083" FromLane="1"
ToSegment="segment103" ToLane="1"></IntConnection>
    <IntConnection FromSegment="segment083" FromLane="0"
ToSegment="segment090" ToLane="1"></IntConnection>
    <IntConnection FromSegment="segment083" FromLane="1"
ToSegment="segment090" ToLane="1"></IntConnection>
    <IntConnection FromSegment="segment090" FromLane="3"
ToSegment="segment110" ToLane="3"></IntConnection>
    <IntConnection FromSegment="segment090" FromLane="2"
ToSegment="segment110" ToLane="2"></IntConnection>
    <IntConnection FromSegment="segment090" FromLane="3"
ToSegment="segment083" ToLane="2"></IntConnection>
    <IntConnection FromSegment="segment090" FromLane="2"
ToSegment="segment083" ToLane="2"></IntConnection>
  </Intersection>
```

(b)

Şekil 4.5. Segment ve kavşak bağlantı verileri.

4.2. KATI CİSİMLERİN MODELLENMESİ

3B cisimler modellenirken tel kafes modelleme, yüzey modelleme ve katı modelleme teknikleri kullanılmaktadır. Tel kafes modeli, köşeler ve köşeleri birbirine bağlayan kenarlardan oluşur. Cismin tel kafes modelle ifadesi nokta, doğru, yay, çember gibi 2B'lu çizim elemanları ile gerçekleştirilir. Bu modelde tasarlanan bir cismin sadece dış yüzeyi modellenirken iç kısmı modellenmemektedir. Bu sebeple cisimlerin hacimleri yoktur.

Bir yüzey model, köşe, kenar ve yüzlerle tanımlanır. Bu teknikte nesneler, sınır yüzeyleri ile tanımlanmaktadır. Yüzey modellemenin tel kafes modelden farkı, nesne temsilinde yüzey birleştirilmelerinin yer almasıdır. Birleştirme için poligonlar kullanılır. Yüzey modelleme, arazi yüzeylerinin modellenmesinde tercih edilir. Yüzey modellemede, tel kafeste olduğu gibi, cismin iç kısmı modellenmediğinden, cisim bir hacim içermemektedir.

Katı modelleme, tel kafes ve yüzey modellemenin eksik yanlarını tamamlamak için geliştirilmiştir. Bu model, cismin farklı 3B parçaları ayrı ayrı tasarlanıp, birleştirilmesi ile oluşturulur. Yüzeyler oluşturulurken tel kafes ve yüzey modelleme yöntemlerinden de faydalanılır. Oluşturulan katı modele ait geometrik ve topolojik bilgi kullanılarak, cismin kütle özellikleri hesabı, katı cisim dinamiklerinin modellenmesi işlemleri yapılabilir.

Katı modelleme için sınır temsili (Boundary Representation-BRep) ve yapısal katı geometrisi (Constructive Solid Geometry – CSG) gibi temsil yöntemleri kullanılmaktadır [45].

Sınır temsil tekniğinde, cisimler köşe, kenar ve yüzey sınırları ile temsil edilirler. Sınır temsili yüzey modelleme tekniklerine dayandırılmaktadır. Katı bir nesnenin sınır temsili, nesnenin iç ve dış taraflarını tanımlayan tüm yüzeylerin tanımlanması ile oluşturulur. Sınır temsili tanımlamaları, topolojik ve geometrik tanımlamaları içermektedir. Geometrik tanımlamalar, nokta, eğri yüzey gibi elemanların geometrik

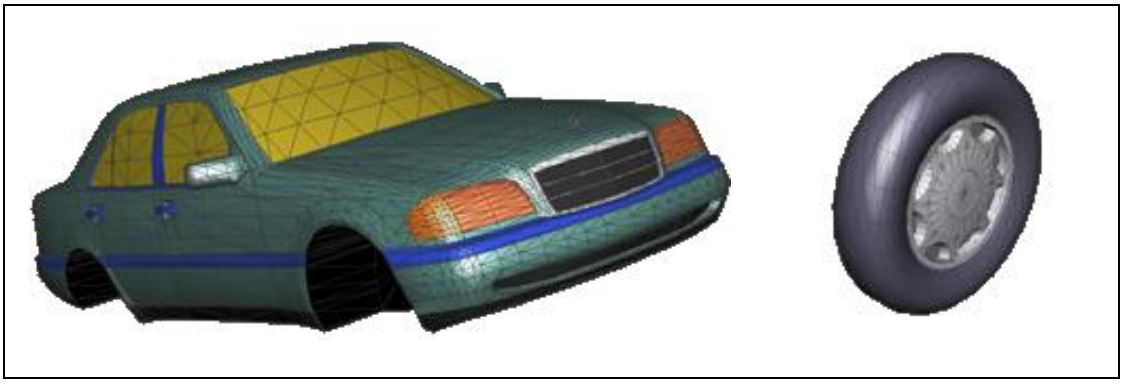
olarak tanımlanmasıdır. Topolojik tanımlamalar ise köşe noktaları, kenar ve yüzeylerin birbiri ile bağlantı durumlarını kapsamaktadır.

Yapısal katı geometrisi (YKG) temsil sisteminde cisimler, temel elemanlar olarak adlandırılan basit elemanların (küp, silindir, koni vb.), mantıksal operatörlerle (ekleme, çıkarma ve arakesit alma) işlenmesi sonucu elde edilirler. Temel elemanlar parametrik denklemlerle tanımlanmıştır. YKG ile temsil edilen cisim, bir ağaç veri yapısında saklanır. YKG ağacında temel elemanlar ağaç dallarında yer alırken, mantıksal operatörler düğümlerde tutulurlar.

4.3. TAŞITLARIN MODELLENMESİ

Simülasyon sisteminde kullanılacak taşıt bilgileri 3B model dosyalarından alınacaktır. Bu amaçla Blender, AC3D ve 3dMax yazılımlarından faydalanılmıştır. 3B dosyalar, dosya başlığı, materyal listesi ve alt model listesi verilerini içerirler. Alt model listesi bütün nesneleri kapsar ve bu listede nesnelerin dönüşüm matrisleri, köşe koordinatları, üçgen yüzleri, normal vektörleri, materyal listeleri ve desen koordinatları yer alır [37].

TSS'de kullanılan taşıtlar, katı modelleme tekniğinde, YKG temsil sistemi ile oluşturulmuştur. Taşıtların gövde ve tekerlek modelleri ayrı ayrı hazırlanmıştır. Farklı taşıt tiplerinde kullanılan gövde ve tekerlek modelleri de değişkendir.



Şekil 4.6. Örnek bir taşıta ait gövde ve tekerin 3B modeli.

4.4. TRAFİK KURALLARI

Aday sürücü, simülasyon esnasında taşıt sürme faaliyetini gerçekleştirirken, gerçek trafikte olduğu gibi uyması gereken kurallar vardır. Bu kurallar, görsel ve görsel olmayan trafik kuralları olmak üzere sınıflandırılabilir. Görsel trafik kuralları, trafik işaretleri ve trafik ışıkları gibi sürücüye uyarı şeklinde önceden bildirilen yaptırımları içerirler. Görsel olmayan trafik kuralları ise herhangi bir işaretçi olmaksızın sürücünün bilmesi ve uyması gereken kurallardır. Şehir içi ve şehirlerarası hız limitleri, şerit değiştirirken veya dönerken sinyal verme davranışları ve çift yönlü yollarda sağda yer alan yolu kullanma gibi kurallar, görsel olmayan trafik kuralları arasında sayılabilirler.

4.4.1. Trafik Kurallarının 3B modellenmesi

Karayolundan yararlananlara, yol, trafik durumu ve yakın çevre ile ilgili gerekli bilgileri vermek, yasaklama ve kısıtlamaları bildirmek suretiyle trafik düzen ve güvenliğini sağlamak amacıyla trafik işaretlerinden faydalanılır. Trafik işaretleri sürücü adaylarınca bilinmesi ve uygulanması zorunlu sembollerdir.

Sürücü adaylarının öğrenmesi gereken önemli bir trafik kuralı da trafik ışıklarıdır. Dünya üzerinde yaklaşık herkes trafik ışıklarının renklerinin anlamlarını bilir, fakat trafik ışıklarına göre davranmayan sürücülerin sayısı azımsanamayacak bir değerdedir.

Bu amaçla TSS sisteminde, [46]'ya uygun olarak karayolu üzerine trafik ışıkları ve trafik işaretleri yerleştirilmiştir. Söz konusu işaretler, Multigen Creator 3.0 yazılımı kullanılarak oluşturulmuştur.

4.4.1.1. Trafik işaretlerinin modellenmesi

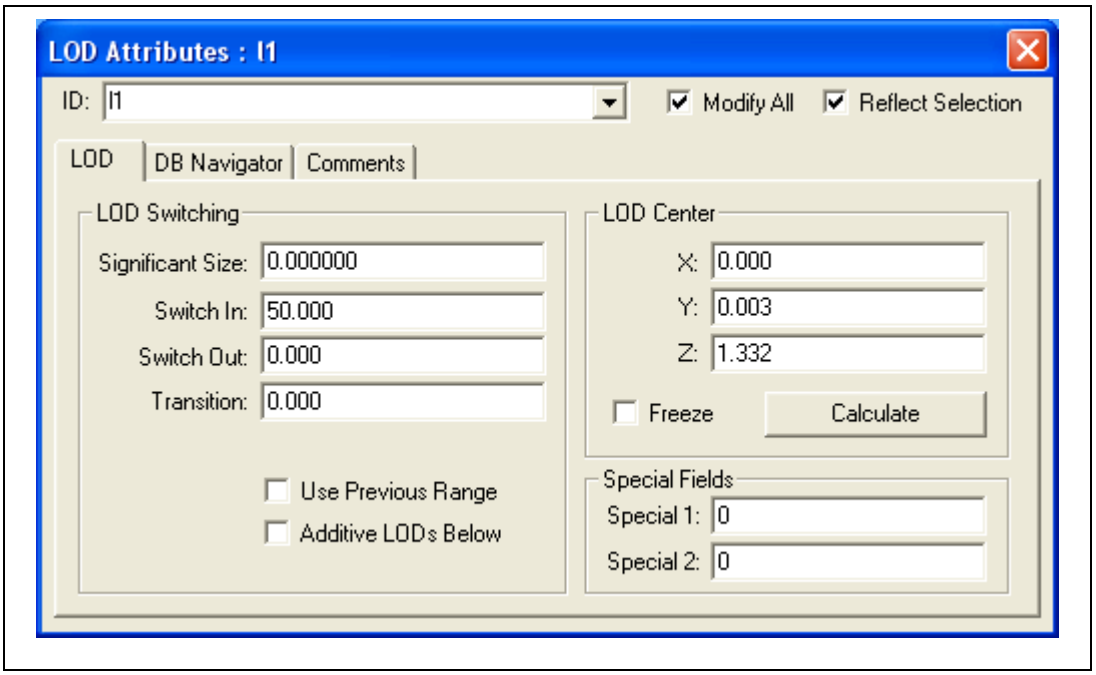
İnsanın bir nesneye bakış mesafesi, nesnenin insan beynindeki algısında farklılıklara sebep olur. Zira bir nesneye yakından bakıldığında daha ayrıntılı özellikleri

görülebilirken, mesafe uzadıkça nesnenin görünür ayrıntıları azalmakta ve belli bir mesafeden sonra nesne artık görünmemektedir. Bu durumun bilgisayar grafiklerinde ifadelendirilmesi için nesneler LoD düğümler kullanılarak modellenmektedir. LoD düğüm kullanılarak oluşturulan modelde, aynı nesne için farklı mesafelerden, farklı görünümler tasarlanabilmektedir. Bu amaçla trafik işaretleri de farklı ayrıntı düzeyinde görüntülenebilmeleri için LoD düğümler kullanılarak modellenmiştir.

Şekil 4.7. Bir trafik işaretinin farklı ayrıntı düzeylerindeki görüntüleri ve ağaç yapısı.

Şekil 4.7'de 30 km hız sınırlamasını gösteren trafik işaretinin farklı ayrıntı düzeyindeki görüntüleri ve grafları gösterilmiştir. I1, I2, I3 düğümleri sp_lim_ nesnesinin LoD düğümleridirler. I1 düğümü nesnenin en ayrıntılı görüntüsünü, I2 düğümü nesnenin daha az ayrıntılı görüntüsünü, I3 düğümü ise nesnenin en az

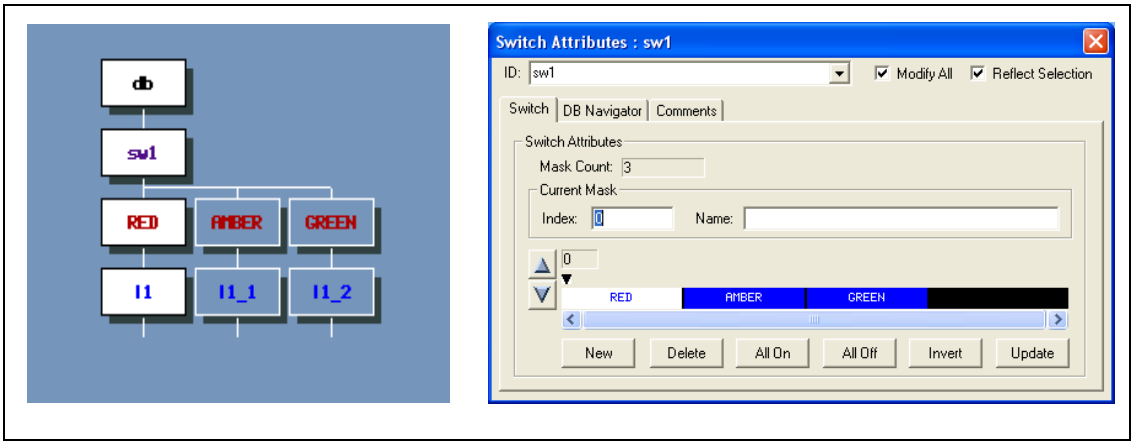
ayrıntılı görüntüsünü içermektedir. Simülasyon esnasında hangi mesafelerde nesnenin hangi görüntüsünün görüntüleneceği nesnenin LoD özellikleri penceresinden verilir (Şekil 4.8.). Bu özellikler kullanılarak istenirse belli bir mesafeden sonra nesnenin görünmemesi de sağlanabilir.



Şekil 4.8. LOD özellikleri penceresi.

4.4.1.2. Trafik Işıklarının Modellenmesi

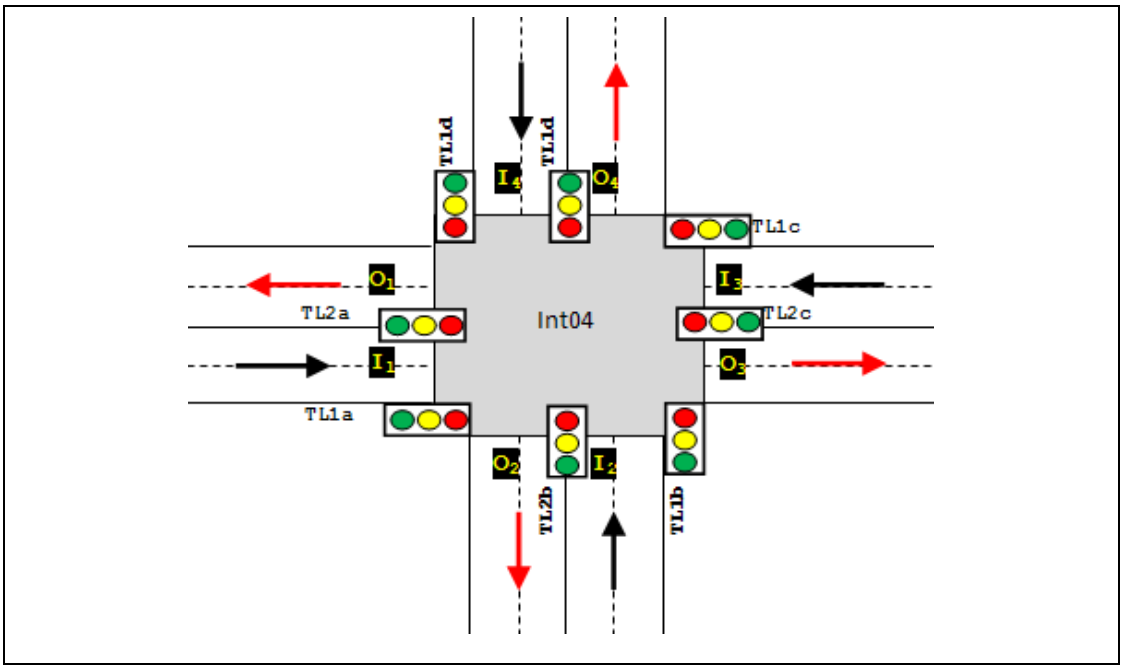
Trafik ışıkları modellenirken LOD düğümlerin yanında, anahtar (switch) düğüm de kullanılmıştır. Bir anahtar düğüm, bir nesneye ait farklı görüntülere sahip olabilir. Trafik ışıkları da belirli bir senkronizasyon içerisinde durum (kırmızı, sarı, yeşil ışıklar) değiştirirler. Bu durumları modellemek için anahtar düğüm kullanılmıştır. Şekil 4.9'da TSS'de kullanılan bir trafik ışığının graf modeli ve anahtar özellikleri penceresi gösterilmiştir. Bu modelde sw1 anahtar düğümü; RED, AMBER, GREEN olmak üzere üç ayrı model içerir. Şekil 4.9'daki durumda trafik lambası RED durumu aktif, diğer durumlar ise pasiftir ve dolayısıyla trafik ışığı kırmızı yanar. Trafik ışıklarının durumları, (örneğin kırmızı yanması, kırmızı-sarı yanması, yeşil yanması v.b. gibi) anahtar özellikleri penceresinden eklenir.



Şekil 4.9. sw1 anahtar düğümünün graf modeli ve anahtar özellikleri penceresi.

4.4.1.2.1. Çalışma frekanslarının ayarlanması

TSS'de trafik ışıklarının kontrol işlemi iki basamaktan oluşmaktadır. Birinci basamak, trafik ışıklarının senkronize bir şekilde durum değiştirme işlemlerini kapsar. İkinci basamak ise trafik ışıklarının durumlarına göre etmen davranışlarının düzenlenmesi faaliyetleridir.



Şekil 4.10. Int04 kavşağının trafik ışık planlaması.

Sistemde trafik ışıkları, bir zamanlayıcı fonksiyonu kullanılarak belirlenen sürelerde ilgili trafik ışığı üzerinde görüntülenmesi gereken durum bulunarak senkronize bir şekilde çalıştırılmaktadır. Şekil 4.10'da gösterilen Int04 isimli kavşağın zamana göre ve periyodik süren durum haritası Çizelge 4.1'de gösterilmiştir. Her bir köşede yer alan trafik lambalarının biri karşıya geçişi ve sağa dönüşü kontrol ederken, diğeri sola dönüşü kontrol etmektedir. Çizelge 4.1'deki kullanıcı tanımlı UD1 ve UD2 ile isimlendirilmiş sütunun zaman değeri, TSS ayarlar ara yüzünde belirtilen süre değerlerinden alınmaktadır. Bu süreler ana yollardaki trafik akışını rahatlatmak veya trafik yoğunluğu artırmak amacıyla kullanılabilir.

Çizelge 4.1. Trafik ışıklarının sinyal frekans aralığı tablosu.

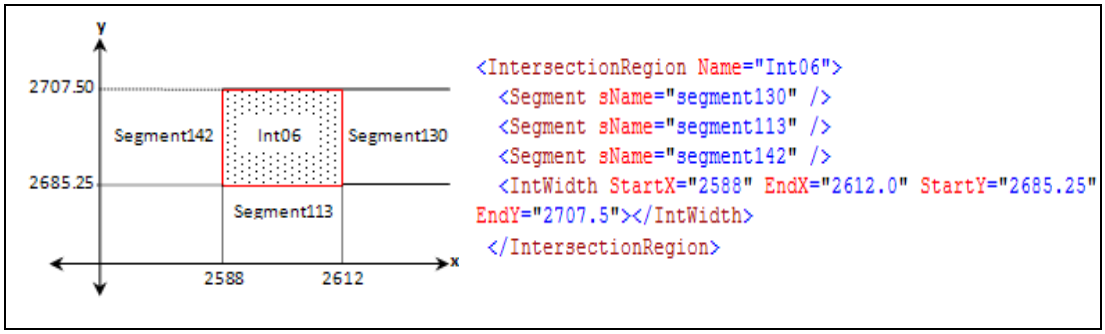
Trafik Işıkları	Zaman (s)							
	5	UD1	5	5	5	UD2	5	5
TL1a								
TL1b								
TL1c								
TL1d								
TL2a								
TL2b								
TL2c								
TL2d								

4.4.2. Trafik Kuralları Veri Kaynağının Oluşturulması

Simülasyon sisteminde ilerleyen bir aracın hangi kurallara uymakla sorumlu tutulduğu bulunduğu konuma bağlıdır. Araç nerede bulunuyorsa orada yer alan trafik işaretlerinden ve trafik kurallarından sorumludur. TSS'de kullanılan şehir, üç boyutlu (3B) Open Flight (flt) modelindedir [37]. Şehirde yer alan bütün karayolları analiz edilerek, aracın konumuna göre yapılacak trafik kontrolünü gerçekleştirmeye yarayan veri kümeleri roads, intersections ve trafficrules isimlerindeki XML dosyalarına kaydedilmiştir. roads.xml dosyasında şehir üzerinde yer alan yollara ait parametrik değerler tutulmaktadır.

Şehirde yer alan yollar kavşaklar ile birbirlerine bağlıdırlar. Her bir yolun son segmenti, farklı bir yolun segmentine bir kavşak ile bağlıdır. Sistemde ilerleyen bir

taşıtın konum bilgisi eğer yolda ise roads.xml dosyasından alınırken, kavşakta ise kavşak veri kaynağı intersections.xml dosyasından temin edilmektedir. Kavşak veri kaynağı dosyası, kavşak ile bağlantısı olan yol segmentleri bilgisini ve kavşak sınır değerlerini saklar. Şekil 4.11'de Int06 kavşağının koordinat eksenindeki görüntüsü ve intersections.xml dosyasına aktarılmış hali gösterilmiştir.



Şekil 4.11. flt modeldeki kavşak verisinin xml kodlarına dönüştürülmesi.

Simülasyon esnasında trafikte yer alan taşıtların, gerçek trafikte olduğu gibi trafik kurallarına uymaları gerekir. Bu kurallar, görsel ve görsel olmayan trafik kuralları olmak üzere sınıflandırılabilir.

Görsel trafik kuralları, trafik işaretleri ve trafik ışıkları gibi sürücüye uyarı şeklinde önceden bildirilen yaptırımları içerirler. Görsel olmayan trafik kuralları ise herhangi bir işaretçi olmaksızın sürücünün bilmesi ve uyması gereken kurallardır. Örneğin sürücü, şehir içi yollarda azami hız sınırının 50 km/sa olduğunu bilmeli ve bu tip yollarda bu hız sınırını aşmamalıdır.

TSS'de aday sürücülerin trafik kurallarını uyum durumları bu iki türden kuralları içeren bir kural kümesi kullanılarak kontrol edilirler. Her bir segmentte yer alan trafik işaretleri, trafik ışıkları gibi görsel trafik kuralları ile görsel olmayan trafik kuralları bilgisi trafficrules.xml dosyasına işlenmiştir. Sürücü davranışları bu kural kümesi kullanılarak kontrol edilirler. Çizelge 4.2'de bu kural dosyasında kullanılan elemanlar yer almaktadır.

Çizelge 4.2. Trafik işaretlerini yapılandıran trafficrules.xml dosyası.

Eleman	Kök	Değer	Trafik işareti/Açıklama	Tanımlama	
Segment	Rule		Trafik kuralları		
sName	Segment	"Segment100"	Segment adı		
turn	Segment	"3"	Sola Dönülmez 	Değer	Açıklama
				0	Serbest
				1	Sola dönülmez, U dönüşü yapılmaz
				2	Sağa dönülmez, U dönüşü yapılmaz
				3	Sola dönülmez
				4	Sağa dönülmez
				5	U dönüşü yapılmaz
TRL	Segment	"1"	Trafik ışığı var.	Değer	Açıklama
				0	Trafik ışığı yok
				1	Trafik ışığı var
TLNumber	TRL	"0"	Trafik ışığı numarası "0"	Sistemde 8 trafik ışığı vardır.	
TLDirection	TRL	"1"	Yön "1" deki taşıtları ilgilendirir.		
velocity	Segment	"1"	30 km/sa hız sınırlaması 	Değer	Açıklama
				0	Serbest
				1	30 km/sa hız sınırlaması
				2	50 km/sa hız sınırlaması
				3	70 km/sa hız sınırlaması
				4	90 km/sa hız sınırlaması
				5	120 km/sa hız sınırlaması
parking	Segment	"1"	Park edilmez	Değer	Açıklama
				0	Serbest
				1	Park edilmez
				2	Duraklama yapılmaz

BÖLÜM 5

TRAFİK SİMÜLASYON SİSTEMİ YAZILIMI

TSS, sürücü adaylarına sürücü eğitimi vermek için geliştirilen kural tabanlı bir uygulama yazılımıdır. Sistem sürücü adaylarına 3B bir simülasyon ortamında, direksiyon ve pedal donanımlarını kullanarak, gerçeğe yakın görüş açıları ile taşıt sürme olanağı sağlamak amacıyla hazırlanmıştır. Sürücü adayları oryantasyon, eğitim ve test olmak üzere üç farklı sürüş modunda taşıt sürebilirler.

TSS simülasyon ve raporlar olmak üzere iki gruptan oluşan bir ara yüz kullanılarak başlatılabilir. Simülasyon ara yüzü ile hava koşulları, simülasyon özellikleri, taşıt tipleri, vekil taşıt sayısı vb. parametrik değerler ayarlanabilir. Kullanıcıların yaptıkları test sürüşlerine ait sonuçlar ise raporlar ara yüzü ile irdelenebilmektedir.

TSS yazılımının kaynak kodları simülasyon, yapay zekâ ve raporlar olmak üzere üç bölümden oluşmaktadır. TSS, Microsoft Visual Studio (MSVS) 2008 yazılımı kullanılarak, C++ dili ile yazılmıştır.

Simülasyon bölümü, Delta3d açık kaynak kodlu oyun ve simülasyon kütüphanesi kullanılarak hazırlanmıştır. Bu bölüm grafik modülü, fizik modülü, mesaj sistemi ve çevre aygıtları modülü yordamlarından oluşmaktadır. Çizelge 5.1'de simülasyon bölümünün modülleri ve açıklamaları gösterilmiştir.

Çizelge 5.1. Simülasyon modülleri.

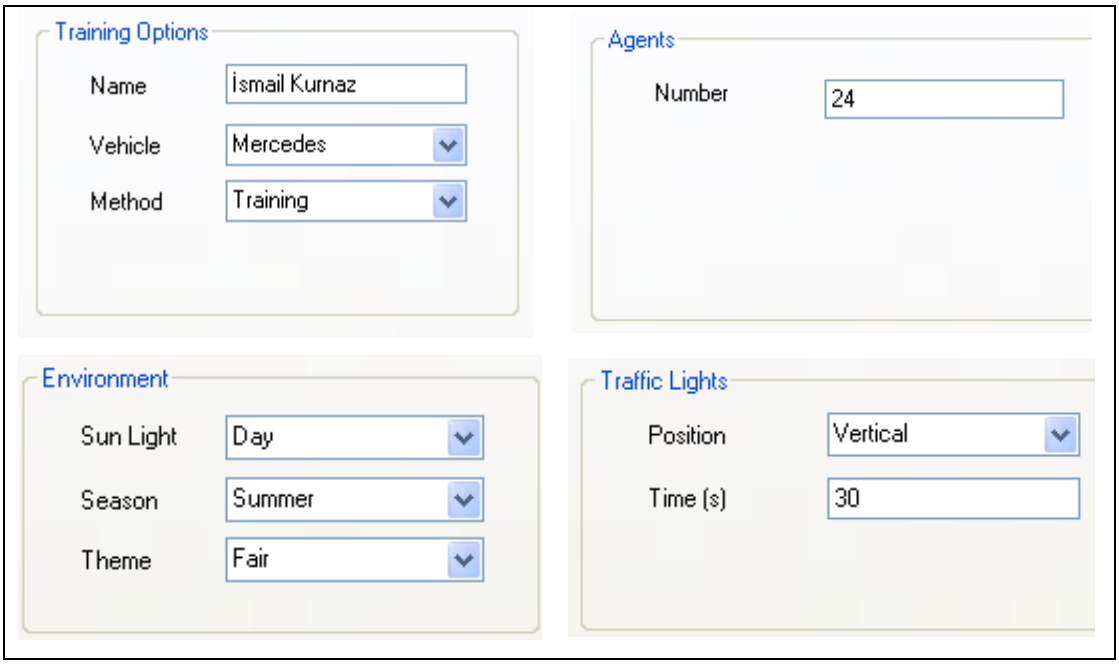
Modül	Kütüphane	Açıklama
Grafik İşlemleri	OSG	Simülasyon ortamının hazırlanması ve simülasyon süresince görüntülenmesi işlevlerini kapsar
Fizik işlemleri	ODE	Sistemde bulunan taşıtların katı cisim dinamiklerine uygun olarak oluşturulması ve hareket ettirilmesi işlevlerini içerir
Mesaj Sistemi	OpenAL	Sürücü eğitimini iyileştirmek için sesli mesaj uyarılarını çalıştırır.
Çevre aygıtları	PLIB	Giriş çıkış donanımlarının kontrol edilmesini organize eder.

Yapay zekâ bölümü, nesne tabanlı programlama kullanılarak yazılmıştır. Bu bölüm otonom hareket eden etmen taşıtların temel sürücü davranışlarını (şerit takibi, taşıt takibi, trafik kurallarına uyma vb.), aday sürücünün davranışlarını ve trafik akışını kontrol eden yordamlardan oluşmuştur.

Raporlar bölümünde aday sürücü eğitimi esnasında kaydedilen sürücü davranışlarının raporlanması gerçekleştirilir. Bu bölüm hazırlanırken OleDb ve CR kütüphaneleri kullanılmıştır.

5.1. KULLANICI ARA YÜZÜ

Kullanıcı ara yüzü simülasyon ve raporlar olmak üzere iki bölümden oluşmuştur. Simülasyon bölümünde, TSS yazılımının çalıştırılmasında kullanılacak parametrik değerler belirlenir. Parametreler girilirken eğitim seçenekleri (Training Options), vekiller (Agents), çevre koşulları (Environment) ve trafik ışıkları (Traffic Lights) alanlarındaki seçenekler kullanılır.



Şekil 5.1. TSS simülasyon ara yüzü.

Eğitim seçenekleri bölümünde kullanıcı adı (Name) girilerek, tercih edilen taşıt tipi (Vehicle) ve sürüş modu seçenekleri ayarlanmaktadır. Sürüş modu olarak oryantasyon (Experiment) veya test (Test) seçildiğinde görüntülenen zaman (Time) alanına simülasyon süresinin belirlenmesi gerekir. Bu değerler 1, 2, 3, 5, 10, 20 (dakika) arasından seçilir. Eğitim (Training) sürüş modunda süre ayarlaması yoktur. Vekiller (Agents) bölümünde ise simülasyonda bulunması istenen vekil sayısı belirlenir. Bu vekiller trafikte normal, agresif ve kural dışı davranışı sergileyecek biçimde üretilirler.

Simülasyon esnasındaki hava durumu, çevre koşulları bölümünde ayarlanır. Burada kullanılan seçenekler güneş ışığı (Sun Light), mevsim (Season) ve tema (Theme) alanlarında belirlenir. Güneş ışığı şafak vakti (dawn), gündüz (day), akşam karanlığı (dusk) ve gece (night) değerlerini; mevsim ilkbahar (spring), yaz (summer), sonbahar (fall) ve kış (winter) değerlerini; tema açık (clear), sisli (foggy) ve yağmurlu (rainy) değerlerini alabilir. Trafik ışıkları bölümünde yatay ve dikey ana yolları birleştiren kavşaklarda yer alan trafik ışıklarının geçiş süreleri (saniye olarak) belirlenmektedir.

5.2. SİMÜLASYON

Bu bölüm, geliştirilen trafik simülatörünün;

1. Şehir, taşıtlar, trafik işaretleri gibi grafiksel öğeleri,
2. Taşıt hareketleri, çarpışma testi gibi fiziksel öğeleri,
3. Sürücü eğitimine yardımcı olacak mesaj sistemi,
4. Kullanıcıya gerçeğe yakın taşıt sürüş imkânı sağlayacak oyun kumanda ve görüntü birimleri,

için hazırlanmış yordamları kapsamaktadır.

5.2.1. Sahne Grafının Hazırlanması

TSS yazılımında simülasyon süresince görüntülenen bütün öğeler (arazi, taşıtlar, trafik ışıkları, çevre koşulları) sahne grafı veri yapısında tutulmaktadır. Sahne grafının oluşturulma ve görüntüleme işlemleri için OSG kütüphanesi kullanılmıştır.

OSG'da sahne kök düğümüne öncelikle çevre koşulları özelliklerinin ayarlanabildiği dtABC::Weather nesnesi eklenir. Bu nesnenin güneş ışığı durumu ve mevsim özellikleri SetTimePeriodAndSeason() metodu kullanılarak, tema özelliği SetTheme () metodu kullanılarak ayarlanır. Bu metotlar ile atanacak özellikler TSS kullanıcı ara yüzünde seçilen çevre koşulları değerleridirler. Bu işlemden sonra AddDrawable() metodu ile oluşturulan hava koşulları nesnesi sahneye eklenir.

