FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

OTONOM ARAÇLARIN GÜZERGAH TAKİBİ İÇİN BİR UYGULAMA

YÜKSEK LİSANS TEZİ

KADER NİKBAY

tarafından

YÜKSEK LİSANS

derecesi şartını sağlamak için hazırlanmıştır.

Ocak 2015

Program: Bilgisayar Mühendisliği

OTONOM ARAÇLARIN GÜZERGAH TAKİBİ İÇİN BİR UYGULAMA

YÜKSEK LİSANS TEZİ

KADER NİKBAY

tarafından

OKAN ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümüne

Yüksek Lisans

derecesi şartını sağlamak için sunulmuştur.

Onaylayan:

Chair
Name
Danışman
Prof. Dr. Bekir Tevfik AKGÜN

Member's
Name
Üye
Prof. Dr. Coşkun SÖNMEZ

Member's
Name
Üye
Yrd. Doç. Pınar YILDIRIM

Ocak 2015

Program: Bilgisayar Mühendisliği

ÖZET

Bu tez çalışması otonom araçların güzergah takibi için geliştirilmiş bir uygulamadır. Çalışmada bir harita üzerinde otonom araca verilen başlangıç noktasından, tanımlanan hedef noktasına derinlik öncelikli arama algoritması ile bir güzergah oluşturarak varması hedeflenmiştir.

Öncelikle konu ile ilgili daha önce yapılmış olan çalışmalar araştırılmış ve konunun gerçekleşmesine temel teşkil edecek simülasyon sistemleri, oyun motorları ve trafik sürüş sistemleri incelenmiştir. Sürücülü ve sürücüsüz araçların birlikte yer aldığı bir deneme oyun platformu geliştirilmiştir. Bu platformda izleme noktaları tek tek menü yarıdımıyla tanımlanarak güzergah takibi yapılmış, ancak bunun kullanılabilirliği zor olduğundan daha soyut tanımlardan güzergah takibinin gerçekleştirilmesi araştırılmıştır. Yol tanımlama ve hedef güzergah yapıları araştırılarak seçilen bir yapı üzerinde bir graf oluşturulmuş ve böylece graf üzerinde bilinen yol bulma algoritmaları denenmiştir. 2B bir sürüş simülasyonu gerçeklenmiş ve örnek senaryolar incelenmiştir. Son olarak ise uygulama sonrası elde edilen sonuçlar ve değerlendirmeler özetlenmiştir.

Anahtar Kelimeler: Araç Sürüş Simülasyonu, Sürücüsüz Araç, Yol Tanımlama Dosya yapıları

ABSTRACT

This thesis studying is an application developed to follow the route of autonomous vehicles and aims the vehicle arrives from the starting point to defined target point on the map via the route which using Depth First Search algorithm.

In the first section, the related studies done before are discussed and simulation systems underlying the realization of the subject, game engines and traffic driving systems have been studied. A trial game platform that included vehicles with and without driver has been developed. On this platform monitoring points have been defined with the help of the menu to follow the route but more abstract definitions has been investigated for route follow-up because of the difficulty of its' realization. Working on the DARPA way to define and target route structures a graph generated from the data in the file so finding known ways on graph algorithms were tested. In addition, a 2D driving simulation was implemented using DARPA structures and example scenarios were examined.Finally, the results obtained after application and assessment are summarized

Keywords: Simulation of Vehicle Driving, Autonomous Vehicle, Route Definition File

TEŞEKKÜR

Tez çalışmam süresince yol göstericiliğini, bilgisini benden esirgemeyen ve her konuda bana liderlik eden danışman hocam Sayın Prof. Dr. Bekir Tevfik AKGÜN'e vermiş olduğu emekler için en içten dileklerimle teşekkür ederim.

Hayatım boyunca bana her konuda yardımcı olan ve her zaman en büyük destekçim olan sevgili Annem ve Babama tüm kalbimle teşekkürlerimi sunuyorum.

KADER NİKBAY

OCAK 2015

İÇİNDEKİLER

TABLO LİSTESİ

ŞEKİL LİSTESİ

KISALTMALAR

DARPA	**:**	ABD Savunma Bakanlığı İleri Araştırma Projeleri Ajansı	The Defense Advanced Research Projects Agency
RNDF	**:**	Yol Ağı Tanımlama Dosyası	Route Network Definition File
MDF	**:**	Hedef Güzergah Dosyası	Mission Data File
DFS	**:**	Derinlik Öncelikli Dolaşma Algoritması	Depth-first search algorithm
BFS	**:**	Genişlik Öncelikli Dolaşma Algoritması	Breadth-first search algorithm
2B/2D	**:**	İyi Boyutlu	Two Dimension
LIDAR	**:**	Lazer darbeleri kullanılarak bir nesne veya bir yüzeyin uzaklığını anlamaya yarayan teknoloji	Laser Imaging Detection and Ranging
OKANOM	**:**	Oyun uygulamasına verilen ad.	

1. GİRİŞ

Günümüzde bilgisayarın insan yaşamına girmesi ile birlikte bilgisayar sistemleri insanın yerini almaktadır. Her alanda olduğu gibi trafikte de sürücülü araçların yerini otonom araç olarak adlandırılan temelde yapay zekâ ve güzergah takip algoritmalarını kullanan bilgisayar sistemleri almaya başlamıştır.

Sürücüsüz araç fikri ilk olarak "Kara Şimşek" dizisinde kullanılmıştır. 1980 yılında Mercedes-Benz mühendisi olan Ernst Dickmanns'ın tasarladığı kamera görüntüsü ile kendini süren robotik araç ile birlikte bu fikir sadece bir dizi senaryosu olarak kalmanın ötesine geçmiştir. 1994 yılında Daimler Benz ve Ernst Dickmanns, Vamp ve Vita-2 adlı araçlar ile Paris'in 3 şeritli bir otobanında, sıradan bir yoğun trafik durumundayken saatte 130 kilometre hız ile 1000 kilometre yol kat etmeyi başardı. Aracın kontrolü tamamen devraldığı durumlar şunlardı: şerit boşken, konvoy şeklinde ilerlerken ve sağ ve sola doğru şerit değiştirirken. 1995 yılında Dickmanns S-sınıfı bir Mercedes-Benz'i 1600 km'lik mesafedeki Münih'ten Kopenhag'a gidip götürmeyi başardı. Araç 175 km/h'lik bir hıza ve %95'lik bir otonomluğa (yolun %95inde kontrol tamamen arabadaydı sadece 9 km' sinde insan müdahalesi oldu) ulaştı. 2007 yılında DARPA (The Defense Advanced Research Projects Agency, (ABD Savunma Bakanlığı İleri Araştırma Projeleri Ajansı)) tarafından düzenlenen şehir içi yarışlarında sürücüsüz, robotik arabalar trafik kurallarına uyarak diğer arabalara yayalara ve çarpılmaması gereken şeylere çarpmadan 60 millik parkuru altı saatin

altında bitirmeye çalışmışlardır. 2010 yılında Vislab adlı şirkettin düzenlediği yarışmada sürücüsüz dört araç 13000 km yol kat etmiş ve İtalya'dan Çin'e, Shangai'daki Expo'ya uğrayarak yarışmayı bitirmişlerdir. Ekim 2010 da ise Google tarafından yapılan araç San Francisco- Los Angeles arasında toplamda 230000 km yol kat etmiştir.[1][2] Teknolojideki bu denli hızlı gelişim sayesinde sürücüsüz araçlar trafikteki yerlerini almaya başlamışlardır.

Sürücüsüz araçlarda ki en büyük problemlerden biri araçların güzergah takibinin yapılmasıdır. Bu sebeple çalışmada öncelikli olarak sürücülü ve sürücüsüz araçların birlikte yer aldığı bir deneme oyun platformu geliştirilmiştir. Bu platformda takibi için gerekli olan izleme noktaları geliştirme ortamı üzerindeki menülerden tek tek tanımlanarak gerçekleştiriliyordu, ancak bunun kullanılabilirliği zor olduğundan daha soyut tanımlardan takibinin gerçekleştirilmesi araştırılmıştır. Yapılan araştırma neticesinde DARPA yol tanımlama ve hedef dosya yapıları seçildi. Çalışma için bahsi geçen bu dosya yapılarını kullanan 2B bir sürüş simülasyon yazılımı geliştirildi.

Tez çalışması süresince; çalışma konusu ile alakalı olarak Eleco2014 (Ek-A) ve Akademik Bilişim 2015 (Ek-B) konferanslarında birer adet yayın çıkmıştır. Bahsi geçen yayınlara "Ekler" bölümünden ulaşılabilir.

2. SİMÜLASYON SİSTEMLERİ VE OYUN MOTORLARINA GENEL BAKIŞ

Bu bölümde genel olarak 3 boyutlu simülasyon yazılımları ve oyun motorları hakkında kısaca karşılaştırmalı bilgi verilecektir.

3B simülasyon yazılımları ile 3B oyun motorlarının temel hedefleri ayrı olsa da kullanıcıya sunulan grafik ürünlerinin gerçek zamanda üretilmesi konusunda birleşirler. Buradaki gerçek zaman tanımı kullanıcının algılayabileceği zaman dilimleri içinde anlamlı ekran görüntülerinin üretilmesini kapsar. 3 boyutlu nesnelerinin bilgisayar ve kullanıcı denetimi uyarınca ekranda gösterilmesi alt düzey programlama (örneğin C dili kullanarak) ile erişilen donanım grafik alt sistemini yöneten kütüphaneler (OpenGL, DirectX) üzerinden yapılır. Grafik kütüphaneleri 3 boyutlu simülasyon/oyun nesnesini doğrudan desteklememekte sadece nokta, çizgi ve çokgen üzerinde çalışmaktadır[3]. Çeşitli firmalar belli alanlara yönelik karmaşık yapılı grafik uygulamalarına yönelik ortam ve nesnelerin üretilmesi ve denetlenmesini sağlayan simülasyon yazılım paketleri/oyun motorları üretmiştir. Bunları sadece grafik üretimi ile sınırlı kalmayıp çoklu ortam bileşenlerine de (ses, efekt, müzik, video vb.) destek vermesi beklenir.

3B simülasyon sistemlerinin bilgisayar oyunlarının aksine ele aldığı konuya sahici yaklaşması ve daha kesin hesaplamalar yapılması gerekir. Dolayısı ile bu sahiciliği

sağlamak için önemli işlem gücüne gereksinim duyulur. Bilgisayar oyunlarının temel amacı ise yaratılmak istenen hayalin içine oyuncuyu çekebilmektir. Bunu gerçek zamanda sağlamak için gerçeklemede görsel kandırmacaları da kapsayan hileler kullanılır. Gelişim süreci göz önüne alındığında gerek donanımlarının pahalı olması ve gerekse özel olarak üretilmesi gereken yazılımların çokluğu simülasyon sistemleri etkin örneklerinin daha önce çıkmasına neden olmuştur. Bu ürünlerin talep sayısı az olmasına rağmen gerek duyan firmalar tarafında yüksek bedellerle satın alınması sonucunda konu üzerinde bilgi ve yazılım birikimi sağlamıştır. Kütüphanelerin ve yazılım paketlerinin, üzerinde çalışması gereken donanım sistemi maliyetleri nedeni ile yaygınlaşması başlangıçta mümkün olmamıştır. Bilgisayar oyunlarını talep eden kitlenin büyüklüğü ve ödemeye hazır oldukları bedeller sonuçta hem grafik donanım/bilgisayar maliyelerini düşürmüş ve hem de performansı yüksek oyun motorlarını ortaya çıkarmıştır. Günümüzde ise oyun motorların simülasyon sistemlerinin de önemli bir parçası olmaya başladığını gözlemlemekteyiz.

3. SÜRÜŞ VE TRAFİK SİMÜLASYON SİSTEMLERİ

Bir sürüş simülatörünün önemli bölümleri 3B grafik yazılımı, konuya yönelik simülasyon modelleme ile hesaplama yazılımları ve mekanik teçhizattır.

Çoğu kez ürün olarak satılan bir simülasyon yazılım paketi yeni bir tasarımda tümüyle yeterli olmamakta; örneğin araç dinamikleri modelleme ve gerçekleme gibi konularda diğer yazılımlar/paketleri kullanarak yeni üretimler yapılması ve bu üretimlerin simülasyon sistemine ilişkilendirilmesi gerekmektedir. Bu bölümde bazı sürüş ve trafik simülasyon sistemleri tanıtılmıştır. Tez konusuna yakın olan içerikleri sebebiyle bu örnekler seçilmiş ve incelenmiştir;

SUMO Alman Havacılık ve Uzay Merkezi Ulaşım Sistemleri Enstitüsü çalışanları tarafından, geniş yol ağlarını işlemek için tasarlanmış bir simülatör paketidir. GPL altında lisanslı ve açık kaynak kodlu bir yazılımdır [4][5].

Carnetsoft Hollanda'da geliştirilmiş simülasyon sistemidir. Sürücülerin trafiğe çıkmadan önce oluşturulan sanal trafik ortamı içerisinde eğitilmelerini amaçlar. Masaüstü simülatör ve kokpit simülatör türleri vardır [6].

City Car Driving Rusya'da kullanılan üç boyutlu eğitim simülatörüdür. Amacı kullanıcılara trafik kurallarını benimsetmek ve trafikte sürüş pratiği kazandırmaktır [7].

Trafic Talent sürüş simülasyonunda kullanıcıya verilen farklı görevlerin tamamlanması istenir. Bu görevler genel olarak verilen hedeflere kaza yapmadan, verilen süre zarfında ve trafik kurallarına uyarak varmak şeklindedir [8].

Ohio State Üniversitesi Sürücü Simülatörü sistem denetimi ve sürücü senaryoları oluşturmada Realtime Technologies Inc. firmasının SimCreator ve SimVista yazılım sistemleri kullanılmıştır. Çoklu gövde dinamik yazılımı ile hareket denetimi geliştirilmiştir. Simülatör; altı eklemli platform üzerine yerleşmiş bir araç kaportası, denetimleri kapsayan mekanik teçhizat ve 260° görüşlü silindirik bir ekran ile donatılmıştır [9][10].

Yonsei Üniversitesi Simülatörü Yonsei üniversitesi tarafından geliştirilmiş otonom araca ait testleri öncelikle bilgisayar ortamında gerçeklemek için oluşturulan bir simülatördür. GPL lisansı ile OpenCarSimManager kullanan açık kaynaklı bir projedir [11].

4. GRAF ÜZERİNDE ARAMA ALGORİTMALARINA GENEL BAKIŞ

Graflar üzerinde çalışan birçok arama algoritması mevcuttur. Fakat üzerinde çalıştığımız örnek grafların karmaşıklıkları genelde basit düzeyde olduğundan "Derinlik Öncelikli Dolaşma"algoritması yeterli olmuştur. Bu nedenle bu bölümde detaylı olarak açıklanmıştır. Graflar üzerinde kullanılabilecek diğer algoritmalarda bu bölümde özet şekilde bahsedilecektir.

4.1. Derinlik Öncelikli Dolaşma

Derinlik öncelikli dolaşma tekniğinde arama işlemine kök düğümden başlanır. Daha sonra kök düğüme ait çocuk düğümlerin en solundakine gidilerek hedef aranır[12]. Bu işleme hedefe erişinceye kadar bir alt seviyedeki düğümlerin en solundakine gidilerek devam edilir. Şayet hedefe ulaşmadan bir terminal düğüme rastlanırsa, bir seviye öne dönülüp, bir adım sağa hareket ettikten sonra hedefe varıncaya kadar sola gidilir [13][14].

İşlem adımlarını maddelersek;

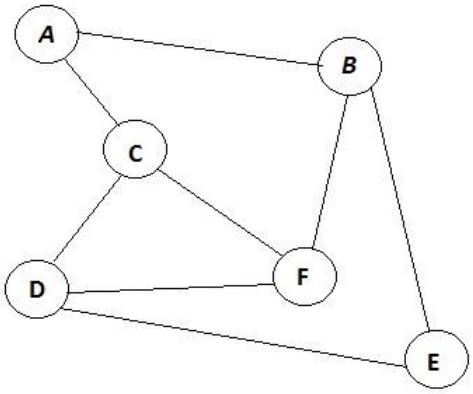
1. Önce bir başlangıç düğümü seçilir ve ziyaret edilir.

2. Seçilen düğümün en soldaki komşusu seçilir ve ziyaret edilir.

3. 2.adım ziyaret edecek komşu kalmayıncaya kadar tekrar edilir.