```
// mWheather nesnesini oluştur
dtCore::RefPtr<dtABC::Weather> mWeather = new dtABC::Weather();
assert(mWeather.valid());
// Özelliklerini ayarla
mWeather->SetTimePeriodAndSeason(mTimePeriod, mSeason);
mWeather->SetTheme(mWeatherTheme);
// mWeather nesnesini sahneye ekle
GetScene()->AddDrawable(mWeather->GetEnvironment());
```

Sistemde kullanılacak 3B verileri içeren arazi bilgisi dosyası FLT biçimindedir. OSG, FLT dosya biçimini tanımaktadır, fakat kullanılan dosyanın büyüklüğünden dolayı yüklenmesi ve çalıştırılması bellek kullanımına kısıtlar getireceğinden dolayı

FLT biçimindeki arazi dosyası, OSG'nin kendi dosya biçimi olan IVE biçimine çevrilmiştir. OSG kütüphanesinin DOS ortamında çalışan osgconv komutu ile bu işlem kolayca gerçekleştirilir.

```
C:\> osgconv town.flt town.ive
```

Sahne grafında gösterilecek bütün düğümler mWeather nesnesine çocuk düğüm olarak bağlanırlar. Bunun sebebi hava koşullarında yapılacak her bir değişikliğin diğer bütün nesneleri etkilemesidir. Sistem arazi bilgisi, dtCore::Object sınıfından üretilen bir nesne ile temsil edilebilir. Bu nesne oluşturulduktan sonra LoadFile() metodu kullanılarak nesneye kaynak 3B model dosyası yüklenir. mWeather düğümüne arazi nesnesi eklenerek, arazinin sahne grafına eklenmesi sağlanır.

```
// Arazi nesnesini oluştur
dtCore::RefPtr<dtCore::Object> mTerrain = new dtCore::Object();
assert(mTerrain.valid());
// Model dosyayı yükle
mTerrain->LoadFile("data/models/worldParts/terrain/town.ive");
// Arazi nesnesini sahneye ekle
mWeather->GetEnvironment()->AddChild(mTerrain.get());
```

Simülasyon sahnesinde yer alan taşıtlar, araziye benzer şekilde sahne grafına eklenirler. Taşıtın sahneye eklenmesindeki farklar çevre koşullarının taşıt modellerinin görüntüsünü etkilemeyeceğinden dolayı sahne kök düğümüne doğrudan eklenmeleri ve taşıtların fizik modülünde eklemleme yöntemi ile oluşturulacağından dolayı 3B dosya modellerinin fizik modülünde yüklenmesidir.

```
// Taşıt nesnesi oluştur
dtCore::RefPtr<dtCore::Object> mVehicle = new dtCore::Object();
assert(mVehicle.valid());
// Add objects to the scene
GetScene()->AddDrawable(mVehicle.get());
```

Sahne üzerinde yer alan trafik işaretleri arazi nesnesi üzerinde tümleşiktir, fakat trafik ışıkları yapay zeka modülü ile kontrol edileceğinden dolayı sahneye ayrı nesneler olarak eklenmiştir. Trafik ışıkları modellenirken önceden bahsedildiği gibi anahtar düğüm kullanılmıştır. Grafik modülünde bir trafik ışığı sahneye eklenirken osg::NodeVisitor sınıfında yer alan findNodeVisitor nesnesi kullanılır. Anahtar

düğüme bağlı, hangi yaprak düğüm gösterilecekse (yapay zeka modülü belirler), ilgili düğüm görüntüsündeki trafik ışığı sahneye eklenir.

```
// Taşıt nesnesi oluştur
dtCore::RefPtr<dtCore::Object> mTL = new dtCore::Object();
// Model dosyayı yükle
mTL->LoadFile("data/models/trafficlights/traflight.flt");
// sw1 anahtar düğümünü bul
osg::Node* SignalNode = mTL->GetOSGNode();
findNodeVisitor findTL("sw1");
SignalNode->accept(findTL);
// Trafik ışığını belirlenen durumda, sahneye ekle
SwitchTL[0]=dynamic_cast <osgSim::MultiSwitch*> (findTL.getFirst());
SwitchTL[0]->setSingleChildOn(0,2);
GetScene()->AddDrawable(mTL.get());
```

5.2.2. Taşıtın Oluşturulması

Sistemde bulunan taşıtların katı cisim dinamiklerine uygun olarak oluşturulması ve hareket ettirilmesi işlevleri, fizik modülünde kullanılan kaynak kodlar ile gerçekleştirilmektedir. 3B olarak modellenen taşıt gövde ve tekerleklerinin birleştirilmesi, katı cisim dinamiklerine göre hareket ettirilmesi ve çarpışma testi işlevleri gerçekleştirilirken ODE kütüphanesinden faydalanılmıştır.

Sistemde kullanılacak taşıtların hareketlendirilmesi için gerekli ilk işlem taşıt başlangıç değerlerinin atanmasıdır. Taşıtın gövde ve teker 3B modelleri, kütleleri, tekerlerin X, Y düzleminde taşıt merkezine uzaklıkları bir taşıt için gerekli sabit değerlerdir. Bu değerler taşıtın tipine göre farklıdırlar.

```
// Taşıt gövdesinin 3B modelini yükle
mVehicle[VEHICLE_BODY]->LoadFile(mVehicleBodyFileName);
// Taşıt gövdesinin kütlesini ayarla
dMassSetBox(&mMass, 1.0, dimension[0], dimension[1], dimension[2]);
dMassAdjust(&mMass, mVehicleBodyMass);
mVehicle[VEHICLE_BODY]->SetMass(&mMass);
// Tekerleğin 3B modelini yükle
mVehicle[wheelType]->LoadFile(mVehicleWheelFileName);
// Tekerleğin kütlesini ayarla
dMassSetCylinder(&mMass, 1.0, 3, dimension[0], dimension[1]);
dMassAdjust(&mMass, mVehicleWheelMass);
mVehicle[wheelType]->SetMass(&mMass);
```

Taşıt arazi üzerinde ilerlerken herhangi bir nesneye çarptığında, bu çarpışmanın taşıtı etkilemesi gerekir. Taşıtın nesneye çarpma durumu çarpışma testi işlemleri ile tespit edilir. Bu tespitin yapılması için taşıtın parçalarını içine alacak şekilde çarpışma geometrileri oluşturulur. Taşıtın gövdesi için kutu geometrisi kullanılırken tekerleri için silindir geometrisi tercih edilir. Çarpışma geometrisinin ihlal bölgeleri bit düzeyinde atanır. Bu sayede çok ufak çarpışmaların taşıt hareketini etkilemesi önlenir.

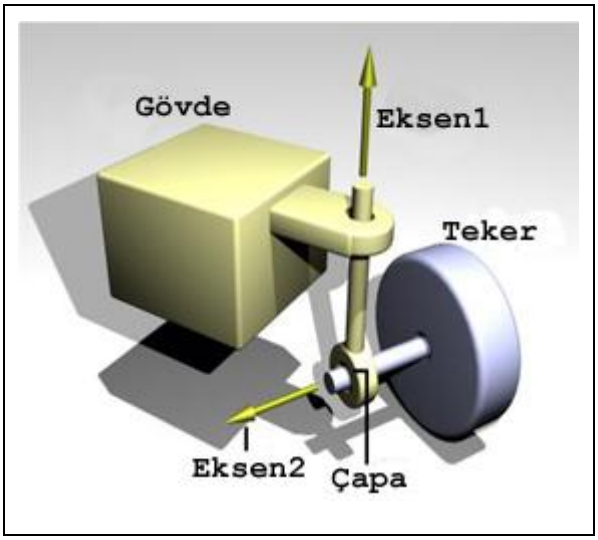
```
// Taşıt gövdesinin çarpışma geometrisini ayarla
mVehicle[VEHICLE_BODY]->RecenterGeometryUponLoad(true);
mVehicle[VEHICLE_BODY]->SetCollisionBox(vehicleBody.get());
// İhlal düzeyini belirle
mVehicle[VEHICLE_BODY]->SetCollisionCategoryBits(bodyCategoryBits);
mVehicle[VEHICLE_BODY]->SetCollisionCollideBits(bodyCollideBits);
// Tekerlerin çarpışma geometrisini ayarla
mVehicle[wheelType]->RecenterGeometryUponLoad(true);
mVehicle[wheelType]->SetCollisionCylinder(vehicleWheel.get());
// İhlal düzeyini belirle
mVehicle[wheelType]->SetCollisionCategoryBits(wheelCategoryBits);
mVehicle[wheelType]->SetCollisionCollideBits(wheelCollideBits);
```

Taşıtı hareket ettirmek için taşıta ivme kazandıracak bir kuvvet uygulanması gerekir. Kuvvetin şiddeti taşıtın kütlesi ile doğru orantılıdır. Taşıt kütlesi, gövde ve tekerlerin kütle değerlerinin toplamıdır. Taşıt gövdesinin ve tekerlerinin uygulanacak kuvvet şiddetine göre verecekleri tepkilerin dinamik kuralları çerçevesinde olması için EnableDynamics() metodu kullanılır. Bu metot ile hareketlinin hızlanma, yavaşlama, çarpışma testi gibi durumları, fizik kurallarına göre yürütülür.

```
// Taşıt gövde ve tekerleri için dinamik hesaplamasını aktif et
mVehicle[VEHICLE_BODY]->EnableDynamics();
mVehicle[wheelType]->EnableDynamics();
```

Hareketli bir nesne olan taşıt gövde ve tekerlerden oluşur. Taşıtın gövde ve teker aksamlarının birleştirmek için eklemlerden faydalanılır. ODE birçok eklem çeşidini destekler. Taşıt mekaniğine uygun eklem tipi, Menteşe2 eklemidir. Bu eklemde tekerler ve gövdeyi yönlendirecek iki eksen noktası vardır. Çapa noktasında teker ve gövde birleştirilir. Eksen1 ile gövdeye hareket yönü belirlenirken, Eksen2 ile tekerler döndürülür. Her bir teker kendi eklemi ile gövdeye eklenir. Bir taşıtta, sahip olduğu

teker kadar eklem bulunur. Eklemler, taşıtın gövdesi ve tekerler hareketliyi oluşturur. Bir hareketli eklem grubu ile temsil edilir.



Şekil 5.2. Hareketli eklem grubu.

```
// Hareketli için eklem grubunu tanımla
dJointGroupID mVehicleJointsGroup;
// Eklem grubuna Hinge2 eklem ekle
dJointID mVehicleJoints= dJointCreateHinge2(GetScene()-
>GetWorldID(), mVehicleJointsGroup);
mVehicleJoints = assert(mVehicleJoints);
// Gövdeyi eklem grubuna ekle
dJointAttach(mVehicleJoints[i], mVehicle[VEHICLE_BODY]->GetBodyID(),
mVehicle[i]->GetBodyID());
// Tekerin bağlantı pozisyonunu ayarla
wheelPos = dBodyGetPosition(mVehicle->GetBodyID());
assert(wheelPos);
// Çapa noktasındaki eklemi oluştur
dJointSetHinge2Anchor(mVehicleJoints, wheelPos);
// Eksen1 eklemini oluştur
dJointSetHinge2Axis1(mVehicleJoints, 0, 0, 1);
// Eksen2 eklemini oluştur
dJointSetHinge2Axis2(mVehicleJoints, 1 , 0, 0);
```

Yukarıda anlatılan işlem bütün tekerler için, teker pozisyonları dikkate alınarak tekrarlanır. Böylece taşıt oluşturulmuş olur. Taşıta hareket ve manevra kabiliyetleri vermek için gerekli taşıt dinamiği parametreleri atanmalıdır. Bu parametreler, taşıtın özelliklerine göre belirlenir. Eksen1'e verilen güç ile taşıtın direksiyon hareketleri kontrol edilir. Eksen2 ise tekerlerin dönerek taşıtın hareket etmesini sağlar. Taşıtın özelliğine göre (önden çekişli, arkadan çekişli veya dört çeker) tekerlere verilecek

güç değeri saptanır. Örneğin önden çekişli bir taşıt oluşturuluyorsa sadece ön tekerlerin Eksen2 eklemine güç verilir, diğer tekerlerin Eksen 2 eklemine güç verilmez. Hareketlinin, gerçek bir taşıt gibi hareket etmesini sağlamak amacıyla ERP, CFM, süspansiyon değerleri parametre olarak atanmıştır.

```
// Motor durdurma parametreleri
dJointSetHinge2Param(mVehicleJoints[i], dParamLoStop, 0.0);
dJointSetHinge2Param(mVehicleJoints[i], dParamHiStop, 0.0);
// Eksen1 için Hız ve güç parametreleri
dJointSetHinge2Param(mVehicleJoints[i], dParamVel, 0.0);
dJointSetHinge2Param(mVehicleJoints[i], dParamFMax, 30.0);
// Eksen2 için Hız ve güç parametreleri
dJointSetHinge2Param(mVehicleJoints[i], dParamVel2, 0.0);
dJointSetHinge2Param(mVehicleJoints[i], dParamFMax2, 100.0);
// Motorun tork, ERP, CFM ve süspansiyon parametreleri
dJointSetHinge2Param(mVehicleJoints[i], dParamFudgeFactor, 0.1);
dJointSetHinge2Param(mVehicleJoints[i], dParamBounce, 0.0);
dJointSetHinge2Param(mVehicleJoints[i], dParamERP, 0.75);
dJointSetHinge2Param(mVehicleJoints[i], dParamCFM, 0.005);
dJointSetHinge2Param(mVehicleJoints[i], dParamStopERP, 0.25);
dJointSetHinge2Param(mVehicleJoints[i], dParamStopCFM, 0.005);
dJointSetHinge2Param(mVehicleJoints[i], dParamSuspensionERP, 0.25);
dJointSetHinge2Param(mVehicleJoints[i], dParamSuspensionCFM, 0.005);
```

5.2.3. Taşıtın hareket ettirilmesi

Sistemde aday sürücü kontrolündeki taşıt ve otonom hareket eden vekil taşıt olmak üzere iki tip taşıt yer almaktadır. Bu taşıtlar tekerlerine verilen güç katsayısı ile hareket ettirilirler. Taşıtların yol üzerinde gerçekleştirecekleri manevralar için de ayrı bir yönlendirme değeri kullanılır. Aday sürücü kontrolündeki taşıt için tekerlere verilecek güç şiddeti ve manevra değeri, kullanılan giriş donanımlarından elde edilir (gaz, fren pedalları ile direksiyon). Otonom hareket eden vekil taşıt için bu değerler hazırlanan yordamlar ile elde edilirler. Bu değerler eklem grubuna atanarak taşıtın hareket ettirilmesi sağlanır.

```
// Yüksek dönüş değeri
dJointSetHinge2Param(mVehicleJoints, dParamHiStop, steeringLimit);
// Düşük dönüş değeri
dJointSetHinge2Param(mVehicleJoints, dParamLoStop, -steeringLimit);
// Direksiyon manevrası
dJointSetHinge2Param(mVehicleJoints, dParamVel, steeringVelocity);
// Tekerlere uygulanan güç
dJointSetHinge2Param(mVehicleJoints, dParamVel2, wheelVelocity);
```

Taşıt hareket ederken grafiksel olarak da taşıtın doğru konumlandırılması için eklem grubunda yer alan bağlantı noktalarının da hareket ettirilmesi gerekir. Bu hareket vektörel bir büyüklüktür ve X - Y düzlemlerinde gerçekleşir. Hareketlinin her bir tekerinde eklem bağlantısı olduğundan dolayı bu işlem bütün teker eklemlerine uygulanır. Böylece taşıt hareketi simülasyon sahnesinde görüntülenir.

```
// Taşıtın sürüş esnasında doğru konumlandırma işlemleri
// Hareket esnasındaki iki frame arasındaki çapa mesafelerinin ve
// eksen noktasının tespit edilmesi
dJointGetHinge2Anchor(mVehicleJoints[i], tempValue);
osg::Vec3 anchor1 = osg::Vec3(tempValue[0], tempValue[1],
tempValue[2]);
dJointGetHinge2Anchor2(mVehicleJoints[i], tempValue);
osg::Vec3 anchor2 = osg::Vec3(tempValue[0], tempValue[1],
tempValue[2]);
dJointGetHinge2Axis1(mVehicleJoints[i], tempValue);
axis1 = osg::Vec3(tempValue[0], tempValue[1], tempValue[2]);
// Ekleme güç uygulanması (tekerin döndürülmesi)
mVehicle->GetBodyWrapper()->ApplyForce(axis1 * -1.0);
```

5.2.4. Mesaj Sistemi

Sürücü adaylarına eğitim esnasında trafik kurallarını öğretmek maksadıyla verilecek geri bildirimler mesaj sistemi tarafından yönetilmektedir. Sürücü adayının her yaptığı trafik ihlali ile alakalı bir hata kodu üretilmektedir. Simülasyon süresince üretilen hata kodları mesaj sistemi tarafından alınır ve bu hata kodu değerine göre yazılı ve sesli mesajlar belirlenir. Hata kodlarına ait mesaj değerleri Çizelge 5.2'de gösterilmiştir.

Çizelge 5.2. Hata kodları ve mesajları.

Hata kodu	Mesaj	Hata Kodu	Mesaj
1	Sağa dönüş hatası	9	50 km/sa hız sınırı aşımı
2	Sola dönüş hatası	10	70 km/sa hız sınırı aşımı
3	U dönüşü yapılmaz hatası	11	90 km/sa hız sınırı aşımı
4	Kırımızı ışık hatası	12	120 km/sa hız sınırı aşımı
5	Sarı ışık hatası	13	Sol sinyal hatası
6	Park etme hatası	14	Sağ sinyal hatası
7	Duraklama hatası	15	Sinyal hatası
8	30 km/sa hız sınırı aşımı	16	Yanlış yol hatası

Eğitim esnasında verilen yazılı bir hata mesajı 5s süreyle görüntülenir. Test esnasında ise hata mesajları görüntülenmez, ama sürücü test raporuna eklenir. Sürücü adaylarına verilen sesli geri bildirimler, her bir hata kodu için önceden kaydedilen ve ilgili hataya ait sesli mesaj içeren ses dosyaları oynatılarak verilir. Sesli uyarılar 1 defa oynatılır.

Sesli mesaj verdirmek için OpenAL [43] API'sinden faydalanılmıştır. OpenAL, çok kanallı, üç boyutlu konumsal ses efektlerini çalıştırmak için kullanılan bir ses API'sidir. Creative tarafından geliştirilmesine devam edilen OpenAL, yeni nesil Windows oyunları için donanım hızlandırmalı ses desteği sağlamaktadır. Her hata koduna karşılık gelen ses dosyalarının isimleri, hata kodlarıyla birlikte errorfiles.dat dosyasında tutulmaktadır. Sürücü eğitim esnasında bir hata yaptığında ilgili hata kodu değeri ile CreateAudioMessage() fonksiyonu çağrılır. Hata kodu errorfiles.dat dosyasında aranır ve bu koda karşılık gelen ses dosyasının ismi errorfiles.dat dosyasından çekilerek 1 defa oynatılır.

Bir ses dosyasını oynatmak için ses yöneticisinin oluşturulması gerekir. Bu yönetici dinamik bellek alanları kullandığından dolayı uygulama kapatılmadan yok edilmelidir. Çalınması istenen ses dosyası için dtAudio::Sound sınıfında bir ses nesnesi oluşturulur ve bu öğeye ilgili ses dosyası yüklenir. Play metodu ile ilgili ses dosyası oynatılır.