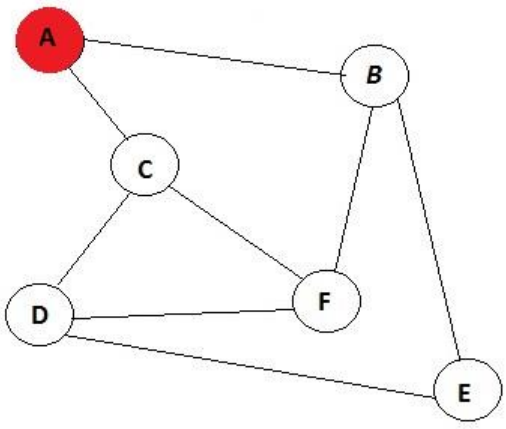
4. Komşu kalmadığında tekrar geri dönülür ve önceki ziyaret edilmiş düğümler için adım 2 ve 3 tekrar edilir [15].

Bu algoritmanın çalışmasını bir örnek üzerinde açıklarsak;



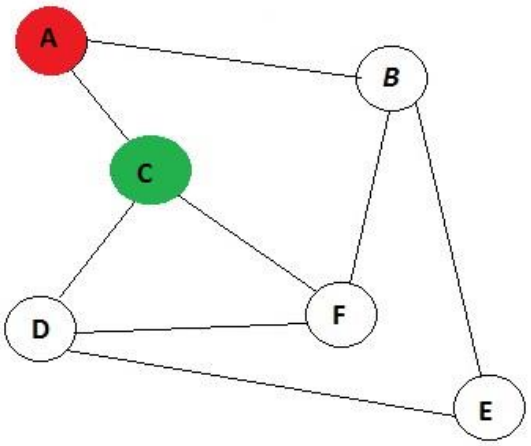
Şekil 4.1. Örnek graf yapısı

Elimizde "Şekil 4.1." de görüldüğü üzere bir graf yapısı olsun ve başlangıç düğümü "A" , gidilecek hedef düğüm "E" olsun. Şekillerde ziyaret edilen düğümler kırmızı renkle, bir sonraki adımda ziyaret edilecek düğüm ise yeşil renkle ve hedef düğüme ulaşılması durumu mavi renkle gösterilecektir.



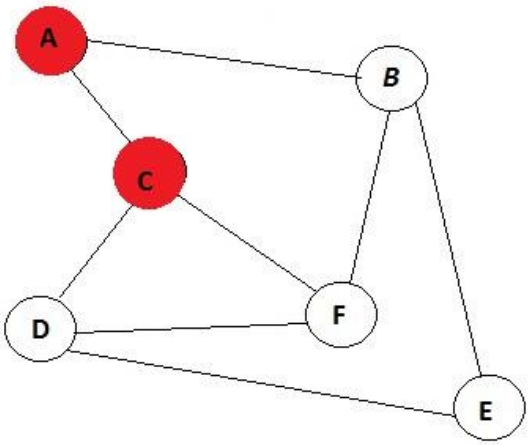
Şekil 4.2. Başlangıç düğümü

“Şekil 4.2.” de görüldüğü üzere başlangıç düğümü olan “A” noktası ziyaret edilmiştir. Bu sebeple kırmızı ile gösterilmektedir.



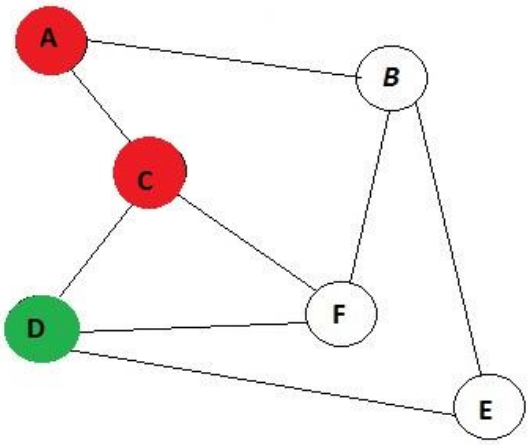
Şekil 4.3. Ziyaret edilecek “C” düğümü

Dolaşmanın bir sonraki adımı “Şekil 4.3.” de görülmektedir. Burada “A” düğümünün en sol komşusu olan “C” düğümü ziyaret edilecektir.



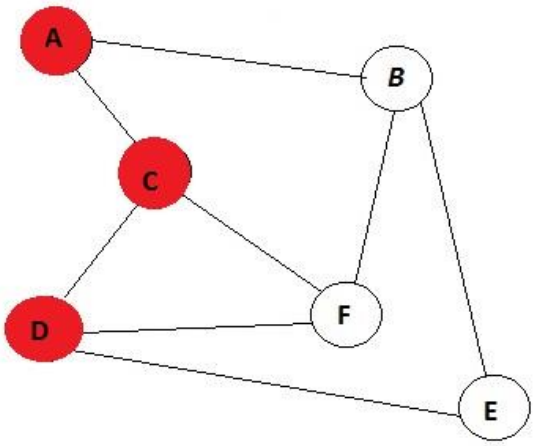
Şekil 4.4. Ziyaret edilen “C” düğümü

Bir sonraki adım “Şekil 4.4.” de görülmektedir. Burada “C” düğümü ziyaret edilmiştir.



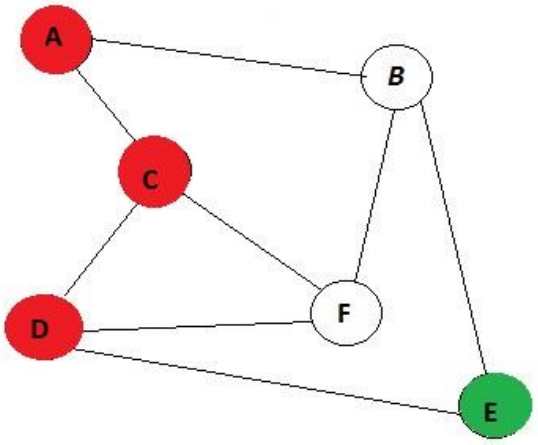
Şekil 4.5. Ziyaret edilecek “D” düğümü

Dolaşmanın bir sonraki adımı “Şekil 4.5.” de görülmektedir. Burada “C” düğümünün en sol komşusu olan “D” düğümü ziyaret edilecektir.



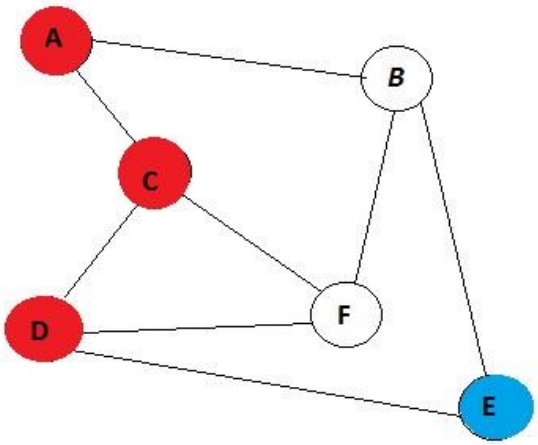
Şekil 4.6. Ziyaret edilen “D” düğümü

“Şekil 4.6.” da görüldüğü üzere, “D” düğümü ziyaret edilmiştir.



Şekil 4.7. Ziyaret edilecek “E” düğümü

Bir sonraki adımı “Şekil 4.7.” de görülmektedir. Burada “D” düğümünün en sol komşusu olan “E” düğümü ziyaret edilecektir.



Şekil 4.8. Hedef düğüm

Dolaşmanın en son adımı “Şekil 4.8.” de görülmektedir. Algoritma hedef düğüme olan “E” düğümüne ulaşarak işlemini başarı ile tamamlamıştır.

Derinlik öncelikli dolaşma algoritmasına göre “A” noktasından “E” noktasına gidiş için dolaşılan düğümler; A-C-D-E şeklinde sıralanmaktadır [16].

4.2. Genişlik Öncelikli Dolaşma

Genişlik öncelikli dolaşma tekniği, derinlik öncelikli dolaşma tekniğinin tersidir. Bu teknikte de, bir alt seviyeye inilmeden önce, aynı seviyedeki düğümler kontrol edilir[12]. Araştırmaya aynı seviyedeki düğümlerden en solda olandan başlanıp, sağa doğru olan düğümlere doğru gidilir [13][14].

İşlem adımlarını maddelersek;

1. Genişlik öncelikli dolaşma ağaçlardaki seviye sıralamaya benzer.

2. Seçilen düğümün tüm komşuları sırayla seçilir ve ziyaret edilir.

3. Her komşu kuyruk yapısı içerisine atılır.

4. Komşu düğüm kalmadığında kuyruk içerisindeki ilk düğüm alınır ve 2.adıma gidilir [15].

4.3. Dijkstra Algoritması

Dijkstra algoritması, bir graf içerisindeki iki nokta arasındaki optimal yolun

hesaplanmasını sağlar[26]. Burada ifade edilen optimal yol kavramı ile duruma göre ‘en kısa’, ‘en ucuz’ veya ‘en hızlı yol’ anlamları düşünülebilir.

Dijkstra algoritmasında kaynak olarak bağlı, ağırlıklı ve ağırlık değerlerinin tamamı pozitif olan bir graf kullanılmaktadır. Algoritma sonuç değer olarak iki nokta arasındaki en kısa yolu vermektedir [27].

İşlem adımlarını maddelersek;

1. Kaynak düğümü seç.

2. S düğümler dizisi tanımla ve boş diziye sıfırla. Algoritma süreci içerisinde en kısa yolu veren düğümler S dizisi içerisine kayıt edilecektir.

3. Kaynak düğümü 0 olarak etiketle ve S dizisi içerisine ekle.

4. Yeni eklenen düğüme bir kenarla bağlanmış olan S dizisi içerisinde olmayan her düğümü göz önünde bulundur. S içinde olmayan düğümü, yeni eklenen düğüm etiketi + kenar uzunluğuyla beraber etiketle.

5. S içinde olmayan düğümü, en küçük etiket değeriyle al ve S dizisine ekle.

6. Adım 4’e git (hedef düğüm S dizisi içerisinde olana kadar veya etiketsiz düğüm kalmayana kadar).

4.4. Floyd Algoritması

Floyd algoritması, bir graf içindeki tüm noktalar arasındaki en düşük maliyetliyolu bulmak için tasarlanmıştır. Algoritma, noktalar arasındaki kenar maliyetlerini gösteren bir matris üzerinde işlem yapar [28].

Bu matrise komşuluk matrisi ismi verilmektedir. Floyd algoritmasının sözde kodu aşağıdaki gibidir [16].

Komşuluk matrisi üzerindeki maliyetleri başlangıç maliyeti olarak kabul et

Rota matrisine boş anlamında değerler yerleştir. (ROTA = -1)

Düğümlerin kendilerine ulaşım maliyetlerini sıfırla (UM[i][i]=0)

```
for (aradüğüm sayacı < düğüm sayısı) {
        for (satır sayacı < düğüm sayısı) {
                for (sütun sayacı < düğüm sayısı ) {
                if (ara düğüm üzerinden gitmek daha kısa ise) {
                aradüğümlü maliyeti en küçük maliyet kabul et;
                Rota bilgisini güncelle}
                }
        }
}
```

4.5. A*Algoritması

Bilgisayar bilimlerinde en kısa yol bulmak için kullanılan algoritmalardan birisidir. Oyun programlamada, oyunda bulunan oyuncuların en kısa yolu bularak hedefe gitmeleri için de sıklıkla kullanılan algoritmadır.Kısaca bir düğümden hedef bir düğüme en kısa hangi düğümler üzerinden gidileceğini bulmaya yarayan “en iyi yerleştirme” algoritmasıdır.

f(n) = g(n) + h(n) denklemindeki

f(n) = hesaplama yapan sezgisel (heuristic) fonksiyon.

g(n) = Başlangıç düğümünden mevcut düğüme kadar gelmenin maliyeti

h(n) = Mevcut düğümden hedef düğüme varmak için tahmin edilen mesafe.

Dikkat edileceği üzere f(n) fonksiyonunun sezgisel olma sebebi, bu fonksiyon içerisinde bulunan ve tahmine dayalı olan h(n) sezgisel fonksiyonudur.

İşlem adımlarını maddelersek;

Algoritma yukarıdaki toplama işlemini kullanan oldukça basit bir yapıya sahiptir. Veri yapısı olarak bir öncelik sırası (priority queue) kullanan algoritmada en öncelikli olan düğüm f(n) değeri en düşük olan düğümdür.

1. Algoritma her adımda en düşük değeri (Ve dolayısıyla en önemli) düğümü alır (yani bu düğüme gider) ve düğümü sıradan (queue) çıkarır.

2. Gidilen bu düğüme göre komşu olan bütün düğümlerin değerleri güncellenir (artık bu düğüme gelmenin bir maliyeti vardır ve dikkat edilirse f(n) fonksiyonu içerisinde bu değer yer almaktadır.)

3. Algoritma yukarıdaki adımları hedefe varana kadar (yani hedef düğümü öncelik sırasında (priority queue) en öne gelene kadar) veya sırada (queue) düğüm kalmayana kadar tekrarlar [29].

5. YOL TANIMLAMA VE HEDEF GÜZERGAH DOSYA YAPILARI

Uygulamada öncelikle yol ağı tanımlama dosyasındaki verileri yol bulma algoritması ile yorumlanarak bir çizgeye dönüştürülmektedir. Ardından otonom araca hedef güzergah tanımlama dosyasındaki veriler aktarılarak, dosyada belirtilen güzergah üzerindeki kontrol noktalarını ziyaret ederek, verilen hedefe ulaşması beklenir.

Araştırmalar sonucunda bu isterleri karşılayan iki farklı kaynağa ulaşıldı. Bu kaynaklardan bölümün ilerleyen kısımlarımda bahsedilecektir.

5.1. DARPA

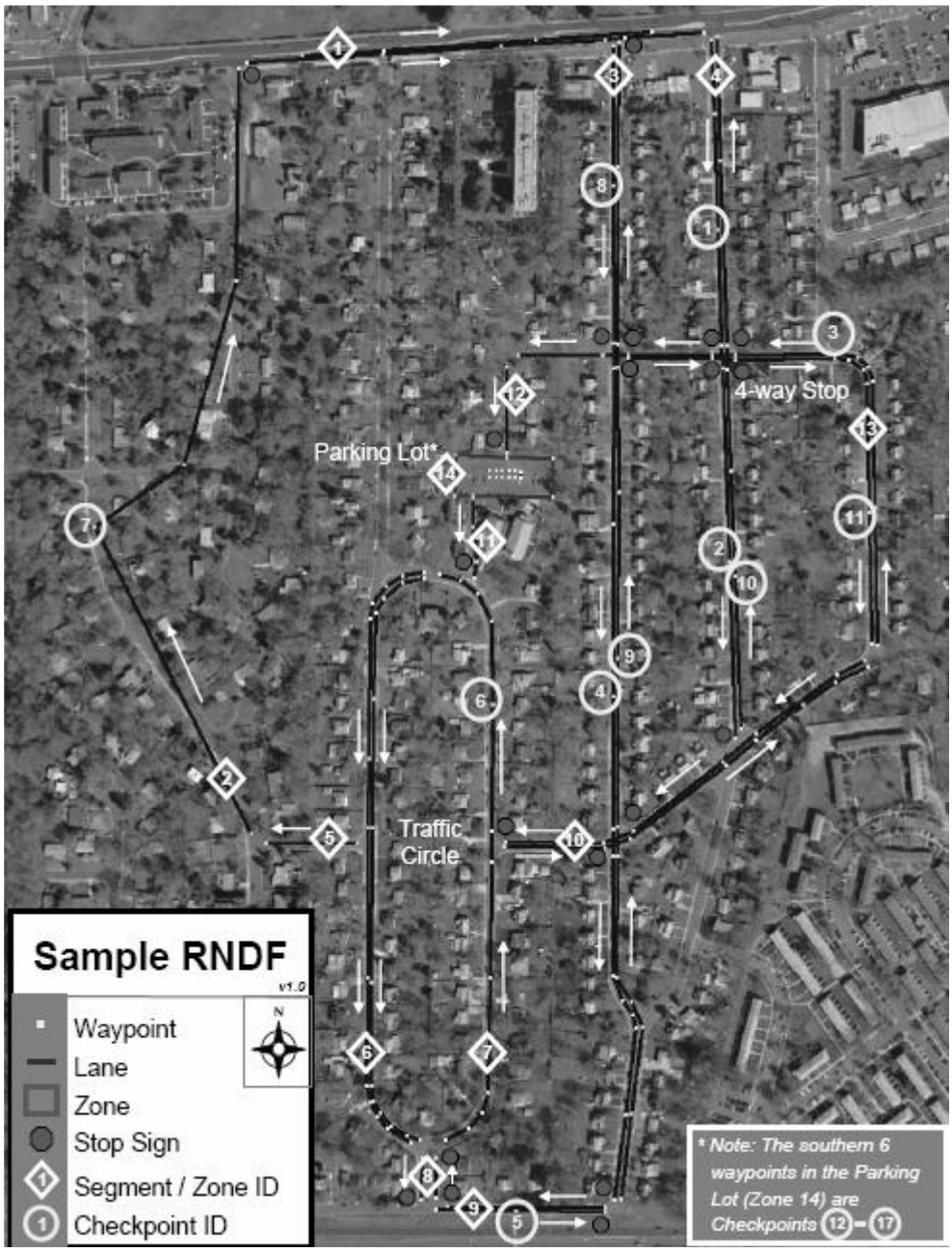
DARPA Türkçe adıyla ABD Savunma Bakanlığı İleri Araştırma Projeleri Ajansı 1958 yılında ARPA (Advanced Research Projects Agency) adıyla Virginia'da kuruldu. Ordu tarafından kullanılmak üzere, yeni teknolojiler üretmekten sorumlu kuruluştur.