```
// Ses yöneticisini oluştur
dtAudio::AudioManager::Instantiate();
// Ses nesnesi oluştur
dtAudio::Sound* drumRoll= dtAudio::AudioManager::GetInstance().NewSound ();
// Ses dosyasını yükle ve oynat
drumRoll->LoadFile(fSound);
drumRoll->Play();
// Ses yöneticisini yok et
dtAudio::AudioManager::GetInstance().Destroy()
```

5.2.5. Donanım Kontrolü

Aday sürücülere trafik eğitimi maksadıyla hazırlanmış simülatör sistemler, gerçek trafik ortamını yüksek seviyede temsil kabiliyetine sahip olmalıdırlar [47].

Simülatör, trafik ortamını bilgisayarda canlandırması yanında; sürücü adayının etkileşimini sağlayacak geniş ekranlara, direksiyon, pedallar ve vites kolu gibi taşıt donanımlarının kullanılmasına imkân sağlayacak bir mimariye sahip olmalıdır [11]. Simülatörlerde kullanılan bilgisayar ve etkileşim donanımlarının, gerçek yaşamdaki trafik ortamı ve sürücü-taşıt etkileşimini gerçeğe yakın bir şekilde gerçekleştirmesi gerekmektedir. Aksi takdirde sürücü adayına simülatör ortamında kazandırılan taşıt sürüş kabiliyeti ve güvenli sürüş becerileri, sürücü gerçek trafik ortamına çıktığında olumsuz sürücü davranışlarına yol açabilir.

TSS'de çıkış birimi olarak kullanılan görüntü elemanlarının kontrolü görüntü platformu ile yapılır.

5.2.5.1. Görüntü Platformu

Trafikte taşıt ilerlerken sürücü gerçek hayatta ön camdan önde olup biteni görebildiği gibi kafasını sağa, sola çevirerek yan camlardan sağ tarafını veya sol tarafını; aynaları kullanarak arka tarafı görebilir.

Trafik eğitimi yapmak için hazırlanan bir simülatörün, sürücünün gerçek hayatta olduğu gibi simülasyon ortamında da farklı yönlerde oluşan trafik hareketliliğini görerek, taşıt sürme performansını sürdürmesine imkân tanıması gerekir. Bir sürücü taşıt sürme faaliyetini gerçeğe yakın bir şekilde sürdürebilmesi için önünde akan trafiğin yanında, sağından, solundan ve arkasından gelen taşıtları da takip etmesi gerekir. Sürücü adayının simülasyon süresince farklı konumlardaki görüntüleri görebilmesi için TSS sistemine bir görüntü platformu eklenmiştir. Sürücü görüntü platformu üzerinde oluşturulan sanal dünyayı görür ve bu ortamda taşıt kullanır. Görüntü platformu iki adet ekran kartı, üç adet monitör ve program kodlarından oluşmaktadır.

Görüntü Platformu Donanımları

TSS'de, her bir simülasyon çerçeve zamanında (frame rate) yeniden üretilen görüntüler, üç adet monitörde gösterilmektedir. Monitörler görüntüleri, iki adet ekran

kartından temin etmektedir. Çizelge 5.2'de TSS görüntü platformunda kullanılan monitörler ve bu monitörlere görüntü sağlayan ekran kartları gösterilmektedir.

Çizelge 5.3. TSS Görüntü Platformu Donanımları.

Görüntü Birimi	Kaynak	Kamera	Görev
BENQ 2220HDA LCD	Quadro FX1800	kamera_ana	Ana Ekran
Vestel PA-788K 17 LCD	Geforce 9500 GT	kamera_sol kamera_ana	Sol Cam, Arka Ayna
Yakumo 17 LCD	Geforce 9500 GT	kamera_sag	Sağ Cam

Görüntü oluşturma

Bilgisayar grafiğinde sahne ekranı üzerinde oluşturulan görüntü, kamera olarak isimlendirilen grafik nesnesi vasıtası ile gösterilir. Daha net bir tabirle normal hayatta insan gözünün yaptığı işi, bilgisayar grafiklerinde kamera yapar. İnsan nereye bakarsa, baktığı noktadan sonraki bakış açısındaki görüntüyü görür. Benzer şekilde simülasyon ortamında, istenen bir bölgenin görüntülenmesi için ilgili alanı görebilecek bir pozisyonda ve açıda bir kameranın yerleştirilmesi gerekir.

TSS'de görüntü, isimleri kamera_ana, kamera_sol ve kamera_sağ olmak üzere üç adet kamera kullanılarak elde edilmektedir. Taşıt ilerlerken simülasyon ortamını gösteren görüntüyü kamera_ana, sağ yan camdan görünebilecek görüntüyü kamera_sağ ve sol yan camdan görünebilecek görüntüyü kamera_sol temin etmektedir. Taşıt ilerlerken kamera görüntülerinin de tazelenmesi gerekir. Taşıt ilerlerken transformasyon matrisi üzerinde yer değiştirir. Simülatörde doğru görüntü elde etmek için sistemdeki kameraların da taşıt istikametinde ilerletilmesi ve doğru açıda konumlandırılması gerekir. Bu işlemi gerçekleştirmek için taşıtın her bir simülasyon çevrim zamanında bulunduğu konumu hesaplayan "follower" isminde bir sınıf oluşturulmuştur. Taşıtın bulunduğu konumun, taşıtın ilerleme yönüne göre biraz gerisi (bu değer x veya y eksen değerlerine göre hesaplanır) kamera_ana isimli kameranın konumudur. kamera_ana ile diğer kameraların konumları aynıdır. Bunun

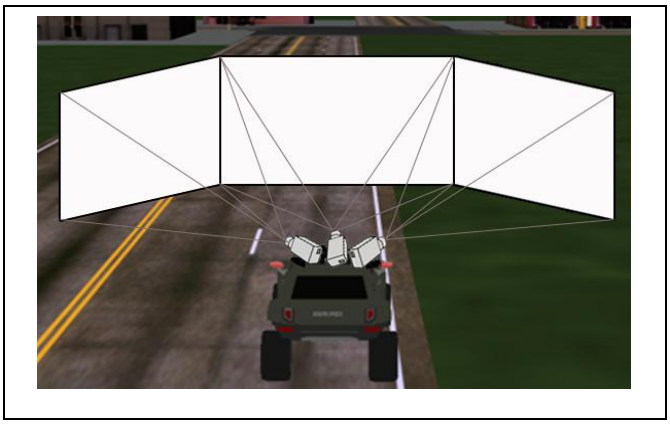
yanında sağ ve sol taraftaki görüntülerin elde edilebilmesi için bu kameraların (kamera_sağ ve kamera_sol), ana kameraya (kamera_ana) göre 45°'lik açılarla yerleştirilmeleri gerekir. Bunun için ilgili kameraların, ana kameranın pozisyonu referans alınarak 45° döndürülmesi gerekir. Böylece sürücü sağ çapraz ve sol çaprazdan gelen trafik akışını görerek yavaşlama, hızlanma veya durma gibi sürücü davranışlarını ayarlayabilir. Sürücünün taşıtın arkasında süregelen olayları (örneğin taşıtı takip eden veya geçecek durumda olan taşıtları) izleyebilmesi için ana kamera 180° geriye döndürülür.

Taşıtın ilerleme istikameti ve bulunduğu eksene göre kameraların döndürülme açıları farklılık göstermektedir. Çizelge 5.4'de taşıtın bulunduğu eksene ve taşıt yönüne göre kameraların döndürülme dereceleri verilmiştir. Elde edilen bu konum bilgisi ile kamerada oluşturulan görüntüler, simülasyonun her bir çevrim zamanında, update camera fonksiyonu ile tazelenir.

Çizelge 5.4. Hareket eksenine göre kamera açıları.

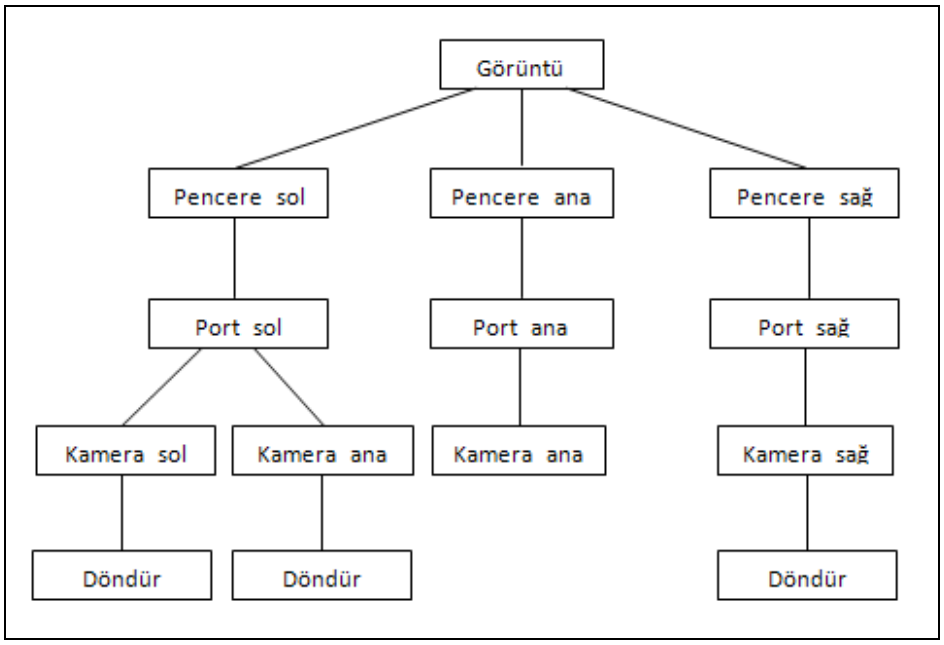
Hareket Ekseni	Sol Kamera (°)	Sağ Kamera (°)	Arka Kamera (°)
y+	45	-45	180
y-	-135	135	0
x+	-45	225	90
x-	-225	-90	45

TSS'in gerçeğe yakın bir trafik ortamını sunması için, kamera kullanılarak elde edilen görüntülerin, görüntü platformunda bulunan monitörlere aktarılması gerekir. Böylece taşıt simülasyon esnasında ilerlerken taşıtın önündeki, sağındaki ve solundaki görüntüler, görüntü platformu üzerindeki monitörlerde gösterilirler.



Şekil 5.3. TSS kameraları ve görüntü platformu.

Sahne grafı yöntemi kullanılarak hazırlanan TSS ortamında farklı kameralardan elde edilen görüntüler yine bir graf üzerinden görüntü platformuna taşınmıştır. Bu amaçla TSS yazılımında, her bir monitöre münhasır olarak çalıştırılmak üzere üç farklı pencere oluşturulmuştur. Her bir pencere içinde kameradan gelecek görüntünün oluşturulacağı görüntü portuna ihtiyaç vardır. Zira kameralardan elde edilen görüntüler, monitörler için hazırlanmış pencerelerde açılan görüntü portları üzerinden yansıtılabilir. Port ana ve Port sağ görüntü portları ayrı ayrı kameralardan (Port ana ← Kamera ana, Port sağ ← Kamera sağ) beslenirken, Port sol görüntü portu iki ayrı kameradan (Port sol ← Kamera ana, Port sol ← Kamera sol) beslenmektedir. Bununla birlikte, Çizelge 5.4'de verilen kriterler referans alınarak, ilgili kameranın, transformasyon matrisi üzerinde döndürülmesi ile elde edilen kamera görüntüleri, görüntü portlarına gönderilmesiyle monitörlerde gösterilecek görüntü oluşturulur. Örneğin taşıt y+ hareket ekseninde ilerlerken, taşıtın sol tarafındaki görüntünün, sol monitöre yansıtılabilmesi için, Kamera sol'un 45° döndürülmesi gerekir.

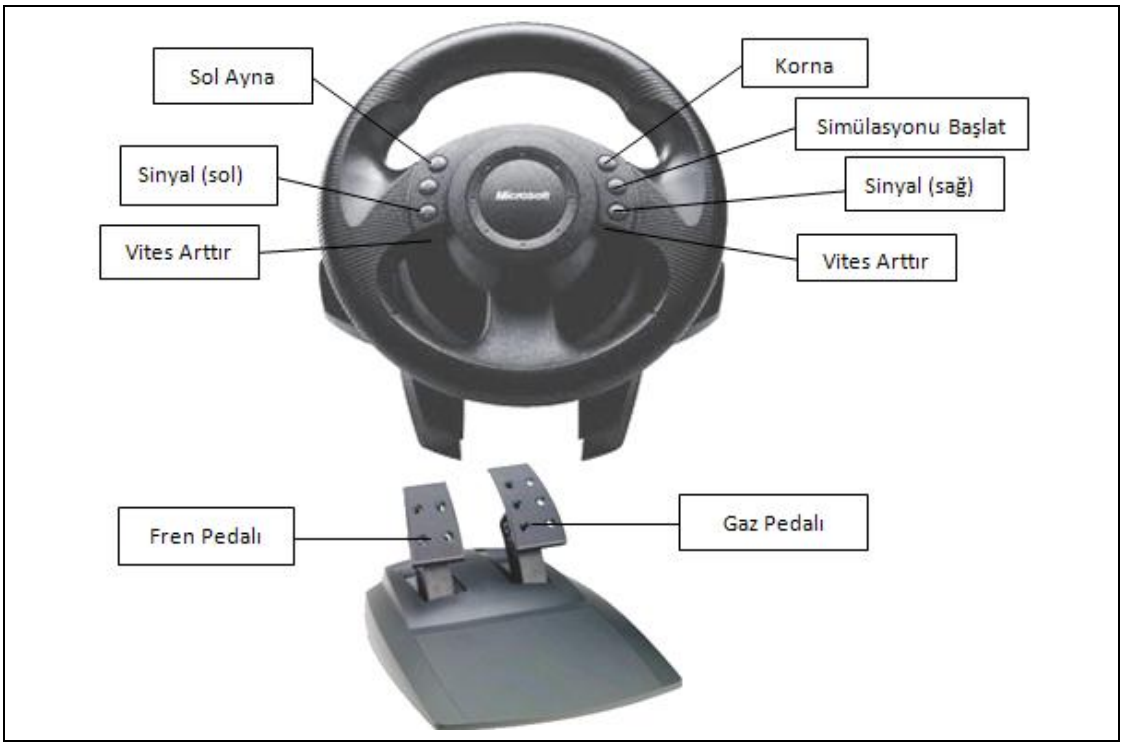


Şekil 5.4. Görüntü platformu sahne grafı.

TSS yazılımında sürücü adayı taşıt kullanırken, arkadan akan trafiğin görüntülenmesi için, kullanılan ön kamera 180° döndürülerek elde edilen görüntü sol monitöre gönderilir. Taşıtın arkasında olup bitenin sol monitörde görüntülenmesi seçimliktir. Bu durum bir tuş kontrolü ile gerçekleştirilir. Yani sürücü taşıtını sürerken kullandığı direksiyon üzerinden bir düğmeye basarak arkada oluşan görüntüyü sol monitöre yansıtabilir. Sürücü işi bittiğinde aynı düğmeyi kullanarak sol görüntüyü tekrar sol monitöre verebilir.

5.2.5.2. Direksiyon ve Pedal Donanımları

Sürücü adaylarının simülasyon ortamında gerçekçi olarak sürücü becerilerini geliştirmeleri için, gerçek taşıtta bulunan direksiyon, pedallar, korna, sinyal kolları, vites gibi donanımları kullanmaları yararlı olabilir. Bu maksatla TSS yazılımına direksiyon ve pedal donanımları eklenmiştir. TSS yazılımında Microsoft SideWinder Precision Racing Wheel kumanda donanımı (joystick) kullanılmaktadır. Bu donanım Şekil 5.5'de görüleceği üzere direksiyon ve pedallardan oluşmuştur. Pedallar direksiyona RJ11 konnektörü ile PSTN (Public Switched Telephone Network) ağı ile bağlanır. Direksiyon ise bilgisayar ile USB bağlantısı ile haberleşir.



Şekil 5.5. TSS direksiyon ve pedal donanımları.

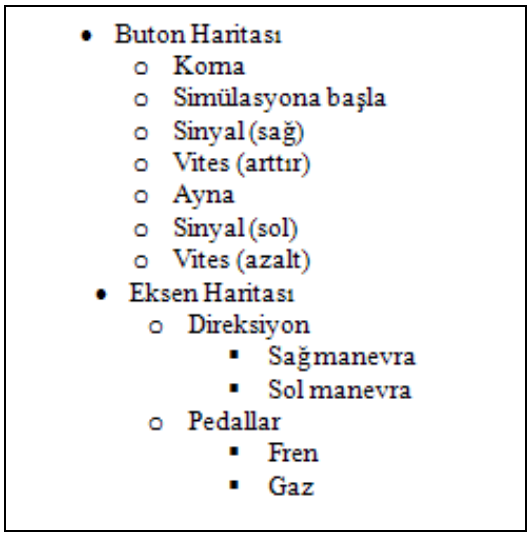
Donanımdan, bilgisayara gönderilen sinyaller iki şekildedir:

1. Direksiyon simidinin sağa sola çevrilmesi ve pedallara basılması ile oluşan eksen (axis) sinyalleri
2. Direksiyon simidi üzerinde yer alan düğmelere basılması ile oluşan buton sinyalleri.

TSS, bu donanımlardan gelen sinyalleri algılamak için PLIB kütüphanesini kullanmaktadır. Direksiyon ve pedal donanımlarını kullanabilmek için öncelikle yazılımın kurucu fonksiyonunda kullanılacak donanımların ve bu donanımlara ait eksen ve buton değerlerinin belirlenmesi gerekir.

Simülasyon süresince, her bir çevrim zamanında kurucu fonksiyonda belirtilen kumanda aygıtları dinlenerek, herhangi bir giriş yapılması durumunda ilgili girişe ait gerçekleştirilmesi gereken fonksiyon tetiklenmelidir. Kumanda aygıtları bilgisayara eksen ve buton sinyalleri göndererek bir giriş bildiriminde bulunabilirler. Simülasyonun her bir çevrim zamanında kumanda aygıtlarından bir giriş eylemi olup

olmadığı eksen ve buton sinyallerinin ayrı ayrı takip edilmesi ile tespit edilebilir. Bu yüzden simülasyon döngüsünde buton ve eksen sinyalleri ayrı ayrı haritalanarak, bu sinyallerin sahip oldukları davranışların tespit edilmesi gerekir. Bu maksatla giriş sinyallerini takip etmek için Şekil 5.6'da gösterilen *giriş takip yapısı* kurulmuştur. Eksen haritasında bulunan donanımlardan gelen sinyaller durum verisi içerirken, buton haritasından gelen sinyaller açık/kapalı (on/off) veya 1-0 biçiminde sayısal veri içerir.



Şekil 5.6. Giriş Takip yapısı.

Simülasyon döngüsü esnasında eksen ve buton haritalarında bulunan kumanda donanımları aracılığı ile giriş yapılma durumu sürekli takip edilir. Sürücü adayının herhangi bir anda kumanda aygıtları ile yaptığı giriş eylem türü (eksen veya buton girişleri), farklı mantıksal fonksiyonlar ile takip edilir. Bir giriş eylemi olduğunda, bu eylem buton haritası kapsamında ise buton durumlarını takip eden fonksiyon değeri true olur ve buton haritasında yer alan ilgili aygıt üzerindeki hangi fonksiyonun tetiklendiği tespit edilir. Tetikleme sinyali 1 veya 0 (aç / kapat) olan veri içerir. Buton sinyallerinin durumu, bağlı bulunduğu işlemi çalıştırabilir, başlatabilir veya başlatılmış bir işlemi sonlandırabilir. Örneğin "Korna" butonuna her basıldığında korna sesi çalınır; "Ayna" butonuna ilk kez basıldığında, taşıt trafikte seyir halindeyken taşıtın arkasındaki trafik akışının görüntüsü sol monitöre yansıtılır; "Ayna" butonuna ikinci kez basıldığında sol monitöre taşıtın trafikte ilerlerken sol penceresinden görülebilecek görüntü yansıtılır. Buton haritasında yer alan bir

davranış tetiklendiğinde, o davranış için yazılmış yordam çalıştırılır. Şekil 5.7'de bir giriş olayı gerçekleştiğinde yapılacak işlemlerin algoritması gösterilmiştir.

```
Simülasyon döngüsü
{
    Buton hareketi var mı?
        "E": buton_id=Hangi Buton
                ButonÇalıştır(buton_id)
    Eksen hareketi var mı?
        "E": eksen_id=Hangi Eksen
                Hesapla(eksen_durum)
                EksenÇalıştır(eksen_id, eksen_durum)
}
```

Şekil 5.7. Buton ve eksen haritalarında gerçekleşen giriş sinyallerinin yürütüm algoritması.

Eksen haritası kapsamında bir giriş eylemi olduğunda, eksen girişlerini takip eden fonksiyonun değeri true olur ve eksen haritasında yer alan ilgili aygıt üzerindeki hangi fonksiyonun tetiklendiği tespit edilir. Tetikleme sinyali, ilgili aygıtın durumunu gösteren veri içerir. Örneğin bu veri "Direksiyon" ekseninin saat yönünde veya saat yönünün tersi yönde ne kadar çevrildiği değerleri olabilir. Eksen haritasında yer alan bir davranış tetiklendiğinde, o davranış için yazılmış yordam ilgili birimi ait giriş sinyalinin değerine göre çalıştırılır. Örneğin "Gaz" pedalına basıldığında taşıt bir ivme ile hızlandırılırken, "Fren" pedalına basıldığında taşıt yine bir ivme ile yavaşlatılır.

5.3. YAPAY ZEKÂ MODÜLÜ

TSS'nin yapay zekâ modülü, trafikte yer alan öğelerin sistem içindeki davranışlarının kontrolünü gerçekleştirir. Sistemde kontrol edilen öğeler sürücü adayı kontrolündeki taşıt, trafik akışını oluşturacak etmen taşıtlar, trafik ışıkları olarak sıralanabilir. Sürücü adayı kontrolündeki taşıt yapay zekâ modülünün ilgili yordamlarını kullanarak, sürücü adayının direktifleri ile hareket ettirilebilir, hızlandırılabilir, yavaşlatılabilir veya dönüş manevraları yaptırılabilir. Bu modülün asıl işlevi taşıt etmenlerinin otonom olarak idare edilmesidir. Etmenlerin trafikte yapması gereken davranışlar şerit takibi yapmak, trafik yoğunluğunu belirlemek, taşıt takibi yapmak,

trafik ışık kurallarına uymak, trafik işaret kurallarına uymak ve şerit değiştirmek olarak sayılabilir. Trafik ışıklarının durumları ve çalışma frekansları yapay zekâ modülü ile kontrol edilmektedir.

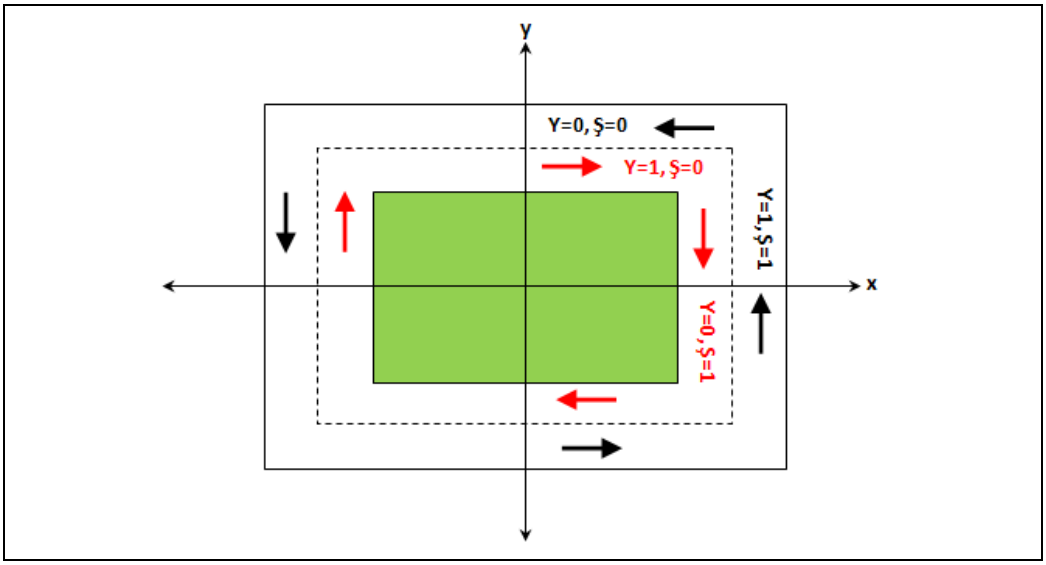
5.3.1. Şerit Takibi

Trafikte ilerleyen taşıtlar kendilerine ayrılan şeritlerde hareket etmek zorundadırlar. Özel bir durum oluşmadıkça taşıtlar kendi şeritlerini takip ederler. Bu yüzden trafik simülasyon sisteminde öncelikle bir etmenin şerit takibi işlevi gerçekleştirilmeye çalışılmıştır.

Bir etmenin şerit takibi yaparak hareket ettirilmesi işlemi şöyledir:

1. Etmen, kara yolu üzerinde bir başlangıç noktasına (x, y, z) konumlandırılır. Etmenin bulunduğu konuma ait yol verisi yol bilgisi dosyasından temin edilerek bir sınıfa aktarılır. Bu sınıfta etmenin bulunduğu konumdaki yol adı, segment bilgisi, şerit bilgisi gibi veriler bulunmaktadır. Etmen yol üzerinde bir şeritte olabileceği gibi bir kavşakta da olabilir. Bu durum kontrol edilerek elde edilen sonuca göre etmenin hareket rotası belirlenebilir.

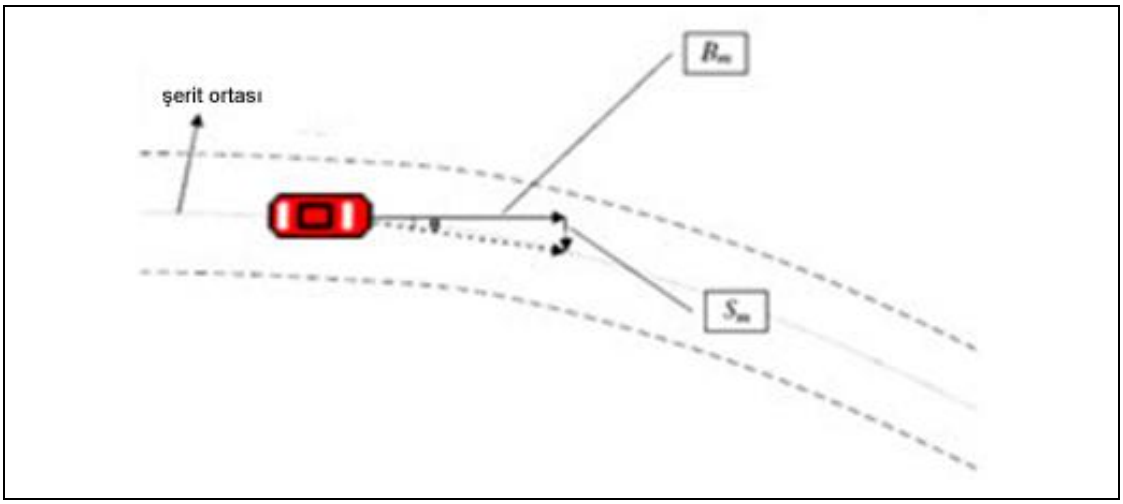
2. Etmen bir yolun şeridinde ise etmenin bulunduğu segmentin başlangıç noktası, sürüş başlangıç noktası olarak alınır. Etmenin hareket yönü ve bulunduğu yolun koordinat eksenine göre durumu bitiş sürüş noktasının belirlenmesinde önemlidir. Zira etmen ilerlerken konum değiştirecektir. Bu konum değiştirme aracın yönüne bağlı olarak yatay eksende veya dikey eksende bulunduğu konumun pozitif veya negatif yönde değişmesine neden olabilir. Pozitif veya negatif değişiklik etmenin sürüş bitiş noktasının tespitinde önem arz eder. Bu sebeple etmenin hareket yönü ve etmenin bulunduğu şeridin koordinat eksenine göre durumu sürüş bitiş noktasının tespitinde belirleyici değerlerdir. Şekil 5.8'de koordinat eksenine göre hareket yönü (Y) ve şerit durumu (Ş) gösterilmiştir.



Şekil 5.8. Koordinat eksenine göre etmenin hareket yönleri ve şerit durumları.

Hareketli etmen için sürüş bitiş noktası yol bilgisi dosyasından elde edilir. Hareket yönü "1" ve şerit yönü "1" olan bir konumda sürüş bitiş noktası, segmentin bitiş noktasıdır. Hareket yönü "0" ve şerit yönü "1" olan bir konumda sürüş bitiş noktası, segmentin başlangıç noktasıdır.

Sürüş bitiş noktası tespit edildikten sonra etmenin direksiyon manevrası yapılıp yapılmayacağına karar verilmesi için yolun eğiminin kontrol edilmesi gerekir. Bu amaçla sürüş başlangıç noktası ile sürüş bitiş noktası arasındaki eğrilik değeri hesaplanarak ilerlenen şeridin düz ya da kıvrımlı oluğu belirlenebilir. Düz segmentlerde etmen herhangi bir direksiyon manevrasına ihtiyaç duymadan şerit takibini gerçekleştirebilir. Fakat kıvrımlı segmentlerde, bir etmen bulunduğu şeritte ilerlerken, anlık olarak hesaplanan ilerleme istikametinde belirli bir mesafe (Bakış mesafesi [Bm]) kadar ilerlemeye devam etmiş olsaydı, bulunduğu şeridin ortasından ne kadar sağa ya da sola sapmış olacağı hesaplanması gerekir (Sapma mesafesi [Sm]). Bu sapmayı düzeltmek için gerekli olan yeni direksiyon konumu belirlenir. Bu sayede, etmenlerin kendilerine ayrılan şeridi takip etme davranışları modellenmiş olur.



Şekil 5.9. Etmenin kıvrımlı yollarda şerit takibi.

Şerit takibi yapan araçlar düz olmayan şeritlerde ilerlerken (Şekil 5.9) bazı direksiyon manevraları yapmak zorundadırlar. Direksiyonun ne kadar çevrileceği şöyle tespit edilir.

$$k \times \theta = tan^{-1}\left(\frac{2}{(Bm)^2} \times Sm\right) \quad [48] \qquad (5.1)$$

k : Direksiyon döndürme katsayısı [sabit]

θ : Sapma açısı [derece]

Bm : Bakış mesafesi [metre]

Sm : Sapma mesafesi [metre]

Etmen kavşakta ise etmenin bulunduğu konum sürüş başlangıç noktası olarak alınır. Etmenin sürüş bitiş noktası ise etmenin geldiği segmentin yol bilgisi dosyasında bağlı olduğu hedef segmentlerden birinin başlangıç noktasıdır. Kavşakta yer alan bir etmen, birden fazla farklı istikamete devam edebilir. Belirlenen hedef sürüş noktası ile başlangıç sürüş noktası arasındaki eğrilik değeri, olası direksiyon manevrası için gerekli katsayıyı tespit eder. Belirlenen hedef farklı eksende ise etmenin yönü diğer eksene paralel oluncaya kadar direksiyon manevrası yaptırılır. Kavşaktaki etmen yönünü değiştirmeden de başka bir segment üzerinden yoluna devam edebilir. Bu durumda yapılacak direksiyon manevrası başlangıç sürüş noktası ile bitiş sürüş noktası noktaları arasındaki eğriliğe göre gerçekleştirilecektir.

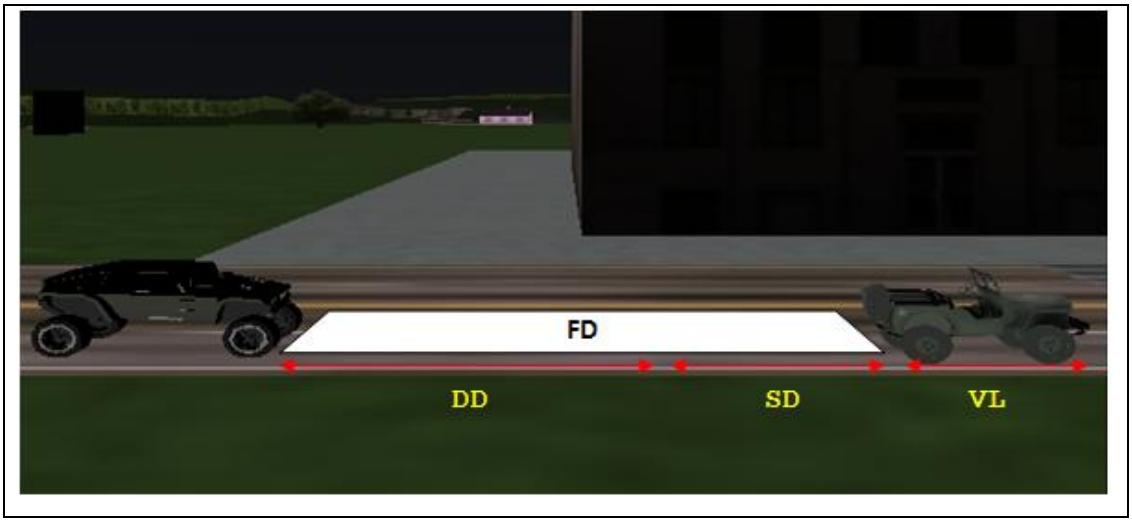
Etmen ilerlerken gideceği bir sonraki hedef bilgisi, sürüş bitiş noktası kontrol edilerek yenilenir. Aynı yol üzerindeki bir segmentin hedef segmenti yol üzerinde kendisinden sonra gelen segmenttir. Farklı bir yola geçerken ise hedef segment birden çok olabilir. Bu durum etmen kavşakta iken oluşur. Kavşağa bağlı olan her bir segment alternatif hedef segmentlerdir. Etmenin gideceği istikamet veya hedef segment ağ akışının yoğunluğunu belirleyen algoritmaya göre belirlenecektir.

Şekil 5.10. Segment ve kavşak bağlantıları.

Şehir üzerinde yer alan segmentler Şekil 5.10'da gösterilmiştir. Aynı yol üzerindeki segmentlerde etmenlerin ilerlemesi bulunulan segmentin başlangıç noktası (a), başlangıç sürüş noktasıdır. Segmentin bitiş noktası (b) ise bitiş sürüş noktasıdır. Farklı yollarda veya kavşaklarda etmenin ilerlemesi için başlangıç sürüş noktası kaynak segmentin bitiş noktası (b), bitiş sürüş noktası ise hedef segmentin başlangıç noktası (c) olur.

5.3.2. Taşıt Takibi

Bir etmen trafikte ilerlerken önünde yer alan diğer etmenlere ve sürücü adayının kullandığı taşıta çarpmadan hareketine ya devam etmeli, ya yavaşlamalı veya durmalıdır. Bu amaçla etmen önünde yer alan taşıtla arasında belirli bir güvenlik mesafesi ayarlamalıdır. Şerit değiştirme veya öndeki taşıtı geçme durumu oluşana kadar etmen önündeki aracı güvenlik mesafesi bırakarak izlemelidir. Bu tip davranışlar bir etmenin taşıt takibi davranış özelliklerini kapsar.



Şekil 5.11. Taşıt takip parametreleri.

TSS'de etmenler taşıt takip edebilme kabiliyetindedirler. Bir etmen taşıt takibi işlevini trafikte yer alan araçların pozisyonlarını kontrol ederek gerçekleştirmektedir. Her bir aracın uzunluğu (VL), güvenlik mesafesi (SD) ve yavaşlama mesafesi (DD) değerleri referans alınarak takip mesafesi (FD) hesaplanır. Şekil 5.11'de bir etmenin bırakması gereken sınır takip mesafesi parametreleri gösterilmiştir. Etmenin yavaşlama mesafesi, etmen hızının binde birinin yarısı kadardır. Güvenlik mesafesi ise aracın uzunluğu değerindedir. Takip mesafesi yavaşlama mesafesi ile güvenlik mesafesi toplanarak hesaplanır. Örneğin bir etmen 70 km/sa hız ile gidiyorsa ve uzunluğu 5 m ise bu etmene uygulanacak takip mesafesi 40 m olarak hesaplanacaktır.