Sürücüsüz araç teknolojilerinin gelişmesi ile birilikte DARPA tarafından otonom araç yarışları düzenlenmeye başlandı. 2002 yılında ABD Savunma Bakanlığı "Büyük Mücadele" anlamına gelen Grand Challenge adıyla bir yarışma başlattı. Sadece Amerikalı'ların katılabileceği bu yarışmada amaç otonom bir aracın Amerika'daki bir çölde belirlenmiş hedefe kendi kendine gitmesini sağlayabilecek bir sistem geliştirmekti. 2004 yılında sonlanan bu yarışmanın ödülü 1 milyon dolardı. Oysa

katılan 25 takımdan bu hedefe ulaşabilen takım çıkmadı. Ancak 2005 yılında yinelenen yarışta beş takım 217 km'lik rotayı tamamlamayı başardı, böylece Stanford Üniversitesi'ni temsil eden takım 2 milyon dolarlık büyük ödülü kazandı. 2007 yılında 3,5 milyon dolara çıkarılan toplam ödül yarışmayı tamamlayabilen altı sürücüsüz araç tarafından paylaşıldı. DARPA Şehir Mücadelesi adıyla bilinen bu yarışmada hedef 89 km'lik şehir içi trafiğinde, ralli stili bir yarış sonunda hedefe ulaşmaktı.

DARPA Şehir Mücadelesinin temel yapısı;

DARPA şehir içi araba yarışında otonom otomobilin önceden MDF yani yol tanımlama dosyasında olan kontrol noktalarını ziyaret etmesi istenmektedir. Bu amaçla herhangi bir konum/kontrol noktası ile bir sonraki kontrol noktasına olan yolun planlanması yapılmıştır. Yol planlayıcı modül öncelikle RNDF yol ağı tanımlama dosyasını bilinen yol bulma algoritmaların uygulanacağı şekilde bir çizgeye dönüştürmektedir. Daha sonra ise arabanın GPS/pusula'sından dan okunan konum-yön bilgisi, oluşturulan çizge, önerilen bir sayıl çarpım fonksiyonu ve A* algoritması kullanılarak ilk kontrol noktasına olan yol bulunmaktadır. Araç ilk kontrol noktasına vardıktan sonra diğer kontrol noktalarına olan yol direk A* algoritması ile bulunmaktadır [30].



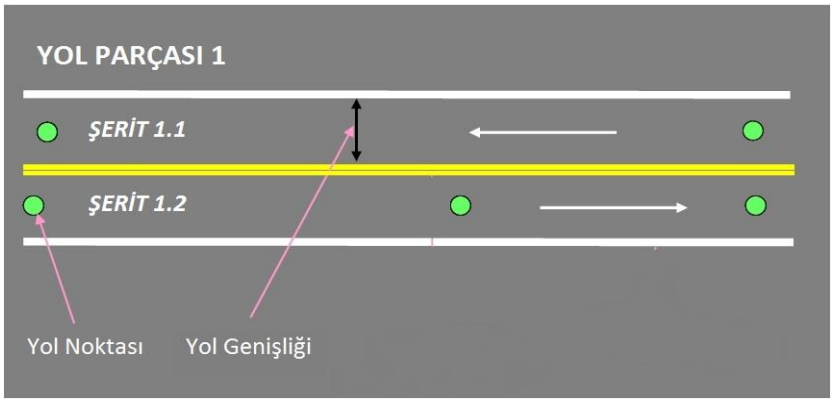
Şekil 5.1. Örnek yarış güzergahı[1]

[1] DARPA Urban Challenge Route Network Definition File and Mission Data File Formats, 2007,1-14

5.1.1. Yol ağı tanımla dosya yapısı (RNDF)

Yol parçaları

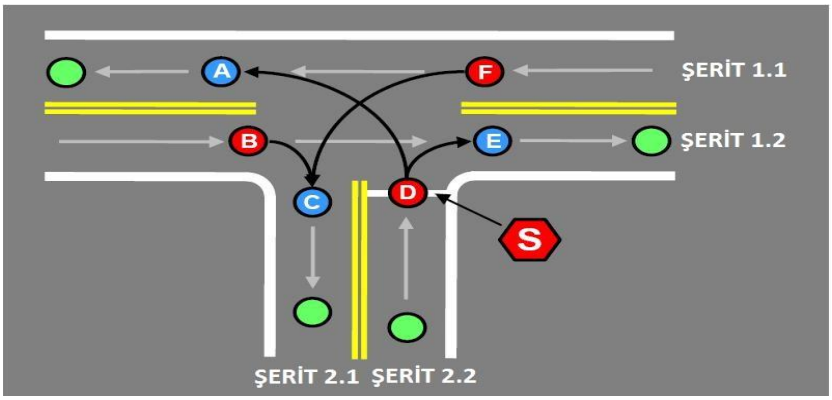
Bir yol ağı bir veya birden çok şeridin bir araya gelmesi ile oluşur. Sözü edilen bu yol parçaları tek şeritli ya da çok şeritli olabilir. Bir şerit içinde şeridin genişliği, şerit çizgileri ve şerit ile ilişkili yol noktaları yer alır. Bu yol noktaları genellikle şeridin merkezine yerleştirilir. Ardışıl olarak yol noktaları takip edilerek aracın seyahat etmesi sağlanır [18][22].



Şekil 5.2. Yol parçası ve şeritler[2]

"Şekil 5.2." de bir yol parçası ve üzerinde sahip olduğu diğer bileşenler görülmektedir. Şeritlerin genişlik bilgisi zorunlu bir bilgi değildir bu sebeple her yol parçası içinde var olma zorunluluğu yoktur. RNDF dosyası içinde yol noktalarına giriş ve çıkış bilgileri atanarak yol parçaları arasında bağlantı sağlanır [18][21].

[2] DARPA Urban Challenge Route Network Definition File and Mission Data File Formats, 2007,1-14



Şekil 5.3. Yol noktaları[3]

"Şekil 5.3."de mavi ile gösterilen yol noktaları girişleri, kırmızı ile gösterilen yol noktaları çıkışları göstermektedir. Örneğin "D" ile gösterilen çıkış noktası için "A" ve "E" diğer şeritlere giriş noktalarını oluşturmaktadır. Şekilde görüldüğü gibi bir çıkış noktası birden fazla giriş noktası ile ilişkilendirilebilir. Ayrıca D noktasına "S" ile gösterilen bir "Dur" bilgisi de eklenmiştir. Bunun nedeni kavşakların kesişim noktalarında trafik durum bilgisinin kontrol edilme zorunluluğudur.

Şeritler üzerinde var olan yol noktalarından bir tanesi kontrol noktası olarak belirlenir ve bu noktalar araçlar tarafından güzergahı takip etmek için kullanılır [18][22].

Park alanları

RNDF dosyası içerisinde yol parça bilgilerine ek olarak park alan bilgileri de mevcuttur. Park alan bilgilerinde giriş, çıkış noktaları, yol noktaları ve park alanın çevre nokta bilgileri yer almaktadır. Yol parçaları ve park alanlarının bağlantıları iki şekilde gerçekleşmektedir;

1.Durum: Yol parçası çıkış noktası ile park alanı giriş noktası ilişkilendirilir.

[3] DARPA Urban Challenge Route Network Definition File and Mission Data File Formats, 2007,1-14

2.Durum: Park alanı çıkış noktası ile yol parçası giriş noktası ilişkilendirilir.

Park bölgesindeki özel olarak belirtilmiş araç park yerlerinde 2 adet yol noktası bulunur ve bu noktalardan biri aynı zamanda kontrol nokta bilgisini de taşır. "Şekil 5.4."de bir park alanındaki belirtilen durumlar görülmektedir [18][20].

Şekil 5.4. Park alanı[4]

RNDF formatı

RNDF'in genel dosya formatı [18];

RNDF_name *filename (string)*

num_segments *number_of_segments (integer>0)*

num_zones *number_of_zones (integer≥ 0)*

<optional file header>

[4] DARPA Urban Challenge Route Network Definition File and Mission Data File Formats, 2007,1-14

<segment 1>

.
<segment M>

<zone M+1>

.
<zone M+N>

end_file

RNDF dosya yapısında yol parçası tanımları;

segment *segment_id (integer>0)*

num_lanes *number_of_lanes (integer>0)*

<optional segment header>

<lane 1>

.

.

<lane N>

end_segment

RNDF dosya yapısında şerit tanımları;

lane *lane_id (x.y; x,y∈ integer>0)*

num_waypoints *number_of_waypoints (integer>0)*

<optional lane header>

<waypoint 1>

.

.

<waypoint P>

end_lane

RNDF dosya yapısında park alan tanımları [18];

zone *zone_id (integer>0)*

num_spots *number_of_parking spots (integer≥0)*

<optional zone header>

<perimeter>

<optional spot 1>

.
.
<optional spot N>

end_zone

5.1.2. RNDF örneği

RNDF dosyası üzerindeki bir yol parçası ve bileşenlerinin tanımı [18].

RNDF_name Sample_RNDF_Rev_1.2

num_segments13

num_zones 1

format_version 1.0

segment 1

num_lanes 2

segment_nameMichigan_Ave

lane 1.1

num_waypoints 4

lane_width 12

left_boundary double_yellow

right_boundary broken_white

1.1.1 38.875413 -77.205045

1.1.2 38.875471 -77.204189

1.1.3 38.875585 -77.202593

1.1.4 38.875673 -77.201373

end_lane

lane 1.2

num_waypoints 6

lane_width 12

left_boundary broken_white

exit 1.2.4 3.1.1

exit 1.2.6 4.1.1

1.2.1 38.875343 -77.205619

1.2.2 38.875438 -77.204198

1.2.3 38.875528 -77.202959

1.2.4 38.875602 -77.201871

1.2.5 38.875616 -77.201664

1.2.6 38.875676 -77.200830

end_lane

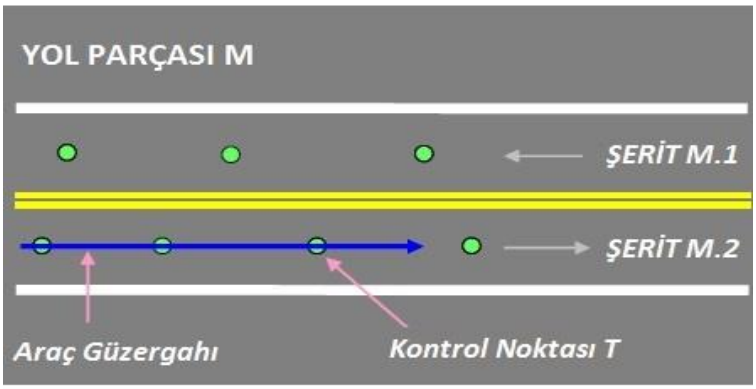
end_segment

5.1.3. Hedef güzergah dosya yapısı (MDF)

Aracın verilen göreve ulaşması için, ziyaret etmesi gereken kontrol nokta bilgilerinin tutulduğu dosyadır. Her MDF dosyası belirli bir RNDF dosyası tarafından yorumlanır. Birçok farklı MDF dosyası aynı RNDF dosyası ile ilişkili olabilir [18][22].

Kontrol noktaları

Aracın verilen hedefe ulaşması için izlenecek rota üzerinde ziyaret edilmesi gereken noktalardır. RNDF dosyasında tanımlanmış olan kontrol noktaları, araç üzerinden geçtiği takdirde ziyaret edilmiş sayılmaktadır ve böylece belirlenen kontrol noktaları ziyaret edildikçe aracın doğru rotada olduğu tespit edilir. “Şekil 5.5.” de bir aracın ziyaret etmiş olduğu kontrol noktaları görülmektedir [18][20].



Şekil 5.5. Araç güzergahı[5]

RNDF içinde tanımlanmış olan tüm kontrol noktaları MDF içerisinde yer almak zorunda değildir. Her hedef için aracın ziyaret etmesi gereken belirli kontrol noktaları vardır. Hedefe ulaşmayı sağlayan son kontrol noktası bitiş çizgisi görevini üstlenmektedir [18][21].

Hız Limitleri

MDF dosyasında aracın hedefe ulaşması sürecinde ziyaret edeceği her yol parçası için tanımlanmış olan minimum ve maksimum hız değerleri mevcuttur. Aracın mevcut olan hızını koruması için minimum ve maksimum değerlere aynı hız değerinin atanmış olması gerekir [18].

[5] DARPA Urban Challenge Route Network Definition File and Mission Data File Formats, 2007,1-14

MDF'in genel dosya formatı [18];

MDF_name *filename (string)*

RNDF *RNDF_name (string)*

<optional file header>

checkpoints

num_checkpoints *number_of_checkpoints (integer>0)*

<checkpoint 1>

.
.
<checkpoint L>

end_checkpoints

speed_limits

num_speed_limits *number_of_speedlimits (integer>0)*

<speed_limit 1>

.
.
<speed_limit M>

end_speed_limits

end_file

5.1.4. MDF örneği

MDF_name Sample_MDF_Rev_1.2

RNDF Sample_RNDF_Rev_1.2

format_version 1.0

checkpoints

num_checkpoints 6

1

2

3

4

5

10

end_checkpoints

speed_limits

num_speed_limits 14

1 10 25

2 0 20

3 10 20

4 10 20

5 10 15

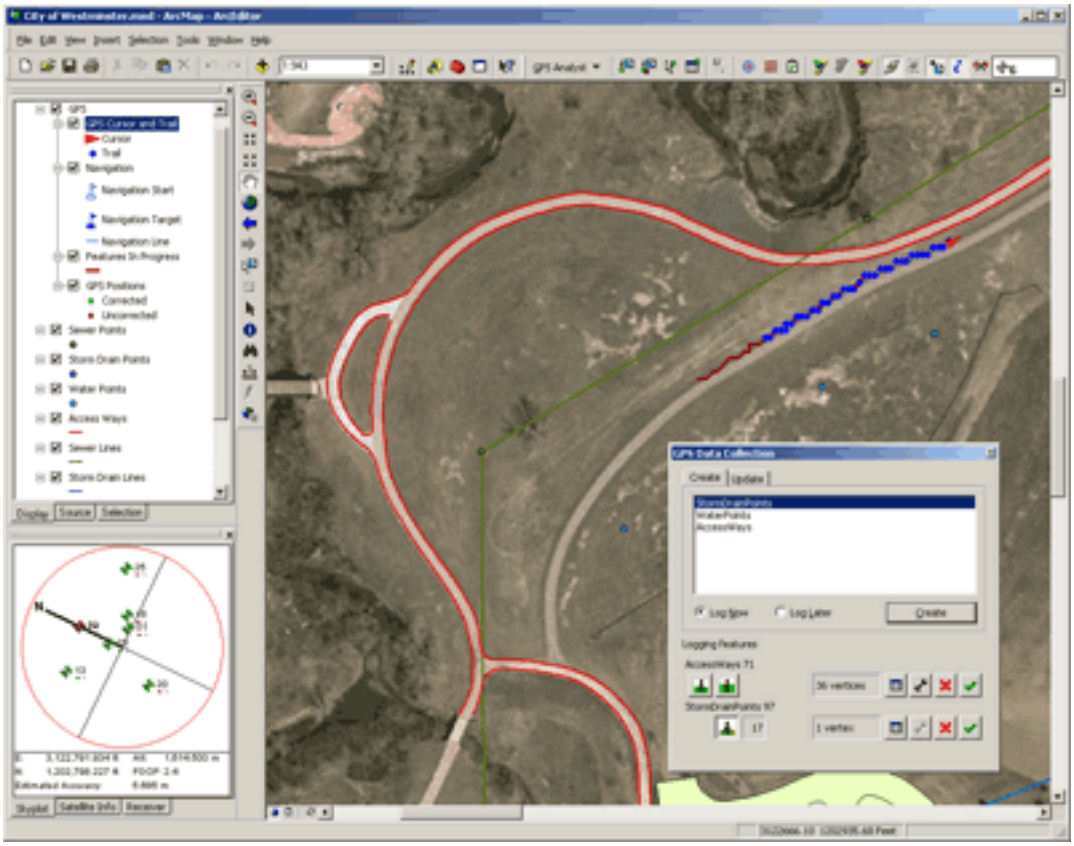
end_speed_limits

end_file

[18].