```
Aktif etmenin takip mesafesi değerlerini hesapla
Diğer taşıtların takip mesafesi değerlerini hesapla
Aktif etmen ile diğer taşıtların pozisyonlarını karşılaştır.
        Şart 1 : FD dışında ise etmen hareketine devam et.
        Şart 2 : DD sınırında ise etmen hızını azalt.
        Şart 3 : SD sınırında ise etmeni durdur.
```

Şekil 5.12. TSS taşıt takip davranışının pseudo kodları.

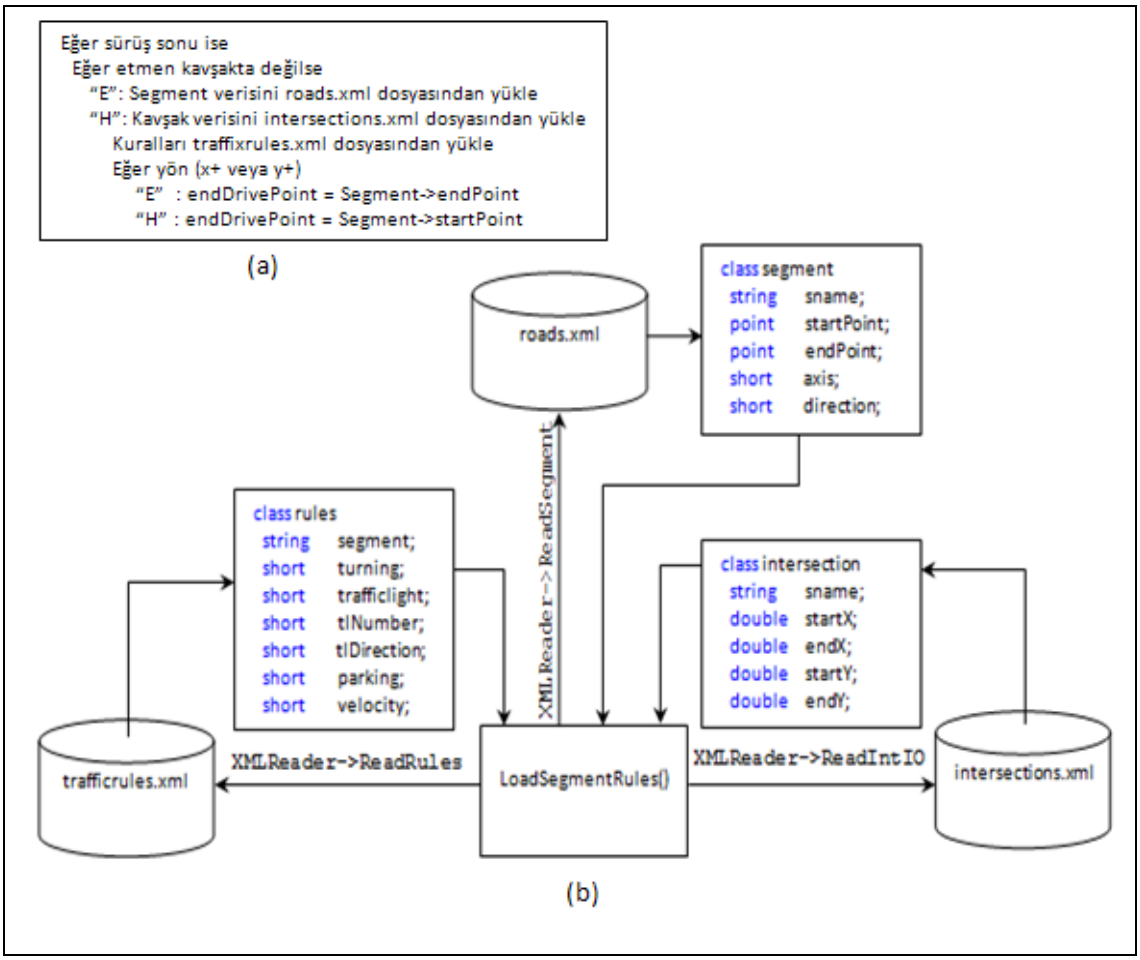
Taşıt takip işleminin pseudo kodları Şekil 5.12'de gösterilmiştir. Her bir simülasyon işlem zamanında taşıt takip işlemleri tekrarlanır. Taşıt takibi için ilk olarak sürücü

adayının aracı için bulunduğu konuma göre takip mesafesi parametreleri bulunur. Trafikte yer alan her bir etmen için (hali hazırda işlenen etmen dışında) takip mesafesi parametrelerinin tespit edilmesi tekrarlanır. İşlenen etmenin simülasyon alanında işgal ettiği konum verisi, takip mesafesi hesaplanan trafikteki bütün taşıtlar ile karşılaştırılır. Etmenin konumu hiçbir takip mesafesi değeri içinde değilse etmen hareketine devam eder. Etmen herhangi bir aracın yavaşlama mesafesinde ise etmen yavaşlayarak hareketine devam eder. Etmen herhangi bir aracın güvenlik mesafesinde ise etmen durur. Bu döngü bütün etmenler için simülasyon süresince sürdürülür. Böylece etmenlerin takip mesafesine uyarak araç takibi işlevi gerçekleştirilmiş olur.

5.3.3. Trafik Kuralları

TSS sürücü adaylarının, bulundukları segmentdeki trafik kurallarına uyum durumlarını kontrol etmektedir. Trafficrules.xml dosyası simülasyon ortamına ait trafik kuralları verisini içerir. Sürücü adayı davranışlarının trafik kurallarına uygunluğunun kontrolü için, adayın kullandığı taşıtın, o anda bulunduğu segmente ait trafik kurallarının bilinmesine ihtiyaç vardır. Bu maksatla bulunulan segmentin kurallarını tutmak için bir kural veri yapısı oluşturulmuştur. Bu veri yapısı o segmente ait trafik kurallarını içerir. Sürücü adayının kullandığı araç, ilgili segmentte kaldığı sürece bu veri yapısında tutulan değerlere tâbi tutulur. Kural veri kümesi eğer ve ancak diğer segmente geçilirse yenilenir. Aracın segment değiştirdiği, bulunulan yolun koordinat eksenine göre durumu ve aracın hareket yönü parametrelerine göre hesaplanan sürüş bitiş noktası değerine ulaşılması ile tespit edilir. (x+ veya y+) eksen yönünde ilerleyen bir araç için sürüş bitiş noktası bulunulan segmentin son noktası iken, (x- ve y-) eksen yönünde ilerleyen bir araç için sürüş bitiş noktası bulunulan segmentin başlangıç noktasıdır.

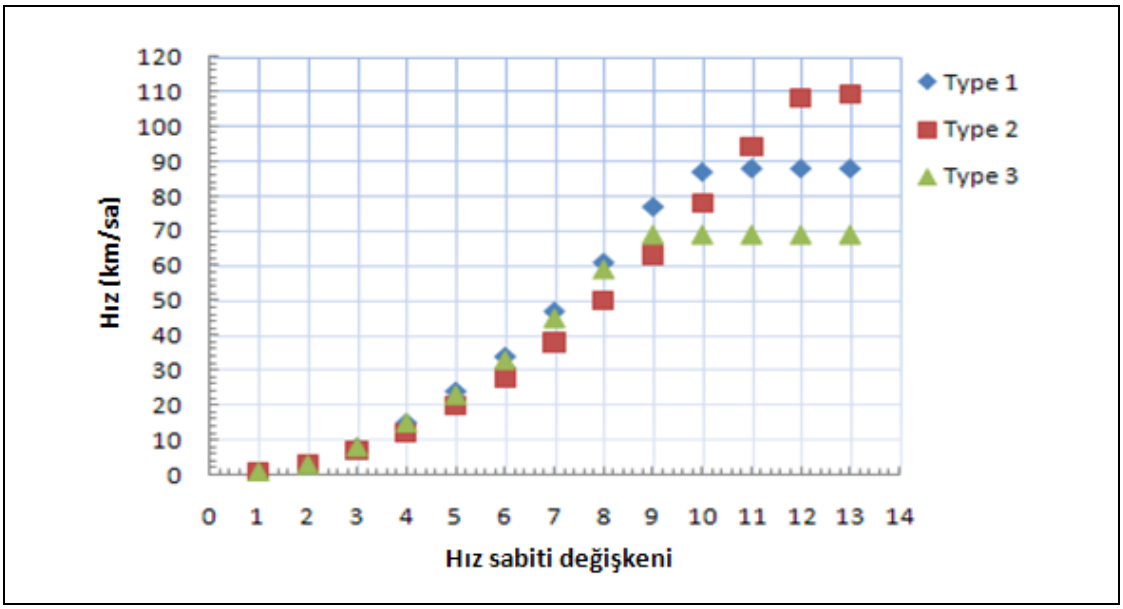
Sistemde yer alan bir araç her segment değiştirdiğinde veya kavşak noktasına girdiğinde LoadSegmentRules() fonksiyonu çalıştırılır. Bu fonksiyon ilgili segmentte kontrol edilmesi gereken kural kümesini ve aracın sürüş bitiş noktasını bulur. Şekil 5.13.(a)'da LoadSegmentRules() fonksiyonunun kısa kodları, (b)'de ise bu fonksiyonun blok diyagramı gösterilmiştir.



Şekil 5.13. LoadSegmentRules fonksiyonunun a) Pseudo kodları, b) blok diyagramı.

5.3.3.1. Hız Kontrolü

Etmenlerin farklı segmentlerde farklı hızlarda hareket etmek durumundadırlar. Kimi zaman hızı belirleyen unsur hız sınırını gösteren trafik işareti olurken, kimi zaman görsel olmayan trafik kuralı, kimi zaman önünde ilerleyen bir taşıt, kimi zaman ise kavşağa giriş ve kavşaktan çıkış pozisyonları olabilmektedir. Farklı her bir durum için etmenler hız sabiti değişkenlerini değiştirerek hızlarının kontrol ederler. Şekil 5.14'de farklı tipteki araçların hız değişkenlerine göre azami hız değerleri gösterilmiştir. Trafikte seyreden etmenlerin kontrollü hareket etmeleri, bulundukları segmentin özellikleri, taşıt takip mesafesi ve trafik ışıklarının durumuna göre hız değişkenleri değiştirilerek sağlanır. Yani etmen için trafik serbest akışında ise hız değişkeni artırılır, aksine etmenin yavaşlaması veya durması gerekiyorsa hız sabiti değişkeni azaltılır.



Şekil 5.14. 3 farklı tipteki aracın hız değişkenlerine göre azami hız değerleri.

Kontrollü - kontrolsüz kavşakların veya hız sınırlama trafik işaretlerinin bulunduğu segmentler için etmenlerin hız sabiti katsayısı, etmenin konum bilgisi özelliği kontrol edilerek saptanır.

5.3.3.2. Trafik Işıkları

Etmenlerin hareket istikametlerinde kavşak bağlantıları olabilir. Etmenler karşılaşabilecekleri kontrolsüz veya kontrollü (trafik ışıklı) bir kavşak durumu verisini roads.xml dosyasından elde ederler. Segment elemanında yer alan TL özelliğinin değeri, etmene istediği kavşak bilgisini sağlar. TL değerinin durumuna göre etmenin göstereceği davranış özellikleri Çizelge 5.4'de gösterilmiştir. Etmen kontrolsüz kavşaklarda yavaşlayabilir veya kavşağa bağlı diğer yollarda bulunabilecek araçların durumuna göre yoluna devam edebilir veya durabilir. Kontrollü kavşaklarda ise etmen yavaşlayabilir veya trafik ışığının durumuna göre davranır.

Çizelge 5.4. Kavşak kontrol parametreleri.

TL Değer	Tanım	Etmen Davranışı
0	Kavşaktan uzak	Hızı koru
1	Kavşağa yakın	Hızı azalt
2	Kavşağa çok yakın	Trafik ışığı veya kavşağa gelen diğer segmentlerdeki etmen durumunu kontrol et: Dur veya devam et

Kontrollü kavşaklarda her bir köşe iki trafik lambası ile kontrol edilmektedir. Çizelge 5.5'de Int04 isimli kavşağa gelen araçların uymaları gereken trafik ışıkları ve bu ışıkların durumlarına göre devam edecekleri istikametler yer almaktadır. Sistemdeki bütün etmenler karşılaştıkları trafik ışıklarına uymak zorundadırlar. Örneğin I_2 alanından kavşağa giren ve hedef segmenti O_1 olan bir etmen ancak TL2b isimli trafik ışığı yeşilken bu istikamette ilerleyebilir, aksi durumda TL2b yeşil olana kadar durur.

Çizelge 5.5. Geçiş kontrolü

Trafik Işığı	Giriş	Çıkış
TL1a	I_1	O_2, O_3
TL2a	I_1	O_4
TL1b	I_2	O_3, O_4
TL2b	I_2	O_1
TL1c	I_3	O_1, O_4
TL2c	I_3	O_2
TL1d	I_4	O_1, O_2
TL2d	I_4	O_3

5.3.3.3. Trafik İşaretleri

TSS'de hareket eden etmenler istikametleri boyunca karşılaşacakları trafik işaretlerinde gösterilen trafik kurallarına uyarlar. Etmenler bir segmente geldiğinde, gideceği hedef segment belirlenir. Devamı aynı yol olan segmentlerde, hedef segment tek iken; devamı farklı bir yol ise hedef segment birden fazla olabilir. Bu durumda etmenin gideceği segment, hedef olabilecek segmentler arasından rastgele

seçilir. Örneğin etmen sağa dönülmez trafik işaretinin bulunduğu bir segmentte ise ve birden fazla hedef segment var ise bu durumda sağa dönülünce ulaşılacak segment hedef segmentler arasından çıkarılır ve rastgele seçim işlemi bundan sonra gerçekleştirilir. Sağa dönülmez işaretine benzer şekilde hız sınırlayıcı, diğer dönülmez işaretleri, durma – duraklama işaretleri, v.b. trafik işaretlerinde de ilgili segmentin kaynak verisi kullanılarak (roads.xml'den alınmış) etmenlerin trafik işaretlerine uyumları sağlanır.

5.3.3.4. Etmen Davranışlarının Modellenmesi

Günlük hayatta trafikte sürücülerin aynı davranışları gösterdiklerini söylemek imkânsızdır. Kimi sürücüler kurallara uyar, kimileri uymaz olarak genellenebilecek bu sürücü davranışları trafik karmaşasının en önemli nedenlerindendir. Sürücü adayı, sürücü belgesini almadan yaptığı uygulama çalışmalarını trafik akışının olmadığı, özellikle yukarıda tanımlanan sürücü davranışlarının sergilenmediği alanlarda gerçekleştirirler. Fakat sürücü belgesini aldıktan sonra artık gerçek trafik ortamında taşıt süreceklerdir ve güvenli bir şekilde taşıt sürmek için bu tipten davranışlara hazırlıklı olmaları gerekir.

Bu maksatla TSS'ne sadece kurallara uyan vekillerin yanında, bazı olumsuz davranış gösteren vekillerde eklenmiştir. TSS yazılımında vekiller aşağıdaki tiplerde davranırlar:

1. Normal
2. Agresif (agressive)
3. Kural dışı (nonobservant)

Davranış tipi "Normal" olarak belirlenen bir vekil, TSS yazılımında tanımlanan bütün görsel ve görsel olmayan trafik kurallarına uyar. Örneğin segmentlerde hız sınırlamalarını uygun hızda hareket ederler, trafik ışıklarına uyarak geçiş yaparlar, takip mesafesini güvenli takip mesafesi ölçüsünde bırakırlar gibi.

Davranış tipi “Agresif” olarak belirlenen bir vekil, TES yazılımında tanımlanan bazı trafik kurallarına uygun bir biçimde hareket ederken, bazı kurallar için farklı davranış sergiler. Agresif vekillerin trafik kurallarına aykırı davranış gösterdikleri durumlar ve bu durumlardaki davranışları şöyle sıralanabilir:

1. Görsel olmayan trafik kuralı olarak belirlenmiş hız sınırlaması bulunan segmentlerde bu kuralın dışına çıkmak,
2. Trafik ışığının sarı yanması durumunda; taşıt hareket ediyorsa hareket devam ettirilir, taşıt duruyorsa harekete başlanır,
3. Taşıt takibi yaparken güvenli takip mesafesini kısa ayarlamak. Normal bir vekil için güvenli takip mesafesi yavaşlama mesafesi (vekil taşıtın hızının yarısına eşittir) ile güvenlik mesafesi (takip edilen taşıtın uzunluğu) değerlerinin toplamı iken, agresif vekil için bu takip mesafesi, güvenli takip mesafesinin yarısı kadardır,
4. Kavşağa yaklaşırken taşıtın hızını azaltmamak. Normal vekil trafik ışık kontrollü veya kontrolsüz bir kavşağa yaklaşırken gerektiğinde durabilmek için hızını azaltırken, agresif vekil trafik ışık kontrollü kavşaklarda hızını azaltır, fakat kontrolsüz kavşaklarda kavşaktan geçecekmiş gibi hızını azaltmadan hareketine devam eder.

Davranış tipi “Kural dışı” olarak belirlenen bir vekil, TSS yazılımında tanımlanan bir çok trafik kuralına aykırı davranış sergiler. Kural dışı vekillerin trafik kurallarına aykırı davranış gösterdikleri durumlar ve bu durumlardaki davranışları şöyle sıralanabilir:

1. Görsel veya görsel olmayan trafik kuralı olarak belirlenmiş hız sınırlaması bulunan segmentlerde bu kuralın dışına çıkmak,
2. Trafik ışığının sarı yanması durumunda; taşıt hareketine devam eder,
3. Trafik ışığının kırmızı yanması durumunda; taşıt hareketine devam eder,
4. Taşıt takibi yaparken güvenli takip mesafesini kısa ayarlamak. Bu durumdaki agresif vekil ile aynı davranışı gösterir. Yani takip mesafesi, yavaşlama mesafesi ile güvenlik mesafesi değerlerinin toplamının yarısı kadardır,

5. Kavşağa yaklaşırken taşıtın hızını azaltmamak. Bu durumdaki agresif vekil ile aynı davranışı gösterir. Yani kavşaklarda hızını azaltmadan hareketine devam eder,
6. Görsel trafik kuralarına (park edilmez, sağa dönülmez, sola dönülmez, u dönüşü yapılmaz) aykırı davranış sergilerler.

TSS yazılımı yapay zekâ modülünde, vekiller oluşturulurken belirlenen özelliklerine göre, vekilin taşıt sürüş davranışları kontrol edilir. Kontrol işlemi vekil taşıtın davranış özelliği (ki bu özellik ancak normal, agresif ve kural dışı davranış tiplerini içerebilir) kontrol edilerek gerçekleştirilir. Vekil taşıtların şerit takibi, taşıt takibi, trafik kurallarına uyum durumları gibi aktiviteleri davranış tiplerine göre farklılık gösterirler. Davranış tiplerine göre vekillerin simülasyon ortamında trafik kurallarına gösterecekleri davranışlar şöyle sıralanabilir:

1. Hız sınırlaması;
    a) Normal ise bütün hız sınırlama kurallarına uy!
    b) Agresif ise görsel olmayan hız sınırı kurallarına uyma!
    c) Kural dışı ise herhangi bir hız sınırlama kurallına uyma!
2. Trafik ışığının sarı yanması durumu;
    a) Normal ise dur!
    b) Agresif ise vekil hareket ediyorsa harekete devam et, vekil duruyorsa harekete başla!
    c) Kural dışı ise harekete devam et!
3. Trafik ışığının kırmızı yanması durumu;
    a) Normal ise dur!
    b) Agresif ise dur!
    c) Kural dışı ise harekete devam et!
4. Güvenli takip mesafesi;
    a) Normal ise yavaşlama mesafesi ile güvenlik mesafesi değerlerinin toplamıdır.
    b) Agresif ise yavaşlama mesafesi ile güvenlik mesafesi değerlerinin toplamının yarısıdır.

c) Kural dışı ise yavaşlama mesafesi ile güvenlik mesafesi değerlerinin toplamının yarısıdır.

5. Kavşağa yaklaşırken taşıtın hızı;
    a) Normal ise azalt!
    b) Agresif ise trafik ışık kontrollü kavşaklarda azalt, kontrolsüz kavşaklarda harekete devam et.
    c) Kural dışı ise kavşaklarda harekete devam et!
6. Görsel trafik kuraları;
    a) Normal ise kurallara uy!
    b) Agresif ise kurallara uy!
    c) Kural dışı ise kurallara uyma!

5.4. RAPORLAMA

Sürücü adayının her bir test sürüşü sonunda performansı test sonuçları veritabanında saklanmaktadır. Bu performansların veritabanında tutulmasının sebepleri aşağıdaki gibi sıralanabilir:

1. Sürücü adaylarının kendi bireysel performanslarını inceleyerek, hatalı davranışlarını tespit etmelerini sağlamak,
2. Sürücü adaylarının toplam test performanslarını görmelerini sağlamak,
3. Sürücü adaylarına, kendi performansları ile diğer sürücü adaylarının performanslarını mukayese imkânı vermek,
4. Eğitimcilerin, sürücü adaylarının performanslarını incelemelerini ve bu sayede doğru geri bildirim eğitimleri vermelerini sağlamak,
5. Sürücü adaylarının en çok yaptıkları hataları anlamalarını sağlamaktır.

5.4.1. Veritabanı İşlemleri

TSS yazılımı test modülünde çalıştırıldığında, sürücünün yaptığı hatalar geri bildirim yapılmadan, arka planda kaydedilir. Test sürüşü bittiğinde bu hatalar sürücüye yazılı bir rapor şeklinde bildirilir. Bununla birlikte test sürüş raporu, test sonuçları veri tabanında sürücü adı ile oluşturulacak bir tabloya kaydedilir. Hata rapor tablosunda

sürücünün hata kodları, hata açıklamaları, test sürüş tarihi ve test sürüş zamanı alanları vardır. Sürücünün değişik zamanlarda yaptığı test sürüşleri de bu tabloya güncel verileri ile kaydedilir. Böylece sürücünün yaptığı her test sürüş performansı, kendi adı ile oluşturulan tabloya hata kodu, hata açıklaması, test tarihi ve zamanı bilgileri ile kaydedilir.

Test sonuçları MS Access veri tabanında tutulmaktadır. TSS yazılımı, veritabanı ile yapılacak işlemlerde OleDb kütüphanesinin nesnelerini kullanmaktadır. Veritabanı üzerinde yapılacak işlemler için SQL sorgulama dilinden faydalanılmıştır. Şekil 5.15'de sürücülerin test performanslarının, test sonuçları veritabanına kaydedilme işlemlerinin kısa kodları verilmiştir.