5.2. TRIMBLE

Trimble firması 1978 yılında Silikon Vadisinde kurulmuştur. Firma GPS ve navigasyon sistemleri üzerine 1982 yılında çalışmaya başlamıştır. DARPA'daki RNDF yapısına benzer bir yapı olan GENIO adını verdikleri dosya üzerinde güzergah ve harita bilgileri tutulmaktadır [33].



Şekil 5.6. Trimble GPS görünümü[6]

[6]http://www.gpslands.com.my/v2/components/com_virtuemart/shop_image/product/trimble-gps-analyst.png

6. SÜRÜCÜSÜZ ARAÇ TRAFİĞİ İLE SÜRÜCÜ DAVRANIŞI İYİLEŞTİRMEYE YÖNELİK BİR OYUN: OKANOM

6.1. OKANOM Oyununun Gerçeklenmesi

Oyunun gerçeklenmesi için kullanılan teknoloji ve oyunun önemli gerçeklenme bölümleri olan araç modeli, yerleşke modeli ve kullanıcı arayüzü tanıtılacaktır. Araç modeli alt bölümünde; araç katı cismi oluşturma, araç modelinin dinamik ve sensör özellikleri ile bazı önemli parametreler yer alır. Yerleşke modeli alt bölümünde ise; yol ve kavşakların oluşturulması ve sürücüsüz araçların belirlenen güzergah üzerinde gitmesi açıklanır. Son olarak kullanıcı arayüzü alt bölümünde, sürücülü araç için kullanıcıya sunulan denetim ve yönlendirme tanıtılır.

6.1.1. Unity3D

Unity3D, bir oyun geliştirme motorudur. Oyun ve bilişim dünyasına getirdiği en önemli yenilik, gelişmiş özelliklere sahip 3 boyutlu oyunların bilgisayara kurulmadan oynanmasını sağlamak olmuştur. Unity3D motorunu kullanan oyunlar, Unity Web Player eklentisi sayesinde hiçbir kurulum işlemi olmadan web tarayıcı üzerinden

çalışabilmektedir. Bu yönü sayesinde oyunlar bilgisayara daha az yüklenirken bir oyuncuyu yeni bilgisayar alma zahmetinden kurtarmış olur. Unity'nin diğer oyun motorlarından üstün taraflarından biri de oyun geliştirme zamanında geliştiriciye program kodu yazma olanağı vermesidir. Diğer oyun motorlarının ekserisi grafik ile kodu ayırmışken, Unity ile grafik ve kod birlikte çalışmaktadır. Bu çalışma mantığı geliştiriciye esneklik sağlamakta, geliştirme süresini kısaltmaktadır.

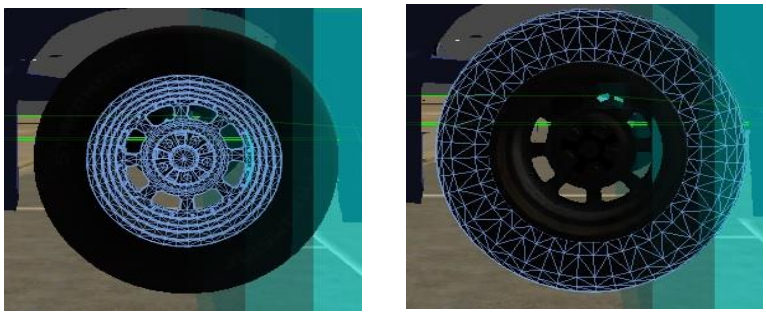
Çalışmanın ilk kısmını oluşturan oyun nesnelerin tasarımında oyun motoru olarak Unity3D kullanılmıştır. Unity3D'nin tercih edilmesinin başlıca sebepleri: betik dili ile programlanabilir olması, kendisinin çeşitli işletim sistemlerinde çalışmasını yanı sıra kendisi ile üretilen oyunların yine çeşitli işletim sistemlerinde çalıştırılır sürümlerinin üretilmesinin kolay olmasıdır [24]. Ayrıca Unity3D kullanan geliştirici sayısı fazla olduğu için bunların ürettikleri çok sayıdaki dokümanları ve eğitim videolarını internet üzerinden kolayca edinmek mümkündür.

6.1.2. Araç modeli

Araç modeli geliştirme aşamasında yapılmış Smart AI Car gibi örnekler incelenmiş, açık kaynak özellikleri olanlardan yararlanılmış ve uygulamanın gerektirdiği yeterlilikte bir araç modeli geliştirilmiştir [23]. Bu araç modeli, kapının oyuncu tarafından açılabilmesi gibi bir arabanın tüm mekanik özelliklerinin karşılayan bir model değildir.

Unity3D de bir nesne modeli 3 bölümden oluşmaktadır. Bu bölümler; katı cisim modeli, dönüşüm modeli, çarpışma modelidir. Bir karmaşık yapılı oyun nesnesi

oluşturmak için parçalardan bütüne doğru bir tasarım aşaması uygulanır. Bu yöntemde, oyun nesnesinin tüm bileşenleri teker teker oluşturulur daha sonra bu bileşenler birbiri ile ilişkilendirilir [24]. Örnek olarak arabanın bir alt bileşeni olarak gerçeklenen tekerleklerin yine aynı yöntemle kendisinin oluşturulmasını gösterirsek; bir tekerleğin öncelikle iç bileşenleri yani mekanik kısmı katı cisim olarak gerçeklenir, daha sonra dış bileşeni yani lastiğin kaplanması aşaması gerçeklenir ("Şekil 6.1."). Oluşturulan bu tekerlek için tasarlanan dönüşüm modeli (tekerleğin dönerek yol alması) ile yukarıda tanıtılan katı cisim modeli birleştirilerek bir oyun nesnesi oluşturulmaktadır. Bu nesne diğer bir nesne ile çarpıştığında davranışı programlanarak işlem tamamlanır.



Şekil 6.1. İç ve dış bileşenler

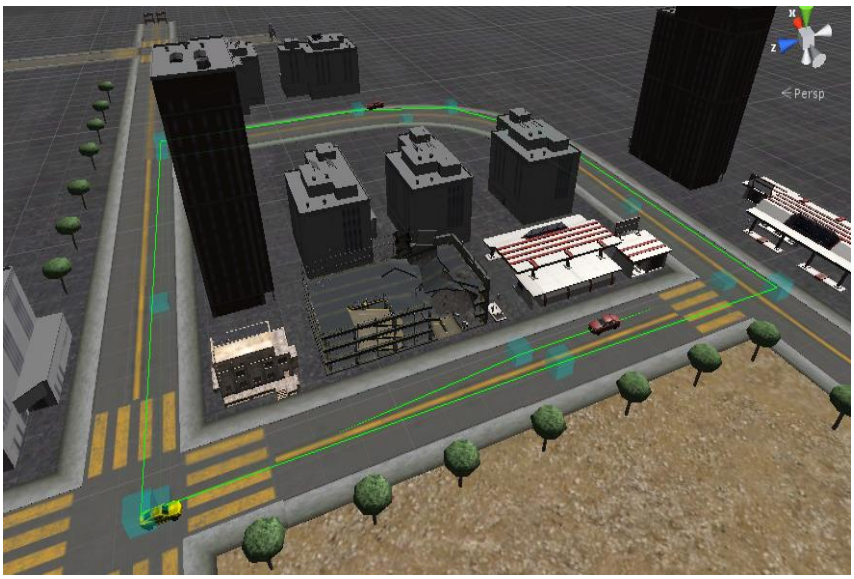
Oyun nesnesi oluşturulduktan sonra nesnenin katı cisim modeli ile özelliklerinin atanması yapılır. Araç nesnesi için katı cisim modelinde fiziksel özellikler, arabanın kütlesi, sürtünme katsayısı, yerçekimi, hızı gibi atamalar gerçekleştirilmiştir.

Otonom araçlarda kullanılan sensörlerden biri olan LIDAR sensörü sayesinde tanımlanmış bölge içinde var olan diğer nesnelerin varlığı ve bunların araçtan uzaklığı

belirlenir [17][25]. Projede ise bu yöntemin daha az ışın kullanan benzeri olarak ışın kesme sensörleri ile bir engel sezme yeteneği oluşturuldu. Projede aracın yanlarında ve önünde olmak üzere ışın kesme sensörleri yaratıldı. Böylelikle sürücüsüz araç belirlenen mesafede olan nesneleri algılayabilmektedir. Sürücüsüz araç geliştirilen senaryolar yardımı ile örneğin önünde bir engel var ise hızını yavaşlatıp manevralar yaparak engeli geçebilir ya da tüm hareket alanını engelliyor ise engel önünden kalkana kadar durabilir.

6.1.3. Yerleşke modeli

Sahneye yerleştirilen düzlem proje için tasarlanan yol modeli eklendi. Buradaki en büyük sorun sürücüsüz aracın yolu tanımasını sağlanması ile yol çizgilerini takip edip hareketi boyunca yolun dışına çıkmamasıdır. Kullanılan basit yöntemlerde araç hareketini güzergah üzerine konulan işaret noktalarına göre gerçekleştirir. Yerleştirilen işaret noktaları her ne kadar yolun üzerinde dahi olsa araç virajlardan dönerken aldığı momentle hareketini gerçekleştirirken ya da bir engelle karşılaştığında o engeli geçerken yolun dışına çıkabilir. Genelde, binalar ve diğer görünür engeller yoldan çıkılmaması için kullanılır. Bunların olmadığı durumlarda ise sorunu çözmek için yol kenarlarına sadece sürücüsüz araçlarının etkilendiği görünmez engeller eklendi (“Şekil 6.2.”). Araç, ışın kesme sensörü sayesinde yol kenarında ki engeli algılayarak yolun dışına çıkmamasını sağlayacak hız denetimlerini dinamik olarak yapar.



Şekil 6.2. Güzergah işaretleme ve yol kenarı engelleri

6.1.4. Kullanıcı arayüzü

Kullanıcıya basit bir arayüz sunulmaktadır. Farklı ortamlarda yaygın kullanılması düşünüldüğünden oyun denetimleri basit tutulmuştur. Ayrıca mobil ortamlar için tek ekranın kullanılmıştır. Bu nedenle ekranın araç göstergeleri ile donatılmasına yerine mümkün olduğunda yol görüntüsüne ayrılması planlanmıştır (Şekil 6.3.). Normal bir sürücünün sadece yola bakması durumu yaratılmıştır. Sürücüye hedeflerin atanması ve sürücünün güzergahta yolunu bulması için basit bir navigasyon olarak kavşak öncesi yön okları çıkmaktadır.

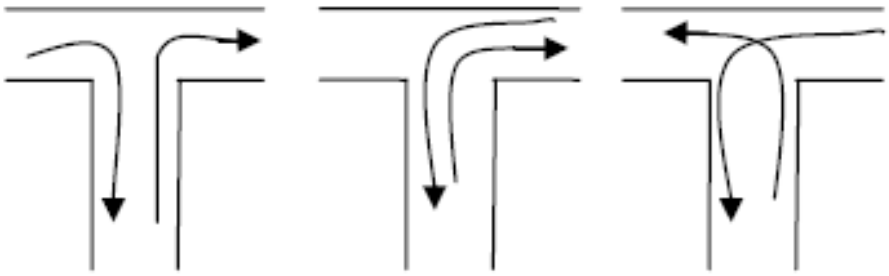
Şekil 6.3. Sahnenin genel görünümü

Oyunda kullanılan tuş kombinasyonları; “Q” ve “E” tuşları sürücülü aracın sinyal ışıklarını yakıp söndürme, “Space” tuşu sürücülü araç için el freni, “A” ve “Z” otonom araçların hızının arttırılıp azaltılması, “yukarı yön” tuşu sürücülü araç için gaz, “aşağı yön” tuşu ise sürücülü aracın hızının azaltılması ve geri vites, “sağ” ve “sol” yön tuşları aracın yönünü değiştirme, “C” tuşu otonom araçların kameralar arası geçişi ve “Esc” tuşu ise oyunun bitirilmesini sağlayan tuş atamalarıdır.

6.2. OKANOM Senaryosu

Çalışma kavşak tabanlı olduğundan bu bölümde örnek bir kavşak senaryosu problemi tanıtılacaktır.

T kavşak senaryosunun türevleri "Şekil 6.4." de gösterilmektedir. Birinci senaryo adımında (Şekil 6.4.(a)) iki aracın bir T kavşakta karşılaşıp farklı yönlere doğru hareketleri söz konusudur [19]. Buradaki hedef araçların sorunsuz şekilde kendi mevcut şeritlerini takip ederek birbirlerini çarpmadan ilgili yönlere yönelmeleridir. Bu senaryoda temel sorun geçiş sırasında araçlar birbirine çarpmamasına rağmen tam köşede geçiş sırasında çok yaklaşmaları nedeni birbirilerini engellemeleridir. Araçlardan biri sürücülü ise sürücü bu durumdan kendini kurtarabilir. Ancak 2 otonom araç birbirini kilitleyecektir.



(a) (b) (c)

Şekil 6.4. T kavşakta geçiş türleri[7]

[7] Özgüner, Ü., Acarman, T., Redmill, K., Autonomous Ground Vehicles, Artech House, USA, 2011

“Şekil 6.4.(b)”de gösterilen senaryoda, T kavşakta iki aracın da aynı dönüş noktasını kullanarak diğerinin geldiği yöne doğru hareketi sağlandı. Buradaki sorun önceki örnekteki karşılıklı engelleme sorunun tüm geçiş boyunca yer almasıdır. Kavşak geçişlerinde uygun ışın kesme açıları kullanılarak sorun çözülür. “Şekil 6.4.(c)”de ise araçların geçiş şeritleri kesişmektedir. Yola önce girenin geçiş üstünlüğünü alarak kaza ve karşılıklı engellenme olmaması sorunu çözüme kavuşturuldu.

7. 2B SÜRÜŞ SİMÜLASYON UYGULAMASI

Bu çalışma; OKAN Üniversitesi Ulaştırma Teknolojileri ve Akıllı Otomotiv Sistemleri Uygulama ve Araştırma Merkezi tarafından gerçekleştirilen otonom araç projesi ile elde edilen bilgi birikiminin kullanılarak, otonom aracın sanal trafik içinde belirlenen senaryolardaki davranışlarını inceleme fikri ile ortaya çıkmıştır.

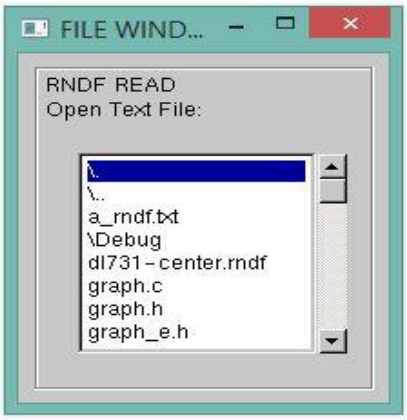
Otonom araçların en büyük problemlerinden biri olan güzergah takibinin DARPA tarafından hazırlanmış olan RNDF ve MDF dosya yapılarını yorumlayarak, otonom aracın verilen hedefe, algoritmanın belirlemiş olduğu güzergah üzerinden ulaşmasını sağlamaktır.

Çalışma öncelikle Unity3D ortamında üç boyutlu bir oyun olarak başladı. Fakat daha sonra otonom aracın güzergah takibini Unity3D'nin sağladığı şekilde menülerden yapmak, çalışmamızın ilerleyen aşamalarını kısıtlandırdı. Bu sebeple RNDF ve MDF dosyalarını kullanarak aracın adım adım hareketini inceleyebileceğiz iki boyutlu bir uygulama geliştirdik. Bu uygulama sayesinde aracın güzergah takibi ile ilgili testlerini gerçekleştirildi. Daha sonra iki boyutlu sürüş simülasyonundan elde edilen yol noktaları text dosya halinde Unity3D üzerinde uygulanabilir bir hale dönüştürülmüştür.

7.1. Uygulamanın Tanıtımı

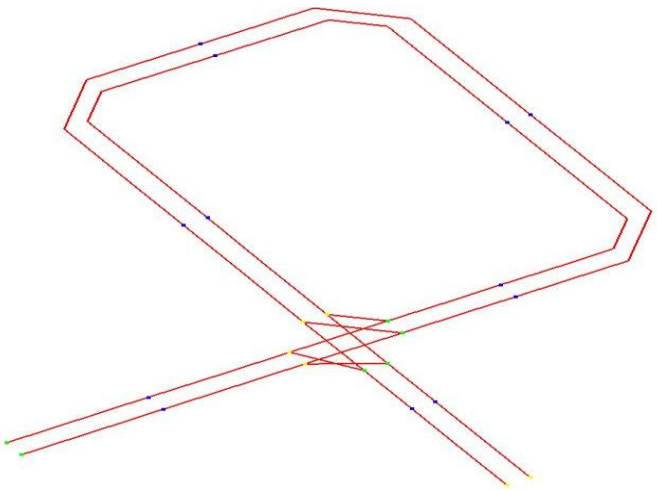
Bu bölümde uygulamanın ekran görüntüleri yardımıyla anlatımı yapılacaktır.

Uygulama çalıştırıldığında ilk olarak açılan ekran kullanıcının dosya seçimi yapabilmesine imkân sağlayan "File Window" sayfasıdır.



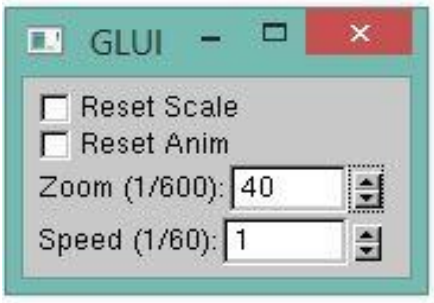
Şekil 7.1. File window ekranı

Kullanıcı bu ekran sayesinde var olan RNDF dosyalarından birini seçerek harita bilgilerini uygulamaya vermiş olur (Şekil 7.1.).



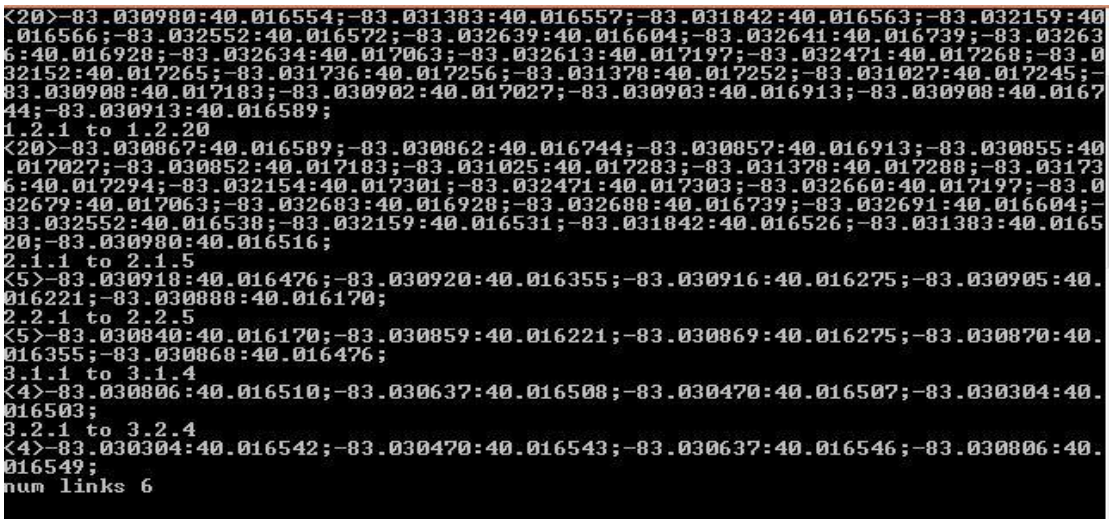
Şekil 7.2. Harita ekranı

Kullanıcı dosyayı seçtikten sonra RNDF dosyasında bilgilerin yorumlanması ile oluşturulan haritanın çizilmiş hali yeni bir ekranda açılmaktadır. “Şekil 7.2.” de görülen ekran örnek bir harita dosyasının seçilmesi ile elde edilmiştir.



Şekil 7.3. Harita araç ekranı

“Şekil 7.3.” de harita üzerindeki çeşitli işlemleri gerçekleştirmek için kullanılan harita araç ekranı görülmektedir. Bu ekrandaki “Zoom” özelliği sayesinde kullanıcı haritayı yakınlaştırıp, uzaklaştırabilmektedir. “Speed” özelliği sayesinde harita üzerinde hareket eden aracın hızını ayarlayabilmektedir. “Reset Scale” özelliği sayesinde haritayı ilk konumuna alabilmektedir. “Reset Anim” özelliği ise aracın başlangıç noktasına dönmesini ve yeniden animasyonun başlamasını sağlamaktadır.

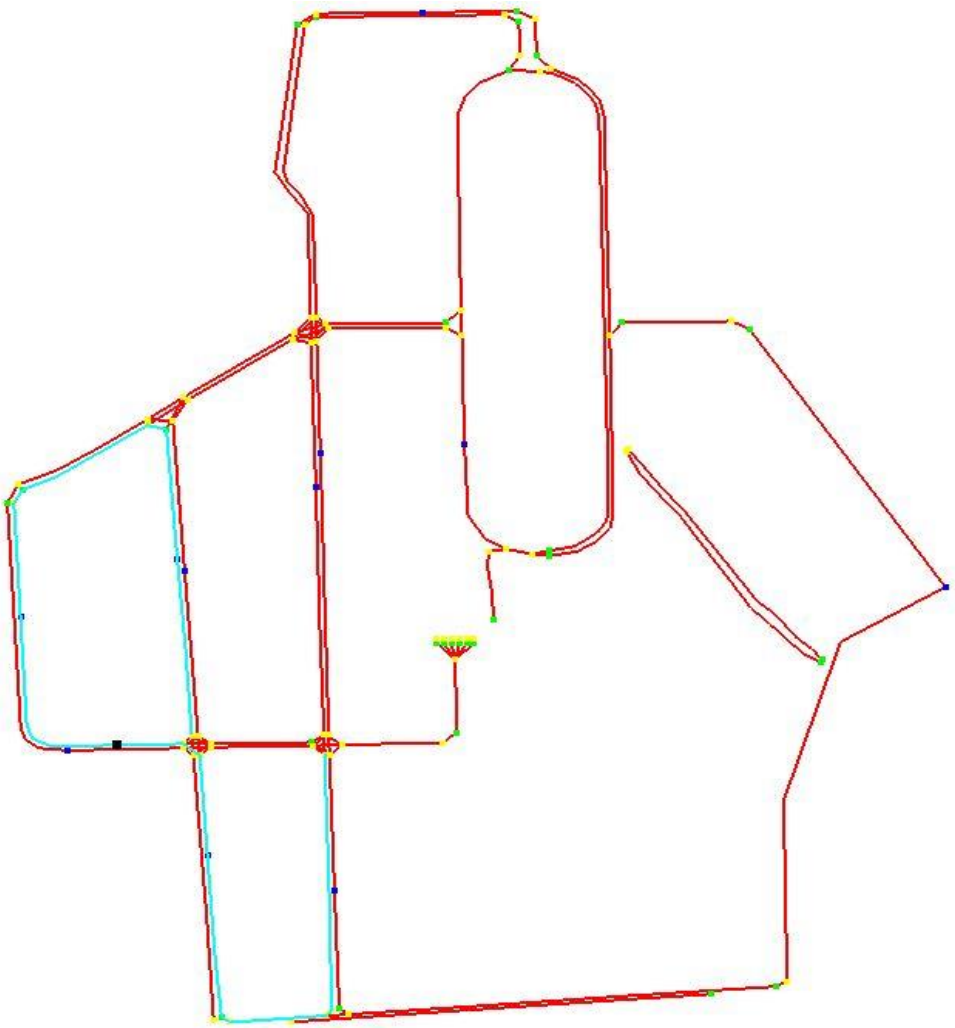


Şekil 7.4. Güzergah takip ekranı

Uygulama üzerinde hareket eden aracın takip ettiği güzergahın şerit isimleri ve yol noktalarının koordinat bilgilerinin bulunduğu ekran "Şekil 7.4." de görülmektedir. Bu ekran sayesinde aracın güzergah takibi kolayca yapılabilmektedir.

7.2. Uygulama Örnek Senaryosu

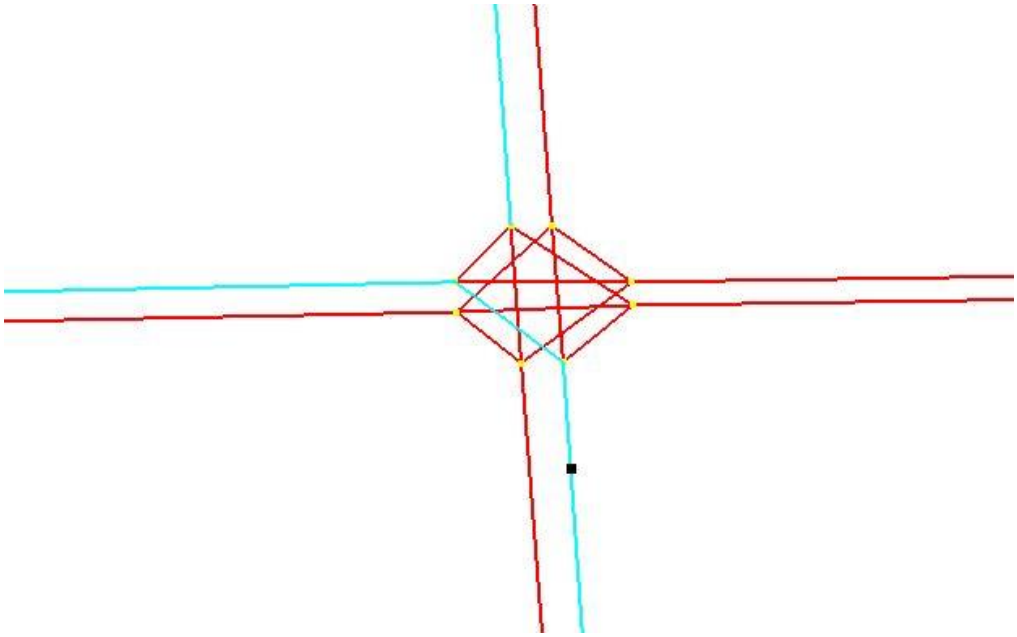
Bu bölümde uygulama üzerinde gerçekleştirilen örnek bir senaryonun açıklaması yapılacaktır. Örnek senaryodaki harita için "*Sample.RNDF*" dosyası kullanılmıştır [18].



Şekil 7.5. Örnek senaryo haritası

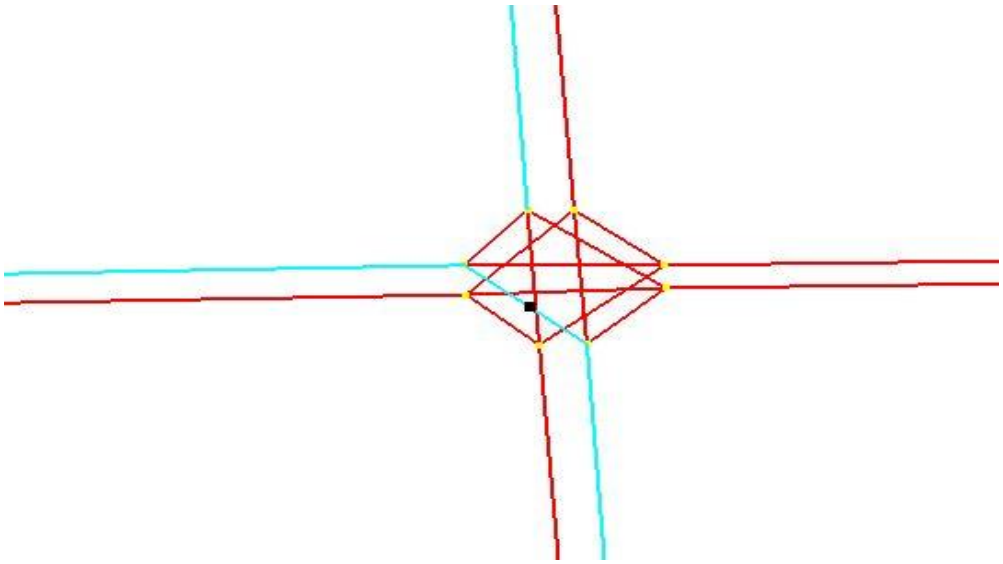
Senaryo için kullanılan harita “Şekil 7.5.” de görülmektedir. Bu haritaya göre kırmızı renk ile gösterilen çizgiler harita üzerindeki şeritleri, sarı noktalar şeritlerin başlangıç noktalarını, yeşil noktalar şeritlerin bitiş noktalarını, mavi noktalar ise şeritler üzerinde bulunan kontrol noktalarını göstermektedir. Harita üzerinde turkuaz renk ile gösterilen şeritler otonom aracın takip edeceği hedef güzergahı göstermektedir.

Araçtan harita üzerinde tanımlanmış olan “Indiana_Road” dan “Central_Parking_Lot” hedefine ulaşması istenmiştir. “Şekil 7.5.” de görülen turkuaz renkli güzergah bu başlangıç ve hedef noktalara göre oluşturulmuş bir güzergahtır. Bu güzergah boyunca otonom aracı temsil eden siyah nesne çeşitli yol parçalarını geçmiştir.



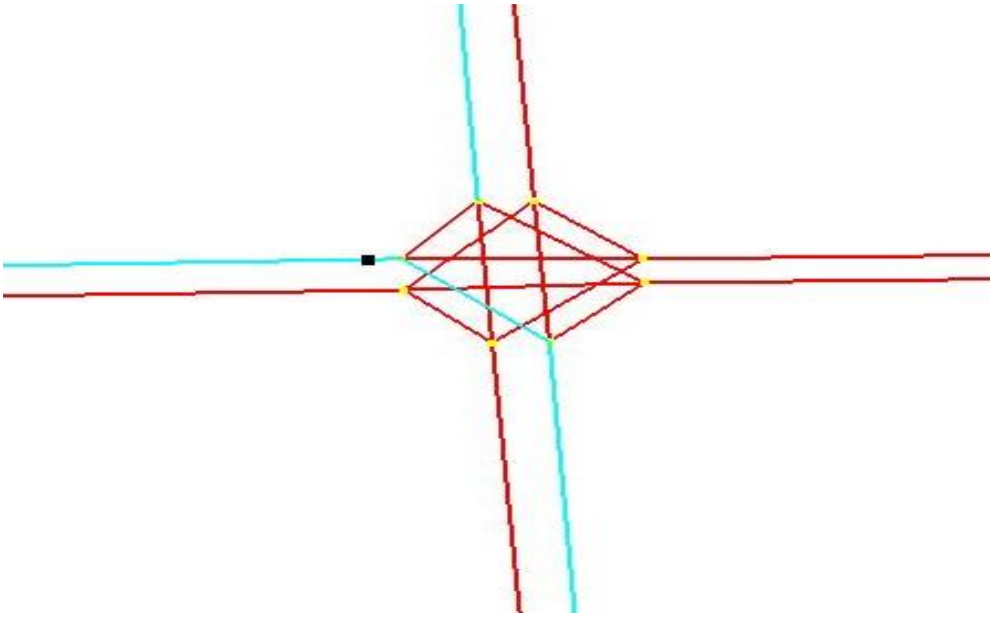
Şekil 7.6. Aracın kavşağa giriş durumu

“Şekil 7.6.” da aracın güzergah üzerindeki bir kavşağa giriş durumu görülmektedir.