```
Eğer(!(tablo(<surucu_ad>)))
    baglan(test_sonuc_vt)
    TabloOlustur(sorgu(ad←<surucu_ad>, alan_ad ←<hata_dizi_indis>,<test_zaman>)
    baglantıyıkapat(test_sonuc_vt)
baglan(test_sonuc_vt)
AlanEkle(sorgu (tablo(<surucu_ad>), (veri ←<hata_dizisi>,<test_zaman>))
baglantıyıkapat(test_sonuc_vt)
```

Şekil 5.15. Sürücü performanslarının veritabanına kaydedilmesi.

TSS yazılımında, sürücü adayı kullanıcı ara yüzünde test modülü ile bağlantılı parametrik değerleri belirlerken kullandığı isim parametresi, sürüş bittikten sonra test performansının kaydedileceği tablonun ismi olacaktır. Test esnasında sürücünün yaptığı hatalar, hata kodu ve hata açıklamaları ile bir hata dizisine kaydedilir. Test sürüşü tamamlandığında, sürücü adında bir tablonun test sonuçları tablosunda varlığı kontrol edilir. Bu isimde bir tablo yoksa OleDb kütüphanesinde bulunan OleDbConnection ve OleDbCommand nesneleri, SQL dili ile hazırlanmış sorgularla birlikte kullanılarak, test sonuçları tablosunda, sürücü adında ve alan isimleri hata kodu, hata açıklaması, test tarihi ve test zamanı olan bir tablo oluşturulur. Tablo oluşturulduğunda veya tablonun varlığı tespit edildiğinde yine OleDb kütüphanesinde bulunan OleDbConnection ve OleDbCommand nesneleri, SQL dili ile hazırlanmış sorgularla birlikte kullanılarak, ilgili tablo alanlarına elde edilen test sonuçları eklenir. Bu tablonun hata kodu ve hata açıklaması alan verileri, hata

dizisinden temin edilirken, o an ki sistem zaman bilgisi test tarih ve test zaman alan verileri için kullanılır. Sürücünün farklı zamanlarda yaptığı test sürüşlerindeki her bir performansı kendi adında açılan tablo üzerine eklenerek kaydedilir.

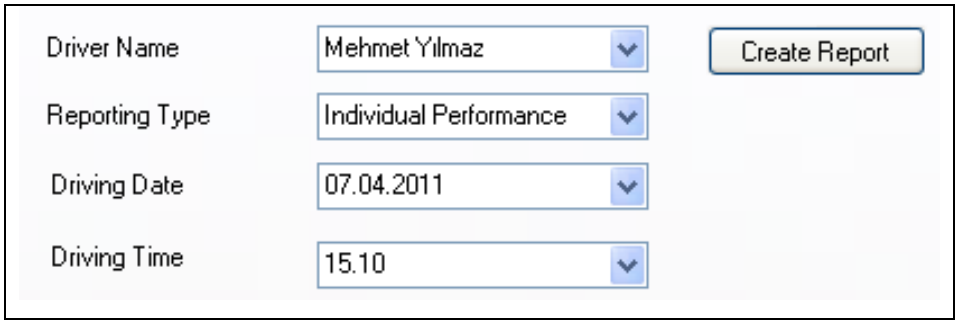
Şekil 5.16. Test modülü için TSS kullanıcı ara yüzü.

5.4.2. Raporlar

Sürücülerin gerçekleştirdikleri test sürüşlerinde gösterdikleri performanslar sürücü hata veritabanında tutulmaktadır. Sürücü test performanslarının anlamlı bir şekilde görüntülenebilmesi için TSS yazılımında, CR raporları kullanılmıştır. Raporlara aktarılan veriler test sonuçları veritabanından sorgulanarak elde edilmektedir. Sorgulama işlemleri için OleDb kütüphanesi nesneleri ve SQL veritabanı sorgulama dilinden istifade edilmiştir. Rapor edilecek veri kümesi oluşturulurken aynı algoritma basamakları farklı sorgular ile çalıştırılarak elde edilir. Bu algoritma aşağıdaki gibi özetlenebilir.

Öncelikle OleDbConnection nesnesi ile test sonuçları veritabanı ile bağlantı kurulur. OleDbDataAdapter nesnesi, SQL sorgusu kullanılarak raporlanacak veri belirlenir. Raporlama için kullanılan Crystal Reports nesnesi veri kaynağı olarak DataSet nesnelerini kullanır. Bu yüzden OleDbDataAdapter sorgusu ile elde edilen veriler ile DataSet nesnesi doldurulur ve Crystal Report nesnesinin veri kaynağı bu DataSet

yapılır. Son olarak rapor, rapor görüntüleyiciye aktarılarak istenilen test raporunun ekrana dökülmesi gerçekleştirilir. İstenirse raporun çıktısı alınabilir veya farklı formatlarda (pdf gibi) manyetik ortamlara kaydedilmesi gerçekleştirilebilir.



Şekil 5.17. Rapor oluşturma ara yüzü.

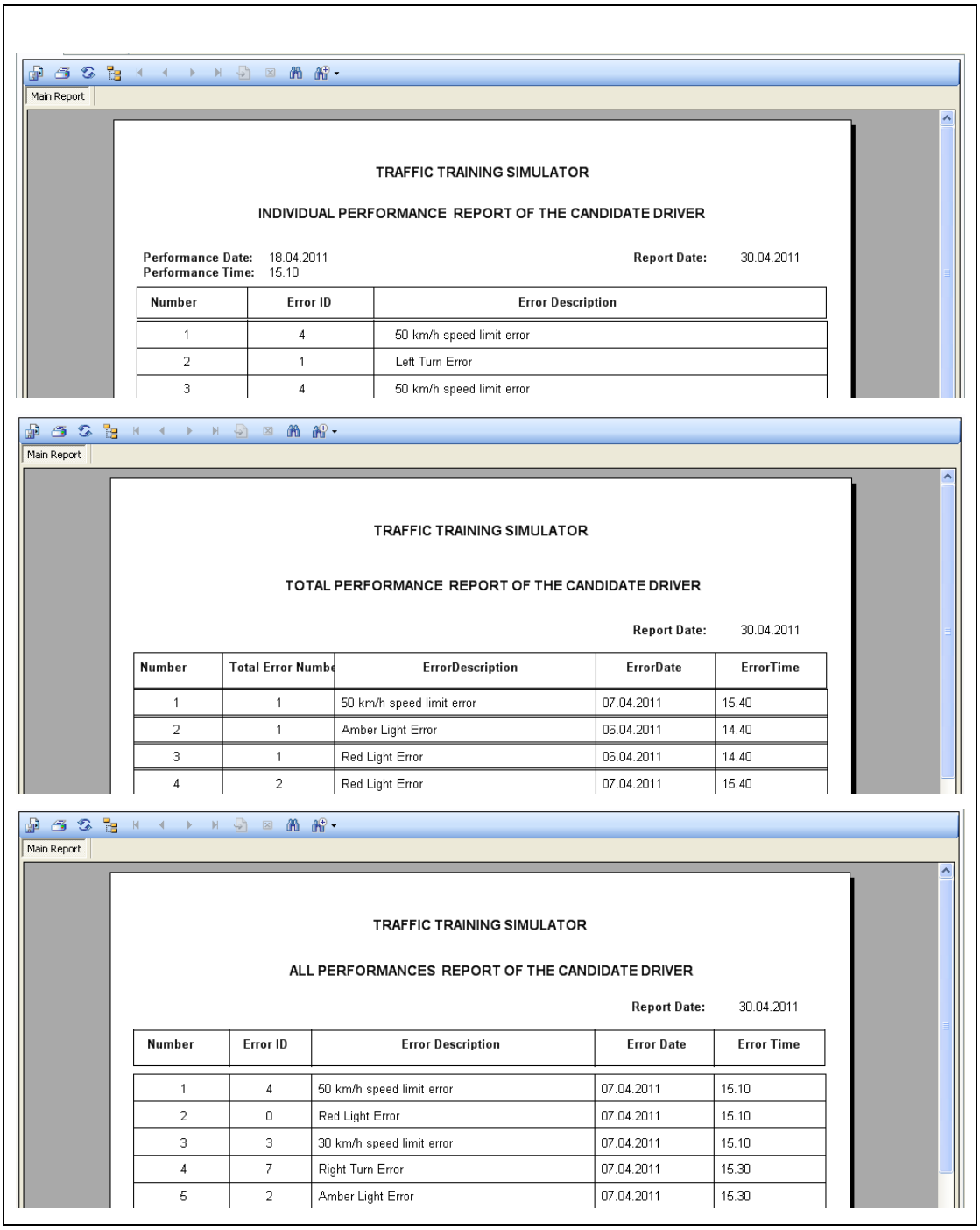
TSS yazılımında veritabanı raporları Şekil 5.17'de gösterilen ara yüz kullanılarak oluşturulur. Sürücü ismi için ayrılan alan, TSS yazılımı ile test sürüşü yapmış olan bütün sürücü adaylarının isimleri ile doldurulur. Zira bu veri test sonuçları veritabanında yer alan tablo isimlerinden ibarettir ve SQL sorguları ile ilgili veriler, test sonuçları veri tabanında çekilir.

TSS yazılımı üç farklı tipte rapor üretmektedir:

1. Bireysel Performans (Individual Performance): Raporlama ara yüzünde "Rapor Türü" (Report Type) *Bireysel Performans* seçildiğinde "Sürüş Tarihi" (Driving Date) alanı görünür ve bu alan sürücü isminde belirtilen değerdeki sürücü adayının kaydedilen bütün test tarihleri ile doldurulur. Bu tarih değerleri sürücü ismindeki tabloda yer alan tarih değerlerinin SQL sorguları kullanılarak çekilmesiyle elde edilir. "Sürüş Tarihi" alanından bir tarih seçildiğinde ise "Sürüş Zamanı" (Driving Time) alanı görünür ve bu alan ilgili sürücünün, belirtilen tarihte yaptığı test sürüşleri zamanları ile doldurulur. Bu zaman değerleri sürücü ismindeki tabloda, belirtilen tarihte kaydedilmiş zaman verisi, SQL sorguları kullanılarak test sonuçları veritabanından çekilmesi ile elde edilir. "Rapor Oluştur" düğmesi işaretlendiğinde ilgili sürücünün, belirtilen tarih ve zamanda gerçekleştirdiği test sürüşünün sonuçları rapor olarak dökülür.

2. Toplam Performans (Total Performance): Bu raporlama türünde ilgili sürücünün yaptığı test sürüşlerinde aynı hataları tekrarlama sayıları rapor olarak dökülmektedir. Rapor için gerekli veri kümesi SQL sorgusu ile elde edilir.

3. Bütün Performanslar (All Perfonmances): Bu raporlama türünde ilgili sürücünün yaptığı bütün test sürüşleri filtrelenmeden rapor olarak dökülmektedir. Rapor için gerekli veri kümesi SQL sorgusu ile elde edilir.



Şekil 5.18. Rapor örnekleri.

TSS raporları kullanılarak aday sürücüler ve eğitimciler, ilgili sürücünün test sürüş yeterliliklerini kontrol edebilirler.

BÖLÜM 6

DENEYSEL BULGULAR

Ülkemizde sürücü eğitimi MEB'na bağlı olarak hizmet veren özel sürücü kurslarında verilmektedir. Sürücü kurslarında verilen eğitim, teorik eğitim ve direksiyon eğitimi bölümlerini içerir. Teorik eğitim de sürücü adaylarına trafik, motor ve ilk yardım başlıklarında taşıt sürerken bilmeleri gereken bilgiler verilir. Direksiyon eğitiminde ise aday sürücünün güvenli sürüş becerilerini kazanması için test sürüşleri gerçekleştirilir.

Hazırlanan TSS yazılımı, MEB'na bağlı özel bir sürücü kursunda sürücü eğitimi almakta olan 24 kişilik bir grup tarafından denenmiştir. Grupta yer alan sürücü adaylarından bir kısmı daha önce direksiyon eğitimi almamış, bir kısmı ise direksiyon eğitimine yeni başlamıştır.

6.1. UYGULAMA DENEYİ

Sürücü adayları TSS uygulama deneyi, beş aşamadan oluşmaktadır:

1. Oryantasyon: Bu aşamada sürücü adaylarının TSS yazılımına intibak etmeleri amaçlanır. Aday, TSS kullanıcı ara yüzünden Oryantasyon modunda ve belirlediği diğer simülasyon seçenekleri ile hazırlanmış trafik ortamında taşıt sürme faaliyeti gerçekleştirir. Taşıt sürerken gaz, fren pedalları ve direksiyon donanımlarını kullanır. Direksiyon üzerinde yer alan sinyal (sağ veya sol), korna, vites gibi kontrolleri öğrenir ve bu kontrollere kullanmada alışkanlık kazanır. Oryantasyon aşamasının süresi, sürücü adaylarının bireysel farklılıklarına göre değişir.

2. Test1: Sürücü adayı oryantasyon eğitimini tamamladıktan sonra test modunda taşıt sürme faaliyeti gerçekleştirir. Sürücünün bu aşamada test edilmesinin sebebi, bir sonraki aşamada yapılacak olan eğitim faaliyetlerinin etkisini görmektir. Aday belirlediği parametrelerde oluşturulan trafik ortamında test sürüşü gerçekleştirir. Adayın test esnasında yaptığı hatalar, arka planda veri tabanına kaydedilir.

3. Eğitim: Test1 aşamasından sonra aday sürücü eğitim modunda taşıt sürme faaliyeti gerçekleştirir. TSS, hız sınırlaması, trafik ışığı, sinyal, dönüş, durma, duraklama, şerit ihlali vb. trafik kurallarını kontrol edebilmektedir. Aday eğitim modunda taşıt sürerken yaptığı trafik kuralı ihlalleri, yazılı ve sesli mesajlar ile geri bildirim olarak adaya iletilir. Eğitim modunda hedef, adayın trafik ortamında karşılaştığı trafik kurallarının anlamlarını öğrenmesini ve bu kurallara uyma alışkanlığını kazanmasını sağlamaktır.

4. Test2: Bu aşama TSS uygulama deneyinin son basamağıdır. Eğitimini tamamlayan sürücü adaylarına test modunda tekrar bir test sürüşü uygulanır. Burada amaç TSS tarafından verilen eğitimin, sürücüye olan katkılarını gözlemlemektir. Adayın test sürüşünde yaptığı trafik ihlalleri, daha sonra değerlendirilebilmesi için arka planda veri tabanına kaydedilmektedir.

5. Anket: TSS uygulamasını tamamlayan sürücü adaylarına, uygulamayı değerlendirmeleri istenen bir anket uygulanmıştır (Şekil 5.19). Değerlendirme anketi sistemin özelliklerini irdeleyen puanlama soruları ve adayın kendi düşüncesini yazabileceği yorum bölümünden oluşmaktadır. Ankette yer alan sorular, 1 ila 5 kademesindeki puanlar ile değerlendirilmektedir. Bu sorular ile TSS'in şehir ortamı, mesaj sistemi, trafik kuralları, sürüş modları, giriş - çıkış donanımları, otonom hareket eden vekil taşıt davranışları, sistemin gerçeğe yakınlığı gibi özellikleri puanlandırılmaktadır.

Trafik Simülasyon Sistemi Yazılımı (TSS) Sürücü Eğitim Anketi

Adı Soyadı :

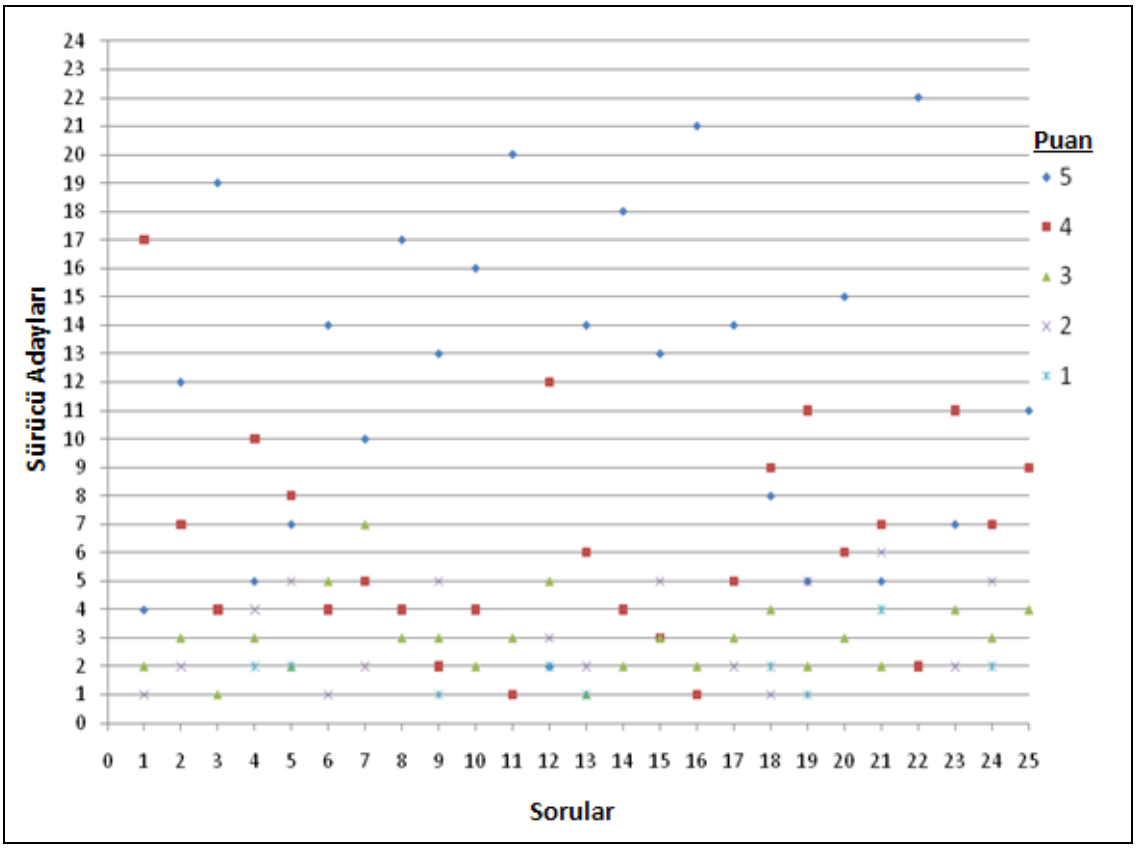
Yaş :

Soru No	Soru	Puanlama				
		5	4	3	2	1
1	Sürüş pisti uzunluğu yeterli mi?					
2	Yazılı geri bildirim mesajları trafik kurallarını öğrenmemde yardımcı oldu?					
3	Geri bildirim mesajlarının sesli olması kuralları öğrenmemi kolaylaştırıyor?					
4	Trafik işaretleri yeterince görsel mi?					
5	Sürüş manevraları gerçeğe yakın mı?					
6	Trafik işaretleri öğrenmemi kolaylaştırıyor?					
7	Test süresi kendimi değerlendirmem için yeterliydi?					
8	Dönüş işaretlerinin anlamını öğrendim?					
9	Park etme işaretlerinin anlamını öğrendim?					
10	Hız limit işaretlerinin anlamını öğrendim?					
11	Şehir içi ve şehirlerarası hız limitlerini öğrendim?					
12	Sinyal verme alışkanlığını kazandım?					
13	Hız azaltılması için yapılması gereken sürücü davranışlarını öğrendim?					
14	Farklı araç tipleri sürme farklılığını hissettim?					
15	Kıvrımlı yolları gösteren trafik işaretlerini öğrendim?					
16	Aracın hızını ayarlamada vites kullanımının gerekliliğini kavradım?					
17	Test sürüşü sonunda verilen hata raporu hatalarımı düzeltmeme yardımcı oldu?					
18	Etmenlerin şerit takibi davranışı gerçekçidir?					
19	Etmenlerin araç takibi davranışı gerçekçidir?					
20	Etmenlerin trafik işaretlerine uyumları gerçekçidir?					
21	Etmenlerin kavşak davranışları gerçekçidir?					
22	Etmenlerin trafik ışıklarına uyumları gerçekçidir?					
23	Farklı sürücü tipleri TSS'nin gerçekçiliğini arttırmıştır?					
24	Direksiyon ve gaz pedalları TSS'nin gerçekçiliğini arttırmıştır?					
25	Görüntü platformu TSS'nin gerçekçiliğini arttırmıştır?					
Yorumlar:						

Şekil 6.1. Anket formu.

6.2. ANKET SONUÇLARI

Sürücü adaylarına uygulanan değerlendirme anketi sonuçları Şekil 6.1'de yer alan grafikte gösterilmiştir. Grafiğin sütunları ankette yer alan soru numaralarını, satırları aday sayısını, renkli madde işaretleri ise ilgili sorulara aynı puanı veren aday sayısını göstermektedir.



Şekil 6.2. Test değerlendirme anketi sonuçları.

Anket sorularına verilen cevaplar incelendiğinde aşağıdaki sonuçlar elde edilmiştir.

1. Adaylar % 63-88 oranında trafik kurallarının öğrenilme durumu, görsel öğelerin gerçekçiliği ve trafik eğitim alanının yeterliliği için TSS'e 4 veya 5 puan vermişlerdir.
2. Adaylar sesli mesajların yazılı mesajlardan daha etkin öğrenme sağladığı kanısındadırlar.

3. Adaylar sürüş manevralarının gerçekçiliğiyle alakalı verdikleri cevaplar incelendiğinde % 63'ü 4 veya 5 puan verirlerken, % 30 oranında 1 veya 2 puan vermişlerdir.
4. Adayların TSS'de yer alan vekil taşıtlar ile ilgili sorulan sorulara verdikleri cevaplar incelendiğinde;
    a) Adaylar, etmenlerin trafik ışıklarına % 100, trafik işaretlerine % 88 oranlarında uyum durumlarına 4 ve 5 puan verdikleri görülmektedir.
    b) Şerit takibi için % 70, araç takibi için % 67 ve kavşak geçişleri için ise % 50 oranlarında 4 ve 5 puan verildiği de gözlemlenmektedir.
    c) Adayların etmen davranışları ile ilgili 1 ve 2 puan verdikleri araç takibi için oran % 25 iken kavşak geçişi için % 42'dir.
5. Adaylar sistemde yer alan farklı davranış tipindeki (normal, agresif ve kuraldışı) vekil taşıtların gerçekçiliği sorusuna % 75 oranında 3, 4 veya 5 puan vermişlerdir.
6. Adaylar görüntü platformunun gerçeğe yakınlığı için % 83 oranında, direksiyon ve pedal donanımlarının gerçeğe yakınlığı için % 54 oranında 4 veya 5 puan vermişlerdir.

Adaylar değerlendirme anketinin yorumlar bölümünde etmenlerin gerek yazılı trafik kurallarına uyarken, gerekse de şerit takibi gibi diğer sürüş becerilerinde sürekli aynı davranışları gösteren vekil taşıt sayısını çokluğundan yakınmışlardır. Gerçek trafik ortamında bulunan farklı tipteki sürücü davranışlarını gösteren taşıt sayısının artması gerektiğini vurgulamışlardır.

Elde edilen bulgular değerlendirildiğinde, aday sürücülerin eğitim uygulanmadan yapılan test değerlendirmesine göre trafik kurallarına uyum durumları, eğitim aldıktan sonra pozitif yönde arttığı gözlemlenmiştir. Daha fazla eğitim ve test yapan sürücü adaylarının değerlendirme raporları incelendiğinde ilk zamanlara göre son zamanlarda iyileşmeler gözlemlenmiştir. Bir başka deyişle adaylar TSS eğitiminden sonra yapılan test de daha az sayıda hata yapmışlardır.

BÖLÜM 7

SONUÇLAR VE ÖNERİLER

Bu çalışmada, sürücü adaylarına sürücü eğitimi vermek için TSS (Trafik Simülasyon Sistemi) isminde kural tabanlı bir platform geliştirilmiştir. Geliştirilen platform yazılım ve donanım bileşenlerinden oluşmaktadır. Yazılım bileşeninde 3B bilgisayar grafikleri kullanılarak sanal bir trafik ortamı oluşturulmuştur. Sürücü adayı geliştirilen bu sanal ortamda donanım aygıtlarını kullanarak taşıt sürme faaliyetini gerçekleştirebilir.

TSS geliştirilirken, vekil taşıtların oluşturulması, davranış türlerini belirlenmesi ve simülasyon ortamının oluşturulması işlemlerinde mikro simülasyon modeline sadık kalınmıştır. Trafik simülatörlerinde vekil taşıtların kontrolü için hücresel otomat, uzman sistemler, sonlu durum makineleri, hiyerarşik eş zamanlı durum makineleri, bulanık mantık, etmen tabanlı modeller kullanılmaktadır. TSS vekil taşıtların trafik hareketlerini eş zamanlı durum makineleri kullanarak sağlamıştır. Vekil taşıtların şerit takibi, taşıt takibi, v.b. gibi davranışları HEZDM'nin alt durumları olarak tasarlanarak her bir etmen için bir HEZDM kullanılmıştır. Otonom hareket eden vekillerin farklı davranışlar sergilemeleri için etmen tabanlı modelleme kullanılmıştır. Trafik ışıklarının çalışma frekansları modellenirken belirli son durumlu makinelerden faydalanılmıştır.