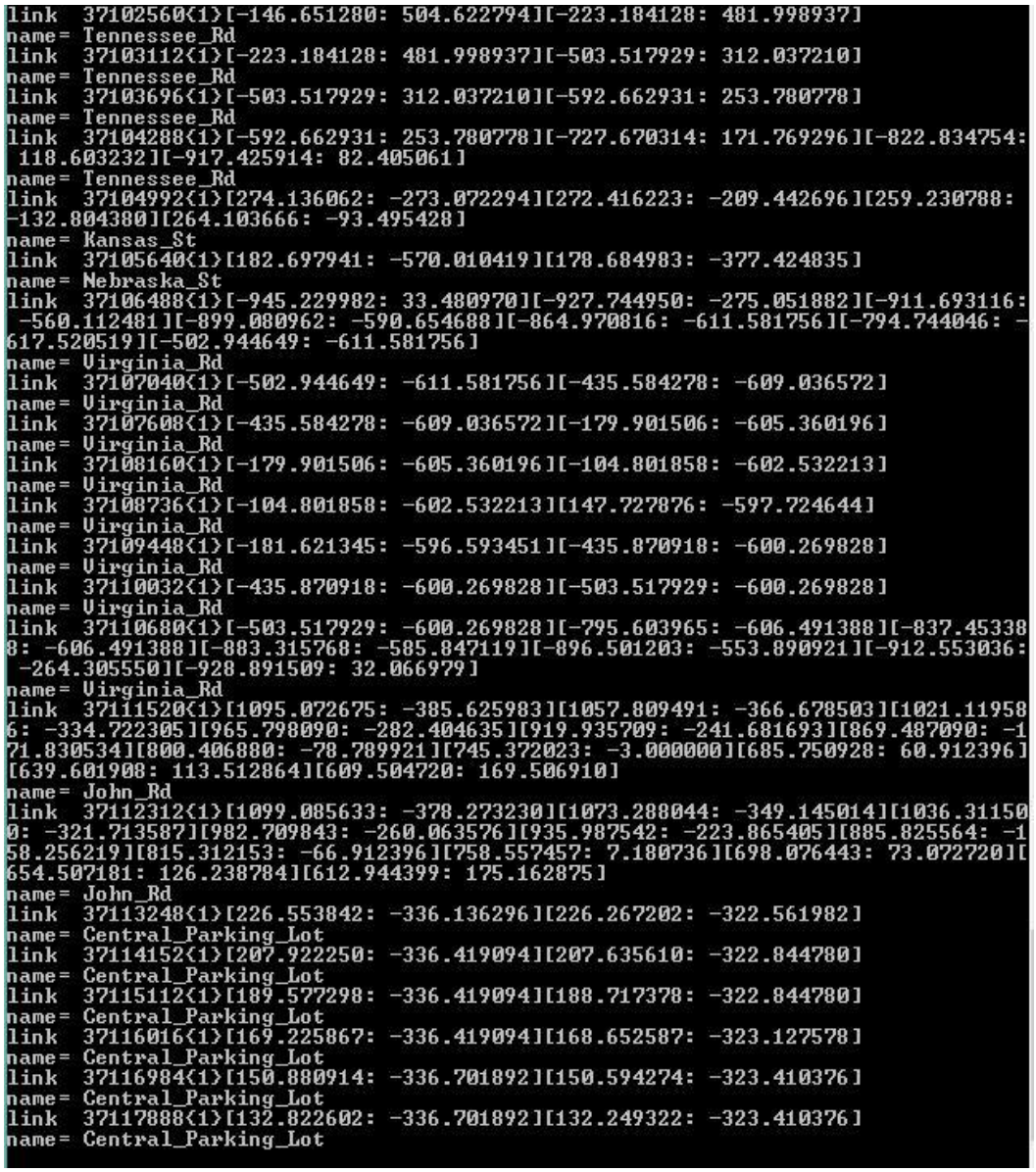
Şekil 7.7. Aracın kavşaktaki durumu

“Şekil 7.7” de aracın güzergah üzerindeki bir kavşaktaki durumu görülmektedir.



Şekil 7.8. Aracın kavşaktan çıkış durumu

“Şekil 7.8.” de aracın güzergah üzerindeki bir kavşaktan çıkış durumu görülmektedir.



Şekil 7.9. Aracın güzergah bilgileri

Aracın geçtiği güzergah bilgi ekranı ise “Şekil 7.9.” da görülmektedir.

7.3. Yol Bilgilerinden Graf Oluşturma

RNDF dosyası içerisinde sadece harita noktalarını tanımları bulunmaktadır. Fakat harita üzerindeki en önemli problemlerden biri kavşakların birbirine nasıl bağlanacağıdır. RNDF dosyası bu bilgiyi tutmamaktadır. Sadece harita noktaları bilinerek bir gezinme sağlamak mümkün değildir. Bu sebeple RNDF dosya bilgileri ile bir graf üretilmeli ve bu grafın üzerinde yol bulma algoritmaları uygulanmalıdır.
RNDF yapısı üzerindeki segment parçaları grafın düğümlerini oluşturmakta, kavşakların ve segment parçalarının bağlantısıda grafın ayrıtlarını oluşturmaktadır. Böylelikle RNDF dosyası yorumlanarak, algoritmaların üzerinde işletilebileceği bir graf yapısı elde edilmektedir.

7.4. Graf Üzerinde Güzergah Belirleme

Verilen iki nokta arasında yol bulma, graf kütüphanesine dahil olmadığından bu çalışma için geliştirilmiştir. Öncelikle yol bulma algoritmasıyla gezilen segment düğümlerden oluşan bir liste elde edilir. Bu liste bir otonom aracın yol üzerinde gitmesi için yeterli değildir. Halbuki gerekli olan, bir güzergaha ait yol noktalarından oluşan bir liste elde etmektir. Düğüme ait yol noktaları bilgileri kullanılarak otonom aracı yönlendirecek bir yol noktaları listesi oluşturulur. Oluşturulan bu liste algoritma da kullanılarak güzergah belirlenmiş olur.

8. UYGULAMANIN GERÇEKLENMESİ

Bu bölümde uygulamayı gerçekleştirmekte kullanılan teknolojiler, uygulamada kullanılan bir algotirma örneği ve uygulamanın yazılım özelliklerinden bahsedilecektir.

8.1. OpenGL

OpenGL, en yaygın kullanılan grafik programlama kütüphanesidir. Hızlı ve basit bir şekilde etkileşimli, 2B-3B bilgisayar grafik programları yapmanıza olanak sağlar. Kullanım alanı çok yaygındır ve bilgisayar grafiklerinin hemen hemen tüm alanlarında yaygın olarak kullanılır. Bazı kullanım alanları: araştırma, bilimsel görselleştirme, eğlence ve görüntü efektleri, bilgisayar destekli tasarım, etkileşimli oyunlar...

OpenGL, donanım-bağımsız bir arayüzdür. Görüntüde bulunan nesneleri tanımlamak ve bu nesneler üzerinde gerek duyulan işlemleri gerçekleştirmek için gerekli komutları içerir. OpenGL'in donanım-bağımsız olmasının nedeni, pencere işlemlerini yapan ya da kullanıcıdan girdi alan herhangi bir komutunun bulunmamasıdır.

OpenGL, 3D nesneleri tanımlamak için yüksek-seviyede komutlar içermez. Bunun yerine; nokta, doğru ve poligon gibi alt-seviye geometrik primitif nesneleri içerir ve bu primitif nesneleri kullanarak karmaşık grafik nesneleri tanımlamamıza olanak sağlar [31].

OpenGL kullanımının sağladığı avantajlar;

1. Platform bağımsızdır (Windows, Linux, Mac) ve tüm OpenGL UPA uyumlu donanımlar üzerinde çalışır.

2. Çok çeşitli sistemler üzerinde koşulabilir. (Kişisel bilgisayarlar, iş istasyonları, süper bilgisayarlar, gömülü sistemler vs.)

3. Sistem kaynaklarını optimum şekilde kullanır.

4. Birçok programlama dili tarafından çağırılarak kullanılabilir.

5. Kolay anlaşılır, hızlı öğrenilir.

8.2. GLUT

OpenGL platformdan bağımsız olduğu için bazı işlemler bu kitaplık ile yapılamaz. Örneğin kullanıcıdan klavye veya fare ile veri almak, bir pencere çizdirmek gibi işler hep kullanılan pencere yöneticisi ve işletim sistemine bağlıdır. Bu yüzden bir an için OpenGL'in platform bağımlı olduğu düşünülebilir. Çünkü çalışma penceresini her pencere yöneticisinde (her ortamda) farklı çizdirecek bir canlandırma programı yazmak demek her bilgisayarda çalışacak ayrı pencere açma kodu yazmak demektir. Bu ise OpenGL'in doğasına aykırıdır. Bu gibi sorunları aşmak için OpenGL Araç Kiti (GLUT - OpenGL Utility Toolkit) kullanılmaktadır [31].

8.3. GUI

GUI (Graphical User Interface) bilgisayar ile etkileşimi görsel mecazlar kullanarak sağlayan yöntemler bütünüdür [34]. GUI, bilgisayar kullanıcılarının komut satırı kodlarını ezberlemeden fare, klavye gibi araçlar sayesinde bilgisayarları kontrol etmelerini sağlamıştır. Günümüzdeki programların birçoğu genel kullanıcı arayüzü ile birlikte gelse de, birçok bilgisayar kullanıcısı daha hızlı olduğu gerekçesiyle komut satırını genel kullanıcı arayüzleriyle birlikte kullanmaya devam etmektedirler [35]. Bu sayede kullanıcının önceki deneyimlerinden yararlanılarak bilgisayarı rahat kullanması sağlanmaktadır [34].

8.4. Uygulamada Kullanılan Bir Algoritma

İki boyutlu uygulama tarafımızdan geliştirilmiş ancak grafı oluşturan bazı parçalar ile veri yapıları ve temel kütüphaneleri OSU tarafından geliştirilen bir çalışmadan alınmıştır. Kaynak olarak alınan veri yapılarının, fonksiyonların üzerinde gerekli değişiklikler yapılarak kullanılması sağlanmıştır[32].

Uygulamada graf üzerindeki aracın güzergahını belirlemek için kullanılan algoritma "Derinlik Öncelikli Dolaşma" algoritmasıdır. Bu algoritmanın kullanılma nedeni; RNDF dosyasındaki bilgi yapısına uygunluğu ve algoritmanın kolay uygulanabilir olmasıdır.

"Derinlik Öncelikli Dolaşma" algoritmasının uygulamadaki kodu "Tablo 8.1." de gösterilmiştir.

Tablo 8.1. DFS algoritması

```
int graph_mission(Graph *graph,Link *t, Link *e)
{
        int i,j,k,n;
        sp=0;
        gp=0;
        stack[sp++]=t;
        gezinti[gp++]=t;
        n=0;
        for(i=0;i<graph->num_links;i++)
        {
                graph->links[i]->visited=0;
                graph->links[i]->level=0;
        }
        gezinti[0]=t;
        t->visited=1;
        while(1)
        {
                if(sp==0)
                        return -1;
                t=stack[--sp];
```

```
            n=t->level;
            gezinti[n]=t;
            if(t==e)
                  return n;
            for(k=0;k<t->num_dest;k++)
            {
                  if(!(t->dest[k]->visited) )
                  {
                        stack[sp++] = t->dest[k];
                        t->dest[k]->visited=1;
                        t->dest[k]->level=n+1;
                  }
            }
      }
      return -1;    }
```

"Tablo 8.1."de görülen "graph_mission" fonksiyonunda yığın DFS algoritmasını gerçekleştirmek için kullanılmıştır. "Gezinti " isimli dizi ise gezilen düğüm bilgilerin tutulduğu dizidir. Ayrıca algoritmaya göre gezilen düğümlerinin seviye bilgisinin tutulmasına ihtiyaç vardır. Hangi seviye komşulukta olduğunu bulmak için. Bu sebeple fonksiyonda "level" bilgisininde olduğu görülmektedir.

8.5. Uygulamanın Yazılım Özellikleri

Bu bölümde uygulamada kullanılan diğer önemli kod parçalarına değinilmiş ve bu kodların işlevlerinden bahsedilmiştir.

Aşağıdaki kod parçasında graf üzerindeki düğümler gezilerek bu düğümlere ait segment adları ve şerit numaları ekrana yazdırılır. Ayrıca, graf üzerinde mevcut olan noktaların x ve y koordinat bigileri verilerek bu noktalar arasına kırmızı renkte ve kalınlığı iki olan çizgiler çizilir. Böylece harita oluşturulur.

```
glLineWidth(2);
        glColor3f(1.0f, 0.0f, 0.0f);
        for(i=0;i<graph->num_links;i++) {
                printf("link  %d", graph->links[i]->segment_name);
                printf("{%d}",graph->links[i]->num_lanes);
                for(j=0;j<graph->links[i]->num_lanes;j++) {
                        glBegin(GL_LINE_STRIP);
                        for(k=0;k<graph->links[i]->lanes[j].num_points;k++) {
                                convert_lat_lon_to_m(graph->links[i]
>lanes[j].lat[k],graph->links[i]->lanes[j].lng[k]);
                                glVertex3f(x, y, 0.0);
                                printf("[%f: %f]",x,y); }
```

Aşağıda görülen kod parçasında ile kavşak üzerindeki bağlantıların yapılması sağlanmıştır. Kavşaklar sayesinde bir şerit ilgili olduğu diğer şeritler ile bağlanmıştır.

```
for(k=0;k<graph->links[i]->num_dest;k++){
        convert_lat_lon_to_m( graph->links[i]->lanes[0].lat[graph->links[i]->lanes[0].num_points-1], graph->links[i]->lanes[0].lng[graph->links[i]->lanes[j].num_points-1]);
        glVertex3f(x, y, 0.0);
        convert_lat_lon_to_m(graph->links[i]->dest[k]->lanes[0].lat[0],graph->links[i]->dest[k]->lanes[0].lng[0]);
                glVertex3f(x, y, 0.0);
```

Aşağıdaki kod parçasında öncelikle graf üzerinde var olan tüm noktaların point_info bilgilerine bakılır. Point_info bilgisinde “Checkpoint” bilgisini taşıyan noktalar mavi renk ile boyanır. Böylece harita üzerindeki kontrol noktalarına erişilmiş olunur.

```
glColor3f(0.0f, 0.0f, 1.0f);
        glPointSize(4);
        for(i=0;i<graph->num_links;i++)
        {
                for(j=0;j<graph->links[i]->num_lanes;j++)
                {
```

```
            glBegin(GL_POINTS);
        for(k=0; k<graph->links[i]->lanes[j].num_points; k++)
            if (graph->links[i]->lanes[j].point_info[k]==CHECKPOINT)
            {
            convert_lat_lon_to_m(graph->links[i]->lanes[j].lat[k],graph-
>links[i]->lanes[j].lng[k]);
            glVertex3f(x, y, 0.0);
            }
         glEnd();
        }
        }
```

Aşağıdaki kod parçasında verilen hedef güzergah üzerinde otonom aracı temsil eden siyah renkli şeklin çizilmesi ve bu şeklin x ve y değerlerinde hareketini sağlayan kod parçası görülmektedir.

```
if (anim) {
        glColor3f(0.0f, 0.0f, 0.0f);
glPointSize(6);
if(step>=nstep){
dotcurr++;
if(dotcurr==dotn)anim=0;
```

```
else{
x0=dotx[dotcurr-1];
yy0=doty[dotcurr-1];
x1=dotx[dotcurr];
yy1=doty[dotcurr];
if(abs(x1-x0)>abs(yy1-yy0))
{
        xn=(x1-x0)/abs(x1-x0);
        yyn=(yy1-yy0)/abs(x1-x0);
        nstep=(abs(x1-x0));
}
else
{
        xn=(x1-x0)/abs(yy1-yy0);
        yyn=(yy1-yy0)/abs(yy1-yy0);
        nstep=(abs(yy1-yy0));
}
}
step=0;
}
if (anim) {
        glBegin(GL_POINTS);
```

```
        glVertex3f(x0, yy0, 0.0);
        step+=segments2;
                x0=xn*segments2+x0;
                yy0=yyn*segments2+yy0;
         glEnd();
glPointSize(4);
}
}
```

Aşağıda görülen kod parçasında “a” ile gösterilen başlangıç noktasından verilen “b” hedef noktasına bir bağlantı olup olmadığı kontrol ediliyor. Eğer “gn” değeri sıfırdan büyük bir değer ise belirtilen noktalar arasında bir bağlantı yani yol vardır ve o zaman ekrana “Link bulundu” yazılıp gezinti listesi içindeki bilgi ekrana yazdırılır. “Gn” değeri sıfırdan büyük değil ise bu durumda bu verilen noktalar arasında bir bağlantı yoktur yani yol yoktur anlamına gelmektedir.

```
gn= graph_mission(graph,graph->links[a],graph->links[b]);
if(gn>0){
        printf("\n Buldu");
         for(i=0; i<=gn; i++)
         {
```

```
                printf(" \n Link %s %f %f ---- ", gezinti[i]->segment_name,gezinti[i]-
>lanes[0].lat[0],gezinti[i]->lanes[0].lng[0]);
        }
 }
else printf("\n  Link Bulunamadı");
```

9. SONUÇLAR

Bu çalışmanın birinci bölümünde araç ve trafik simülasyonunun bir ciddi oyun olarak uygulanmasındaki aşamalar irdelenmiştir. Bir oyun hedefi belirlenerek ve bu hedef doğrultusunda simülasyon gereksinimleri azaltılarak seçilen bir oyun motoru Unity3D ile hedef gerçeklenmiştir.

Çalışmanın ikinci bölümünde DARPA tarafından oluşturulan RNDF ve MDF dosyalarındaki veriler uygulama tarafından başarı ile iki boyutlu uygulamaya dahil edilmiştir. Bu veriler uygulama tarafından bir graf yapısı haline dönüştürülerek bir harita elde edilmiştir. Simülasyon isterleri karşılanmış ve başlangıç noktası belirtilen bir aracın istenilen hedefe ulaşması "Derinlik Öncelikli Dolaşma" algoritmasıyla gerçeklenmesi sağlanmıştır.

Sonuçta iki boyutlu uygulamanın hem yerleşkenin oluşturulması hemde araçların otonom olarak hareketinin sağlanması için Unity3D ortamında plug-in olarak tasarlanması yüksek lisans tez kapsamının dışında tutulmuş. Bu konu sonraki bir çalışma olarak planlanmıştır.

10. KAYNAKLAR

[1] http://www.acikbilim.com/2012/02/dosyalar/surucusuz-arabalar.html (Ziyaret tarihi Ocak 2015)

[2] http://spectrum.ieee.org/cars-that-think/transportation/self-driving/ (Ziyaret tarihi Ocak 2015)

[3] Bayarri, S., Fernandez, M., Perez, M. "Virtual reality for driving simulation."*ACM 39, 1996*, 72-76

[4] http://sumo-sim.org/ (Ziyaret tarihi Ocak 2015)

[5] Pursula, M. "Simulation of traffic systems-an overview." *Journal of Geographic Information and Decision Analysis,1999*, 1-8.

[6] http://www.carnetsoft.com/(Ziyaret tarihi Ocak 2015)

[7] http://citycardriving.com/ (Ziyaret tarihi Ocak 2015)

[8] http://www.unity3dgames.eu/traffic-talent.html (Ziyaret tarihi Ocak 2015)

[9] http://drivesim.osu.edu/about/ (Ziyaret tarihi Ocak 2015)

[10] http://citr.osu.edu/CrIS/?page_id=362 (Ziyaret tarihi Ocak 2015)

[11] Lee,S., Cho, J., and Young Ju, D. "Autonomous Vehicle Simulation Project." *International Journal of Software Engineering and Its Applications Vol.7, 2013*, 393-402

[12] Kaçtıoğlu,S.,Kılağız,Y., "Yapay Zeka Bilgi İşlem Teknolojisive Bileşenleri" *İktisadi ve İdari Bilimler Dergisi, Cilt: 14, Haziran 2000 Sayı: 1*, 357-375

[13] Küçükceylan, O., Yüksel, T., Sezgin,A., "Enine arama algoritmasını kullanarak en kısa yol probleminin çözümünün lego mındstorm ile gerçeklenmesi" *IV. Otomasyon Sempozyumu, Samsun, 2007,* 25 – 29

[14] Nabiyev, Vasfi V.,"Algoritmalar Teoriden Uygulamalara", *Seçkin Yayınevi, 2007, Ankara, Türkiye*

[15] Dener, M., Akçayol, M.A., Toklu, S., Bay, Ö.F., "Zamana Bağlı Dinamik En Kısa Yol Problemi İçin Genetik Algoritma Tabanlı Yeni Bir Algoritma" *Gazi Üniv. Müh. Mim. Fak. Der. Cilt 26, 2011,* 915-928

[16] Çölkesen, R.,"Veri Yapıları ve Algoritmalar", *Papatya Yayıncılık 2003,* 348-349

[17] Akgün, T., Koç, Z., Güner, Ş., Öztürk, B., Özkan, B., Üstün, Ö., Tuncay, N., Özgüner, Ü., "A Study on Autonomous Vehicle Development Process at Okan University", *2012 IEEE International Conference on Vehicular Electronics and Safety, 2012,* 369-374

[18] DARPA Urban Challenge Route Network Definition File and Mission Data File Formats, 2007,1-14

[19] Özgüner, Ü., Acarman, T., Redmill, K., "Autonomous Ground Vehicles", *Artech House, USA, 2011*

[20] Fu, L., Yazıcı,A.,Özgüner,Ü., "Route Planning For OSU-ACT Autonomous Vehicle in DARPA Urban Challenge", *IEEE Intelligent Vehicles Symposium, 2008,* 781-786

[21] Yazıcı,A.,Özgüner,Ü., Fu, L., "OSU-ACT Otonom Otomobili için sehir içi yol planlaması" *12. Elektrik Elektronik Bilgisayar Biyomedikal Mühendisliği Ulusal Kongresi, 2007,*

[22] http://www.darpa.mil/default.aspx (Ziyaret tarihi Ocak 2015)

[23] http://u3d.as/content/bone-cracker-games/smart-ai-car-2-1/6gu (Ziyaret tarihi Ocak 2015)

[24] http://unity3d.com/ (Ziyaret tarihi Ocak 2015)

[25] http://www.sick.com/ (Ziyaret tarihi Ocak 2015)

[26] Barbehenn, M., 1998, "A Note on the Complexity of Dijkstra's Algorithm for Graphs with Weighted Vertices", IEEE Transactions on Computers, vol. 47, No: 2.

[27] Johnsonbaugh, Richard, 2005, "Discrete Mathematics", Pearson Prentice Hall Publisher, 6. Baskı.

[28] Darkridge,http://www.darkridge.com/~jpr5/archive/alg/node88.html, (Ziyaret tarihi Ocak 2015)

[29] http://bilgisayarkavramlari.sadievrenseker.com/2009/03/02/a-yildiz-arama-algoritmasi-a-star-search-algorithm-a/ (Ziyaret tarihi Ocak 2015)

[30] http://archive.darpa.mil/grandchallenge/ (Ziyaret tarihi Ocak 2015)

[31] KTÜ Bilgisayar Mühendisliği Bölümü – Bilgisayar Grafikleri Laboratuarı Notları,

[32] http://www2.ece.ohio-state.edu/citr/Demo97/osu-av.html (Ziyaret tarihi Ocak 2015)

[33] http://www.trimble.com/Corporate/About_History.aspx (Ziyaret tarihi Ocak 2015)

[34] http://seminer.linux.org.tr/wp-content/uploads/linuxgui.pdf (Ziyaret tarihi Ocak 2015)

[35] https://yaydin.wordpress.com/2011/02/22/guigraphical-user-interface-nedir/ (Ziyaret tarihi Ocak 2015)

EKLER

EK-A

Sürücüsüz Araç Trafiği ile Sürücü Davranışı İyileştirmeye Yönelik Bir Oyun: OKANOM

OKANOM: A Game Aimed at The Improvement of Drivers' Attitudes with Using The Autonomous Vehicle Traffic

Kutay Ata ŞEN[1], Kader NİKBAY[1], B. Tevfik AKGÜN[1]

[1]Bilgisayar Mühendisliği Bölümü
OKAN Üniversitesi
kutayatasen@gmail.com, kader.nikbay@okan.edu.tr, tevfik.akgun@okan.edu.tr

Özet

Bu çalışmada, bir 3 boyutlu oyun ortamı oluşturularak sürücüsüz araçların varlığı ile zenginleştirilmiş bir sanal trafikte sürüş deneyimleri ile sürücü davranışlarını ölçülmesi ve iyileştirilmesi hedeflenmiştir. Bu çok-kullanıcılı ciddi oyun için sürücülü ve sürücüsüz araç modelleri, yollar ve binalar içeren yerleşkeler geliştirilmiştir. Bu bildiride, çalışmanın gerçeklenmesi, simülasyon sistemleri ve oyunlar, araç/sürüş/trafik simülasyon sistemleri ve simülatörleri ve oyun seviyeleri özetlenmiştir.

Abstract

In this work, the measurement and improvement of drivers' attitudes are aimed by using driving experiences in virtual traffic which is created by the 3D game platform and is enriched by including autonomous vehicles. For this multi-user serious game, vehicles for drivers, driverless vehicles, roads and locations including buildings are developed. In this paper, development of work, simulation systems and games, simulation systems and simulators of vehicle/driving/traffic and levels of game are summarized.

1. Giriş

Bu çalışmada 3 boyutlu bir oyun kullanarak; sürücülü ve sürücüsüz (otonom) araçların birlikte bulunduğu bir sanal trafik ortamında sürücülerin (oyuncuların) verilen hedefleri gerçeklemesi sırasındaki sürücü davranışlarının incelenmesi ve iyileştirilmesi hedeflenmektedir. OKANOM olarak adlandırılan oyun çok oyunculu bir yapıda tasarlanmıştır. OKANOM ortamında oyuncular kendilerine atanan hedeflere (yerleşkelere) ulaşırken yaratılan bir trafikte yer almaktadır. Bu trafikte diğer oyuncuların denetlediği araçların yanı sıra sürücüsüz araçlar da yer almaktadır. Sürücüsüz araçlar kurallara uygun hareket etmektedirler. Sürücüsüz araçların varlığının ve sayısının mevcut trafiği ne derece de etkilediği ve sürücülerin sürücüsüz araçlara ve diğer sürücülü araçlara olan

davranışlarının nasıl değiştiği araştırılacaktır. Beklenen ya da istenen sonuç; trafikte kurallara uygun seyreden sürücüsüz araçların sayısının belli bir oranda artmasının trafiğin daha iyi akmasına ve kullanıcı davranışlarının gelişmesine destek vermesidir. Ancak sürücülerin sürücüsüz araç davranışlarını öğrendikçe bunları kendi lehlerine istismar etmeleri de mümkündür. Örnek olarak arkasındaki sürücülü aracı görmeden bir sürücüsüz aracı kavşakta zorlayan bir sürücünün kaldığı durumların ve trafikte kendisine benzer şekilde davranan sürücülerin karşılıklı yarattıkları durumlar oluşabilir. Oyunda taksiler sarı renkteki sürücüsüz araçlardır. Oyuncular kendilerine sunulan model ve renklerdeki araçlar ile farklılaşma sağlamaktadır. Burada, gerçek hayattaki aksine düşünülen taksilerin kurallı davranışları ile trafiğe olumlu katkısının olması fikri özellikle seçilmiştir.

Çalışmanın eğitim ve araştırma amaçlı bir ciddi oyun olarak tasarlanmasının temel amaçları: oyun biçiminde sunulan proje hedeflerinin kullanıcılar tarafından benimsemesini kolaylaştırması ve gerçeklemeyi destekleyen oyun motorlarının varlığıdır. Çalışmanın bir oyun olarak tasarlanmasının olumsuz yönü oyuncuların düzgün araç kullanma konusunda özenli olmayabilecekleridir. Oyunun ödüllendirilerek özendirilmesi ya da kötü puanlanmanın oyuncuya olumsuz geri dönüşü bu oyunun belli bir sorumlulukla oynanmasını sağlayabilir. Çalışmanın yararının deney grupları üzerinde ölçülmesi hedeflenmektedir.

Trafik ve sürücü simülasyonları ve araç simülatörleri; ekipman ve gerekse araç ve ortam modelleri, ışıklar, trafik yönetim sistemi varlığı gibi içeriği açısından gerçek yaşama oldukça yaklaşabilir. Hedefimiz konunun bir oyun olarak ele alınması ile yaygınca kullanılması olduğundan özel ekipmanlardan ve güçlü bilgisayar gereksinimlerinden uzak durulmuştur. Bunun işlem gücü gereksinimine bir yansıması olarak gerek trafik yönetimi, ışıklar ve gerekse sanal araç model çeşitliliği ve karmaşık yapılı trafik düzenlemelerinden de kaçınılmıştır. Bu bağlamda otonom araç modelinde bir araç simülasyon sisteminde görülebilecek düzeyde karmaşık yapılı sensörler ya da üst düzeyde bilgilendirme yer almamakta ve gerçek bir araca yüklendiğinde çalışacak düzeyde model tanımlamaları, araç denetim yazılımları ve senaryoların tamamı gibi hedefler de bulunmamaktadır. Bu indirgemenin diğer bir nedeni geliştirilen oyunun mobil aygıtlarda oynanabilirliğini ve dolayısı ile oynanma oranını arttırmaktır. Oyun geliştirme platformu olarak seçilen Unity3D oyun motoru, geliştirilen bir uygulamanın farklı mobil işletim sistemleri için sürümlerini üretme konusunda önemli bir destek vermektedir.

Türkiye'de trafik kazalarının %70 ve fazlasının kavşaklarda olduğu belirtilmektedir. Özellikle trafik ışıklarının olmadığı kavşaklardaki araç önceliklerinin sadece kurallarla yönetilir olması ve bu kuralların doğru uygulanmaması sonucunda yalın kavşaklardaki kaza artışına neden olmaktadır. Ayrıca kavşak dönüşlerinde bina, bitki, şehir mobilyası ya da diğer engelleyicilerin varlığı hızla girilen kavşaklarda önemli görüş sorunları kaynağı oluşturmaktadır. OKANOM oyunu geliştirilmesinin ilk aşamasında trafik ışıklarının yer almadığı kavşaklar konu edilmekte, ikinci aşamada ise yine trafik ışıkları yer almadan yaya geçitleri ve yayalar eklenmesi planlanmaktadır. Otonom ya da sürücü destekli araçlardaki üstün özellikler ve desteklerden, bu oyunda sürücüye verilmesi zor olmamasına rağmen, amaç sürücü becerisi geliştirmek olduğundan özellikle kaçınılmıştır. Oyunun oynanmasını sağlamak üzere, sürücüye sadece güzergahı yönlendiren basit yön okları ile bir navigasyon desteği verilmiştir. Bu navigasyonda harita üzerinde güzergah gösterme desteği yine aynı düşüncelerle verilmemiştir.

OKANOM oyununun birden fazla özelliklere göre düzenlemiş seviyeleri mevcuttur. Birinci seviye grubu oyunun amacına yönelik olarak araç sayısı ve araç sınıflarına (sürücülü ve sürücüsüz) ile yaratılır (Bölüm 6). Birinci seviye grubunun alt seviyeleri ise ikinci seviye olarak adlandırılır. Burada aşama aşama basit trafik düzeninden daha zor trafik düzenine geçilmesi oyun seviyelerini oluşturur. İkinci tür seviyeler, verilen görevlerin geçtiği güzergah üzerindeki kavşak türleri ve yerleşke türleri ile yaratılır. Yerleşke türünde kavşakların köşelerinde görüşü engelleyecek elemanların varlığı ile zorluk dereceleri oluşturulmaktadır. Güzergah üzerinde kavşak türlerinin varlık sayısı da bir zorluk derecesi oluşturmaktadır.

Bu çalışmanın konusu oyun bir öğrenci projesi olarak başlamış ve geliştirme süreci devam etmektedir. Bildirinin hazırlandığı tarihlerde geliştirmenin ilk evresi olarak araç modeli, otonom araç denetimi, sürücü denetimi, ortam gerçekleme ve temel kavşak senaryoları gibi konular tamamlanmış ve oyun seviyelendirme çalışmaları başlamıştır. OKANOM Oyun Projesi, Okan Üniversitesi'nin Bilgisayar Araştırma ve Uygulama Merkezi ile Ulaştırma Teknolojileri ve Akıllı Otomotiv Sistemleri Uygulama ve Araştırma Merkezi tarafından desteklenmektedir.

2. Simülasyon Yazılımları ve Oyun Motorları

Bu bölümde genel olarak 3 boyutlu simülasyon yazılımları ve oyun motorları hakkında kısaca karşılaştırmalı bilgi verilecek ve kullanılan oyun motoru Unity3D tanıtılacaktır. Sürüş ve trafik simülasyon sistemleri/simülatörleri ise bir sonraki bölümde konu edilmiştir.