TSS yazılım bileşeni, MSVS editöründe, C++ 9.0 programlama dili ile Delta3d kütüphanesi kullanılarak hazırlanmıştır. Simülasyon ortamı Open Scene Graph sahne grafı ile oluşturulmuştur. Simülatörlerde şehir bilgisi için 3B modeller veya CBS kullanılmaktadır. Temin edilen CBS verilerinin düşük çözünürlükte olması sebebiyle enterpolasyon gerekliliği probleminden dolayı TSS'de, şehir verisi FLT 3B modelinden elde edilmiştir. Taşıt bilgileri ise 3B 3DS modelinden temin edilmiştir.

TSS'de yer almayan yeni taşıtlar, 3B modellerin yüklenmesi yolu ile TSS sistemine eklenebilir.

Taşıtları hareketlendirmek ve manevra kabiliyetleri vermek için Open Dynamics Engine fizik kütüphanesi kullanılmıştır. Sistemde hareket eden taşıtlar hareket dinamiği kurallarına uygun olarak davranış göstermektedirler.

Simülasyonda sesli mesajlar ve taşıta ait sesleri oynatma işlemleri için OpenAL kütüphanesi kullanılmıştır. Direksiyon ve pedal donanımlarının kullanımı için PLIB kütüphanesinden faydalanılmıştır. Sürücü adayı direksiyon, gaz ve fren pedalları donanımlarını kullanarak taşıtını sürerler.

TSS, kullanıcıların yaptıkları test sürüşlerinde gösterdikleri performansları veritabanına kaydetmektedir. Kullanıcılar ve eğitimciler veritabanında tutulan test sürüş verilerine OleDb ve CR kütüphanelerini kullanarak raporlayabilmektedirler.

TSS yazılımında, sürücü adayı kontrolündeki taşıt ve otonom hareket eden vekil taşıtlar olmak üzere iki tür taşıt tipi vardır. Kullanıcılar sürücü adayı kontrolündeki taşıt ile görüntü sistemi ve kumanda donanımlarını kullanarak, sanal olarak oluşturulan 3B trafik ortamında sürüş faaliyetlerini gerçekleştirebilirler. Otonom hareket eden vekil taşıtlar şerit takibi, taşıt takibi, trafik kurallarına uyum, yavaşlama, hızlanma, kavşak geçişleri gibi güvenli sürüş becerileri ile donatılmıştır. Trafikte karşılaşılabilecek normal, agresif ve kural dışı davranacak sürücü davranış modelleri, vekil sürücü davranış türü olarak vekillere atanabilmektedir. Vekil taşıtlar trafikte ilerlerken yolun kıvrımlı veya düz olmasına göre manevra yapabilmektedirler.

Sistem sürücü adaylarına 3B bir simülasyon ortamında, direksiyon ve pedal donanımları kullanarak, üç monitörlü görüntü platformu üzerinden gerçeğe yakın görüş açıları ile taşıt sürme olanağı sağlar. Sürücü adayı taşıt sürerken, önündeki, arkasındaki, sağ yanındaki ve sol yanındaki olup biten olayları görüntü platformundan takip edebilmektedir. Görüntü platformunda oluşturulan görüntüler, 2 ekran kartı ve 3 monitör üzerinde sahne grafı teknikleri kullanılarak oluşturulmuştur.

Simülasyon ara yüzü kullanılarak hava koşulları, simülasyon özellikleri, taşıt tipleri parametrik olarak ayarlanabilir.

Sürücü adayları oryantasyon, eğitim ve test modlarında taşıt sürebilirler. Oryantasyon modunda amaç sürücü adayının görüntü platformuna, direksiyona, pedallara ve simülasyon öğelerine alışmasına yardımcı olmaktır. Eğitim modunda geri bildirim mesajları, trafik kurallarının aday sürücüye öğretilmesi için, yazılı ve sesli olarak bildirilmektedir. Test modülünde ise geri bildirim verilmeden sürücü adayının belirli bir süre test sürüşü yapması sağlanmaktadır. Kullanıcıların farklı zamanlarda yaptıkları testlere ait sonuçlar test sonuçları veritabanına kaydedilmektedir. TSS raporları kullanılarak bu raporlar irdelenebilmektedir.

TSS sisteminin, benzer simülasyon sistemleri ile karşılaştırıldığında elde edilen sonuçlar Çizelge 7.1'de gösterilmiştir. TSS, kendinden önce geliştirilen trafik simülatörlerin genel özelliklerini kapsayacak biçimde hazırlanmıştır. Ayrıca bu temel özelliklerin yanında farklı sürücü tipleri, görüntü platformu, mesaj sistemi, raporlama, v.b. özellikler ile de donatılmıştır.

Çizelge 7.1. Trafik simülasyon sistemlerinin karşılaştırılması.

Özellik	CORSIM	SIMTRAFFIC	VISSIM	HUTSIM	PARAMICS	AIMSUN	WATSIM	TRAFİKENT	TSS
Nesne Yönelimli Mimari	-	?	+	+	+	+	+	+	+
Taşıt Takibi	+	+	+	+	+	+	+	+	+
Şerit Takibi	+	+	+	+	+	+	+	+	+
Takip Mesafesi	+	+	+	+	+	+	+	+	+
Dönüş Hızı	-	+	+	-	-	-	-	-	+
Farklı Sürücü Tipleri	-	-	-	-	-	-	-	*	+
Sarı Işık Tepkisi	*	*	*	?	*	*	+	+	+
Hızlanma / Yavaşlama Katsayısı	+	+	+	+	+	+	+	+	+
Görüş Uzaklığı Ayarı	+	?	+	?	-	+	?	-	+
Manevra Hatalarının Kaydedilmesi	+	+	+	+	+	+	+	+	+
Dönüş Sinyallerinin Modellenmesi	-	?	-	?	-	-	-	+	+
Kavşak Kontrolü	+	+	+	+	+	+	+	+	+
3B	-	-	+	-	+	+	+	+	+
Görüntü Platformu	-	-	-	-	+	-	-	*	+
Pedal	-	-	+	-	+	+	+	+	+
Direksiyon	-	-	+	-	+	+	+	+	+
Mesaj Sistemi	-	-	-	-	?	-	?	*	+
Raporlama	-	-	-	-	?	-	?	-	+
Farklı Çalışma Modları	-	-	-	-	-	-	-	-	+

(+) : Var
(-) : Yok
(?) : Bilinmiyor
(*) : Kısmen veya davranış duyarlı

KAYNAKLAR

1. Hirata, T., "Development of driving simulation system: Movic-t4 and its application to traffic safety analysis in underground urban expressways", Doktora Tezi, ***Tokyo Institute of Technology Department of Civil Engineering***, Tokyo, Japan, 2-3, 20-22, 157-160 (2005).

2. Das, S., Bowles, B. A., Houghland, C. R., Hunn, S. J. and Zhang, Y. "Microscopic simulations of freeway traffic flow,", ***Thirty-Second Annual Simulation Symposium***, San Diego, USA, 79 (1999).

3. Yongwu, M., Ulrich H. and Niels P., "Naughty agents can be helpful: training drivers to handle dangerous situations in virtual reality", ***Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT'06)***, IEEE Computer Society, 735-739 (2006)

4. Sümer, N., "Trafik kazalarında sosyal psikolojik etmenler: sürücü davranışları, becerileri ve sosyal politik çevre", ***Türk Psikoloji Yazıları***, 5 (9-10): 1-36 (2002).

5. Yasak, Y., "Trafik psikolojisi", ***TSOF-PSIKOTEKNIK: Sürücü Değerlendirme Eğitim ve Araştırma Merkezi***, Ankara, 3-11 (2002).

6. Evans, L., "Traffic safety", Science Serving Society, ***Bloomfield Hills***, MI, 32-45 (2004).

7. Demir, M., "Trafik eğitimi için zeki etmenler geliştirme", Doktora Tezi, ***Gazi Üniversitesi Bilişim Enstitüsü***, Ankara, 3-18 (2008).

8. Sümer, N. ve Ünal, A.B., "Acemi sürücülerde tehlike algısı becerisi", ***Üçüncü Trafik ve Yol Güvenliği Ulusal Kongresi***, Ankara, 177-182 (2005).

9. Ece, H., "Trafik kazaları özeti", ***T.C. Bayındırlık ve İskan Bakanlığı Karayolları Genel Müdürlüğü***, Ankara, 3-5 (2003).

10. Sümer, N., Ünal, A. B. ve Birdal, A., "Assessment of hazard perception latencies using real life and animated traffic hazards: Comparison of novice and experienced drivers", ***Proceedings of the Fourth International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design***, Iowa, USA, 488-494 (2007).

11. Pursula, M., "Simulation of traffic systems - An overview", ***Journal of Geografic Information and Decision Analysis***, 3 (1): 1-9 (1999).

12. Jakovljevic, G. and Basch, D., "Implementing multiscale traffic simulators using agents", ***26th Int. Conference on Information Technology Interfaces***, Croatia, 519-524 (2004).

13. Das, S., Bowles, B. A., Houghland, C. R., Hunn, S. J. and Zhang, Y. "Microscopic simulations of freeway traffic flow", ***Thirty-Second Annual Simulation Symposium***, San Diego, USA, 79 (1999).

14. Brooks, R. A., "A layered control system for a mobile robot", ***IEEE Journal of Robotics and Automation***, 2 (1): 14-19 (1986).

15. Maerivoet S. and De Moor B., "Cellular automata models of road traffic", ***Physics Reports***, 1 – 64 (2005).

16. Sun T., Wang J., "A traffic cellular automata model based on road network grids and its spatial and temporal resolution's influences on simulation", ***Simulation Modelling Practice and Theory***, 864–878 (2007).

17. McHaney, R., "Artificial intelligence and computer simulation", ***Proceedings of the 1997 Winter Simulation Conference***, Orlando, USA, 1118-1121 (1993).

18. Internet: Directorate General VII - Transport of the European Commission "SMARTEST - Review of Micro-Simulation Models", **http://www.its.leeds.ac.uk/projects/smartest/deliv3.html#a2** (1997).

19. Lee, B. and Lee, E. A., "Interaction of finite state machines and concurrency models", ***Proceeding of Thirty Second Annual Asilomar Conference on Signals, Systems, and Computers***, California, USA, 1715-1719 (1998).

20. Harel, D., "A visual formalism for complex systems," ***Science of Computer Programming***, 8: 231-274 (1987).

21. Lucas, P. J., "An object-oriented language system for implementing concurrent, hierarchical, finite state machines", Yüksek Lisans Tezi, ***University of Illinois***, Illinois, USA, 1-16, (1993).

22. Okutanoğlu, A. ve İşler, V., "Simülasyon sistemlerinde esnek senaryo altyapıları", ***Birinci Ulusal Savunma Uygulamaları Modelleme ve Simülasyon Konferansı***, Ankara, 115-126 (2005).

23. Cremer, J. and Papelis, Y., "The software architecture for scenario control in Iowa driving simulator", ***Proc. of 4th Computer Generated Forces and Behavioral Representation Conference***, Orlando, USA, 523-532 (1994).

24. Zadeh, L. A, "Fuzzy logic, neural networks, and soft computing", ***Communications of the ACM***, 37 (3): 77-84 (1994).

25. Gao, X. Z., "Soft computing methods for control and instrumentation", Doktora Tezi, ***Institute of Intelligent Power Electronics Publications***, Finland, 1-11 (1999).

26. Petropoulakis, L., “Intelligent control using agents and fuzzy behavioral structures”, ***Proceedings of the 15th IEEE International Symposium on Intelligent Control***, Greece, 389-393 (2000).

27. George G. and Cardullo, F., “Application of neuro-fuzzy systems to behavioral representation in computer generated forces”, ***Proc. 8 Conf. on Comp. Generated Forces***, Orlando, USA, 575-585 (1999).

28. Saffiotti, A., “Fuzzy logic in autonomous robotics: behaviour coordination”, ***Proceedings of the 6th IEEE International Conference on Fuzzy Systems***, Barcelona, Spain, 573-578 (1997).

29. Saniee, M. and Habibi, J., “A fuzzy ranking method for automated highway driving”, ***IEEE Intelligent Vehicles Symposium***, Parma, Italy, 218-221 (2004).

30. Franklin, S. and Graesser, A., “Is it an agent or just a program?: A taxonomy for autonomous agents”, ***Proceedings of the 3th International Workshop on Agent Theories Architectures and Languages***, Berlin, Germany, 21-35 (1996).

31. Russell, S. J. and Norvig, P., “Artificial intelligence: A modern approach”, ***Prentice Hall***, New Jersey, USA, 32-58 (2003).

32. Hill, R., Kim, Y. and Gratch, J., “Anticipating where to look: Predicting the movements of mobile agents in complex terrain”, ***Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems***, Italy, 821-827 (2002).

33. Wooldridge, M., “Agent-based software engineering”, ***IEEE Proceedings on Software Engineering***, 144 (1): 26-37, (1997).

34. Ehlert, P. A. M., “Intelligent driving agents: The agent approach to tactical driving in autonomous vehicles and traffic simulation”, Yüksek Lisans Tezi, ***Delft University of Technology Faculty of Information Technology and Systems***, Netherlands, 15-20 (2001).

35. Bauer, B., “Lecture notes in computer science; revised papers and invited contributions from the second international workshop on agent-oriented software engineering II”, ***Springer-Verlag***, London, England, 101-118 (2001).

36. Li, Y., Ma, S., Li, W. and Wang, H., “Microscopic urban traffic simulation with multi-agent system”, Communications and Signal Processing, ***Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information***, China, 1835 – 1839 (2003).

37. Gang C., Shang X., GuanQun J. and Quan D., “Scene simulation platform based on data fusion of multiple format 3d models”, ***IEEE International Conference on Computer Modeling and Simulation***, 342-346 (2009).

38. McDowell, P. “Delta3D and open source software”, ***Delta3D Defense GameTech Users Conference***, (2008).

39. Internet: Open Scene Graph, “Support”, **http://www.openscenegraph.org/ projects/osg/ wiki/ Support/**, (2010).

40. McDowell, P., Darken R., Sullivan J., Johnson E., “Delta3D: A complete open source game and simulation engine for building military training systems”, ***The Journal of Defense Modeling & Simulation***, 3 (3), 143–154 (2006).

41. Internet: Open Dynamics Engine, “Support”, **http://www.ode.org/support/**, (2010).

42. Internet: Open Dynamics Engine, “Open dynamics engine v0.5 user guide”, 13-14, 22-34, **http://ode.org/ode-latest-userguide.html** (2006).

43. Internet: Creative Labs, “OpenAL”, **http://connect.creativelabs.com/ openal/default.aspx**, (2010).

44. Internet: Sourceforge, “PLIB”, **http://plib.sourceforge.net/**, (2011).

45. Şahin İ., “Uzman sistem kullanarak iki boyutlu izdüşümlerden katı modeller oluşturma”, Doktora Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 66-72 (2008).

46. Internet: Mevzuat Bilgi Sistemi, “T.C. Karayolları Trafik Yönetmeliği”, **http://www.mevzuat.gov.tr/Metin.Aspx?MevzuatKod=7.5.8182&sourceXmlSearch=trafik&MevzuatIliski=0** (2010).

47. Kemeny, A. and Panerai, F., “Evaluating perception in driving simulation experiments”, ***Trends in Cognitive Sciences***, 7 (1): 31-37 (2003).

48. Archer, J. ve Kosonen, I., “The potential of micro-simulation modelling in relation to traffic safety assessment”, ***Proceedings of the ESS Conference***, Hamburg, 427-431 (2000).

ÖZGEÇMİŞ

İsmail KURNAZ 1978 yılında Ankara'da doğdu; ilk ve orta öğrenimini aynı şehirde tamamladı. 1997 yılında Gazi Üniversitesi Teknik Eğitim Fakültesi Bilgisayar Sistemleri Öğretmenliği Bölümü'nde öğrenime başlayıp 2001 yılında mezun oldu. 2001 yılında Beypazarı Ticaret Meslek Lisesi'nde bilgisayar öğretmeni olarak göreve başladı. Yenimahalle Meslek ve Anadolu Meslek Lisesi'nde bir yıl görev yaptı. 2003 yılında GÜ Fen Bilimleri Enstitüsü Bilgisayar Eğitim Anabilim Dalında yüksek lisans eğitimini bitirdi. Türk Telekom Anadolu Teknik Lisesi'nde 2003-2009 yılları arasında altı yıl öğretmen olarak çalıştı. 2009 yılında KBÜ TEF Elektronik Bilgisayar Eğitim Bölümü'nde öğretim görevlisi unvanı ile göreve başladı ve halen aynı kurumda çalışmaya devam etmektedir. Evli ve bir kız çocuk sahibidir. İyi derecede İngilizce bilmektedir.

ADRES BİLGİLERİ

Adres : Karabük Üniversitesi
Teknik Eğitim Fakültesi Elektronik Bilgisayar Eğitimi Bölümü
Balıklar Kayası Mevkii / KARABÜK

Tel : (505) 874 6373
E-posta : ikurnaz@karabuk.edu.tr