3B simülasyon yazılımları ile 3B oyun motorlarının temel hedefleri ayrı olsa da kullanıcıya sunulan grafik ürünlerinin gerçek zamanda üretilmesi konusunda birleşirler. Buradaki gerçek zaman tanımı kullanıcının algılayabileceği zaman dilimleri içinde anlamlı ekran görüntülerinin üretilmesini kapsar. 3 boyutlu nesnelerinin bilgisayar ve kullanıcı denetimi uyarınca ekranda gösterilmesi alt düzey programlama (örneğin C dili kullanarak) ile erişilen donanım grafik alt sistemini yöneten kütüphaneler (OpenGL, DirectX) üzerinden yapılır. Grafik kütüphaneleri 3 boyutlu simülasyon/oyun nesnesini doğrudan desteklememekte sadece nokta, çizgi ve çokgen üzerinde çalışmaktadır[1]. Çeşitli firmalar belli alanlara yönelik karmaşık yapılı grafik uygulamalarına yönelik ortam ve nesnelerin üretilmesi ve denetlenmesini sağlayan simülasyon yazılım paketleri/oyun motorları üretmiştir. Bunları sadece grafik üretimi ile sınırlı kalmayıp çoklu ortam bileşenlerine de (ses, efekt, müzik, video vb.) destek vermesi beklenir.

3B simülasyon sistemlerinin bilgisayar oyunlarının aksine ele aldığı konuya sahici yaklaşması ve daha kesin hesaplamalar yapılması gerekir. Dolayısı ile bu sahiciliği sağlamak için önemli işlem gücüne gereksinim duyulur. Bilgisayar oyunlarının temel amacı ise yaratılmak istenen hayalin içine oyuncuyu çekebilmektir. Bunu gerçek zamanda sağlamak için gerçeklemede görsel kandırmacaları da kapsayan hileler kullanılır. Gelişim süreci göz önüne alındığında gerek donanımlarının pahalı olması ve gerekse özel olarak üretilmesi gereken yazılımların çokluğu simülasyon sistemleri etkin örneklerinin daha önce çıkmasına neden olmuştur. Bu ürünlerin talep sayısı az olmasına rağmen gerek duyan firmalar tarafında yüksek bedellerle satın alınması sonucunda konu üzerinde bilgi ve yazılım birikimi sağlamıştır. Kütüphanelerin ve yazılım paketlerinin, üzerinde çalışması gereken donanım sistemi maliyetleri nedeni ile yaygınlaşması başlangıçta mümkün olmamıştır. Bilgisayar oyunlarını talep eden kitlenin büyüklüğü ve ödemeye hazır oldukları bedeller sonuçta hem grafik donanım/bilgisayar maliyelerini düşürmüş ve hem de performansı yüksek oyun motorlarını ortaya çıkarmıştır. Günümüzde ise oyun motorların simülasyon sistemlerinin de önemli bir parçası olmaya başladığını gözlemlemekteyiz.

Yapılan araştırma sonucunda oyun motoru olarak Unity3D kullanılmaya karar verilmiştir. Unity3D'nin tercih edilmesinin başlıca sebepleri: betik dili ile programlanabilir olması, kendisinin çeşitli işletim sistemlerinde çalışmasını yanı sıra kendisi ile üretilen oyunların yine çeşitli işletim sistemlerinde çalıştırılır sürümlerinin üretilmesinin kolay olmasıdır. Geliştirme aşamasında yeterli olacak özelliklerde ücretsiz bir öğrenci/deneme sürümüne sahiptir. Temel özellikleri arasında; hızlı uygulama geliştirmeyi destekleyen kolaylıkları, arayüzleri sayesinde değişkenlerin değerlerinin rahatlıkla değiştirilmesi ile ayarlama kolaylığı, web sitesinde yer alan mağaza üzerinden birçok ücretli ve ücretsiz örnek program parçacıkları ve eklentilere, grafik nesnesi ya da 3B model kaynaklarına aynı yerden ulaşma kolaylığı sayılabilir [2]. Ağ desteği ile birden fazla kullanıcının oyunu kullanmasına izin verir. Ücretsiz eklentisi sayesinde hiçbir kurulum işlemi olmadan web tarayıcı üzerinden çalışabilir. Java Script, C# ve Boo gibi üç farklı betik dili ile programlama mümkündür. Geniş bir doküman kütüphanesi mevcuttur. Ayrıca Unity3D kullanan geliştirici sayısı fazla olduğu için bunların ürettikleri çok sayıdaki dokümanları ve eğitim videolarını internet üzerinden ücretsiz edinmek mümkündür.

3. Sürüş ve Trafik Simülasyon Sistemleri

Bir sürüş simülatörün önemli bölümleri 3B grafik yazılımı, konuya yönelik simülasyon, modelleme ve hesaplama yazılımları ve mekanik teçhizattır. Bu bölümde bazı sürüş ve trafik simülasyon sistemleri tanıtılmıştır. Proje konusunu yakın olarak içerdikleri için bu örnekler seçilmiştir. Çoğu kez ürün olarak satılan bir simülasyon yazılım paketi yeni bir tasarımda tümüyle yeterli olmamakta; örneğin araç dinamikleri modelleme ve gerçekleme gibi konularda diğer yazılımlar/ paketleri kullanarak yeni üretimler yapılması ve bu üretimlerin simülasyon sistemine eklemlendirilmesi gerekmektedir. Giriş bölümünde belirtildiği gibi çalışmada sunulan projenin amacı ürün gerçekleme maliyetleri ve bu ürünün çalıştırıldığı bilgisayar gereklilikleri konusunda farklılık yaratmaktadır. Proje amacının bu bölümde tanıtılan sistemler üzerinde geliştirilmesinin hem maliyeti yüksek olacaktır ve hem de uygulamanın çalıştırılacağı bilgisayarın isterleri ve lisans maliyetleri ile yaygın kullanımının da sağlanması kolay olmayacaktır.

3.1. SUMO

Alman Havacılık ve Uzay Merkezi Ulaşım Sistemleri Enstitüsü çalışanları tarafından, geniş yol ağlarını işlemek için tasarlanmış bir simülatör paketidir. GPL altında lisanslı ve açık kaynak kodlu bir yazılımdır [3][4].

3.2. Carnetsoft
Hollanda'da geliştirilmiş simülasyon sistemidir. Sürücülerin trafiğe çıkmadan önce oluşturulan sanal trafik ortamı içerisinde eğitilmelerini amaçlar. Masaüstü simülatör ve kokpit simülatör türleri vardır [5].

3.3. City Car Driving
Rusya'da kullanılan üç boyutlu eğitim simülatörüdür. Amacı kullanıcılara trafik kurallarını benimsetmek ve trafikte sürüş pratiği kazandırmaktır [6].

3.4. Trafic Talent
Trafic Talent sürüş simülasyonunda kullanıcıya verilen farklı görevlerin tamamlanması istenir. Bu görevler genel olarak verilen hedeflere kaza yapmadan, verilen süre zarfında ve trafik kurallarına uyarak varmak şeklindedir [7].

3.5. Ohio State Üniversitesi Sürücü Simülatörü
Sistem denetimi ve sürücü senaryoları *oluşturmada Realtime Technologies Inc. (RTI)* firmasının *SimCreator* ve *SimVista* yazılım sistemleri kullanılmıştır. Çoklu gövde dinamik yazılımı ile hareket denetimi geliştirilmiştir. Simülatör; altı eklemli platform üzerine yerleşmiş bir araç kaportası, denetimleri kapsayan mekanik teçhizat ve 260° görüşlü silindirik bir ekran ile donatılmıştır [8][9].

3.6. Yonsei Üniversitesi Simülatörü
Yonsei üniversitesi tarafından geliştirilmiş otonom araca ait testleri öncelikle bilgisayar ortamında gerçeklemek için oluşturulan bir simülatördür. GPL lisansı ile OpenCarSimManager kullanan açık kaynaklı bir projedir[10].

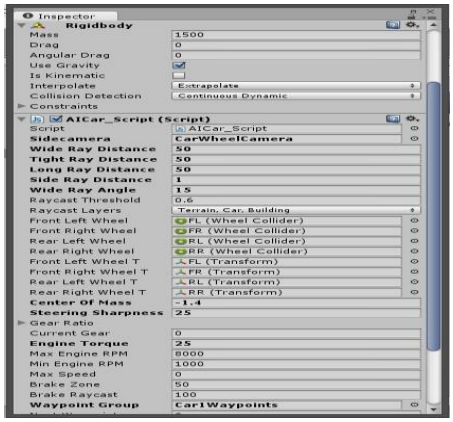
4.Oyunun Gerçeklenmesi

Bu bölümde oyunun önemli gerçeklenme bölümleri olarak 3 temel bölüm ele alınabilir. Araç modeli alt bölümünde; araç katı cismi oluşturma, araç modelinin dinamik ve sensör özellikleri ile bazı önemli parametreler yer alır. Yerleşke modeli alt bölümünde ise; yol ve kavşakların oluşturulması ve sürücüsüz araçların belirlenen güzergah üzerinde gitmesi açıklanır. Son olarak kullanıcı arayüzü alt bölümünde, sürücülü araç için kullanıcıya sunulan denetim ve yönlendirme tanıtılır. Senaryolar ise sonraki bölümün konusudur.

4.1. Araç modeli

Araç modeli geliştirme aşamasında yapılmış Smart AI Car gibi örnekler incelenmiş, açık kaynak özellikleri olanlardan yararlanılmış ve uygulamanın gerektirdiği yeterlilikte bir araç modeli geliştirilmiştir [11]. Bu araç modeli, kapının oyuncu tarafından açılabilmesi gibi bir arabanın tüm mekanik özelliklerinin karşılayan bir model değildir. Unity3D de bir nesne modeli 3 bölümden oluşmaktadır. Bu bölümler; katı cisim modeli, dönüşüm modeli, çarpışma modelidir. Bir karmaşık yapılı oyun nesnesi oluşturmak için parçalardan bütüne doğru bir tasarım aşaması uygulanır. Bu yöntemde, oyun nesnesinin tüm bileşenleri teker teker oluşturulur daha sonra bu bileşenler birbiri ile ilişkilendirilir. Örnek olarak arabanın bir alt bileşeni olarak gerçeklenen tekerleklerin yine aynı yöntemle kendisinin oluşturulmasını gösterirsek; bir tekerleğin öncelikle iç bileşenleri yani mekanik kısmı katı cisim olarak gerçeklenir, daha sonra dış bileşeni yani lastiğin kaplanması aşaması gerçeklenir (Şekil 1). Oluşturulan bu tekerlek için tasarlanan dönüşüm modeli (tekerleğin dönerek yol alması) ile yukarıda tanıtılan katı cisim modeli birleştirilerek bir oyun nesnesi oluşturulmaktadır. Bu nesne diğer bir nesne ile çarpıştığında davranışı programlanarak işlem tamamlanır.

Şekil 1*:* İç ve dış bileşenler



Şekil 2*:* Araç modelinin nesne özellikleri

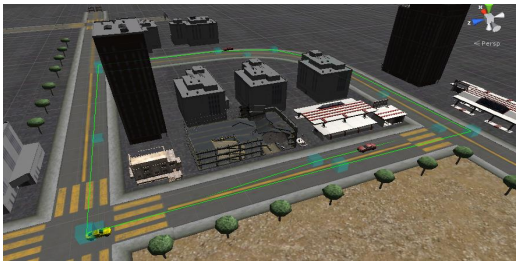
Oyun nesnesi oluşturulduktan sonra nesnenin katı cisim modeli ile özelliklerinin atanması yapılır. Araç nesnesi için katı cisim modelinde fiziksel özellikler, arabanın kütlesi, sürtünme katsayısı, yerçekimi, hızı gibi atamalar gerçekleştirilmiştir (Şekil 2).
Otonom araçlarda kullanılan sensörlerden biri olan LIDAR (Laser Imaging Detection and Ranging) sensörü sayesinde tanımlanmış bölge içinde var olan diğer nesnelerin varlığı ve bunların araçtan uzaklığı

belirlenir [12][13]. Projede ise bu yöntemin daha az ışın kullanan benzeri olarak ışın kesme nesneleri (sensörleri) ile bir engel sezme yeteneği oluşturuldu. Projede aracın yanlarında ve önünde olmak üzere ışın kesme sensörleri yaratıldı. Böylelikle sürücüsüz araç belirlenen mesafede olan nesneleri algılayabilmektedir. Sürücüsüz araç geliştirilen senaryolar yardımı ile örneğin önünde bir engel var ise hızını yavaşlatıp manevralar yaparak engeli geçebilir ya da tüm hareket alanını engelliyor ise engel önünden kalkana kadar durabilir.

Defense Advanced Research Projects Agency (DARPA) tarafından oluşturulan Route Network Definition File (RNDF) ve Mission Data File (MDF) benzeri bir yapı kullanılarak hedef ve bu hedefe ulaşılabilmesi için gidilebilecek rota bilgilerinin otonom araca aktarılması sağlandı[14].

4.2. Yerleşke modeli

Sahneye yerleştirilen düzlem proje için tasarlanan yol modeli eklendi. Buradaki en büyük sorun sürücüsüz aracın yolu tanımasını sağlanması ile yol çizgilerini takip edip hareketi boyunca yolun dışına çıkmamasıdır. Kullanılan basit yöntemlerde araç hareketini güzergah üzerine (ayrılan şeridin ortasına) konulan işaret noktalarına (waypoint) göre gerçekleştirir. Yerleştirilen işaret noktaları her ne kadar yolun üzerinde dahi olsa araç virajlardan dönerken aldığı momentle hareketini gerçekleştirirken ya da bir engelle karşılaştığında o engeli geçerken yolun dışına çıkabilir. Genelde, binalar ve diğer görünür engeller yoldan çıkılmaması için kullanılır. Bunların olmadığı durumlarda ise sorunu çözmek için yol kenarlarına sadece sürücüsüz araçlarının etkilendiği görünmez engeller eklendi (Şekil 3). Araç, ışın kesme sensörü sayesinde yol kenarında ki engeli algılayarak yolun dışına çıkmamasını sağlayacak hız denetimlerini dinamik olarak yapar.



Şekil 3*:* Güzergah işaretleme ve yol kenarı engelleri

4.3. Kullanıcı arayüzü

Kullanıcıya basit bir arayüz sunulmaktadır. Farklı ortamlarda yaygın kullanılması düşünüldüğünden oyun denetimlerinin basit tutulması istenmiştir. Ayrıca mobil ortamlar düşünüldüğünde tek ekranın kullanılması söz konusudur. Bu nedenle ekranın araç göstergeleri ile donatılmasına yerine mümkün olduğunda yol görüntüsüne ayrılması planlanmıştır (Şekil 4). Normal bir sürücünün sadece yola bakması durumu yaratılmıştır. Sürücüye hedeflerin atanması ve sürücünün güzergahta yolunu bulması için basit bir navigasyon olarak kavşak öncesi yön okları çıkmaktadır.
Oyunda kullanılan tuş kombinasyonları; “Q” ve “E” tuşları sürücülü aracın sinyal ışıklarını yakıp söndürme, “Space” tuşu sürücülü araç için el freni, “A” ve “Z” otonom araçların hızının arttırılıp azaltılması, “yukarı yön” tuşu sürücülü araç için gaz, “aşağı yön” tuşu ise sürücülü aracın hızının azaltılması ve geri vites, “sağ” ve “sol” yön tuşları aracın yönünü değiştirme, “C” tuşu otonom araçların kameralar arası geçişi ve “Esc” tuşu ise oyunun bitirilmesini sağlayan tuş atamalarıdır. Mobil aygıtlar için tıklanabilir sanal tuşlar oluşturulacaktır.

Şekil 4*:* Tasarım aşamasında sahnenin genel görünümü

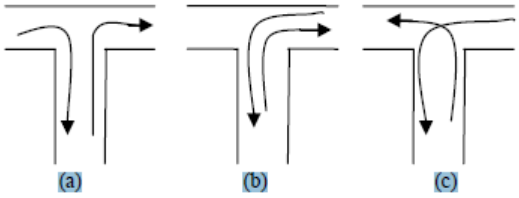
5. Senaryolar

Otonom araç sürme 2 aşamada ele alınabilir. İlki temel sürüşün sağlanması diğeri ise normal olanın dışındaki bir durumda aracın nasıl davranacağının belirlenmesi. Çoğu kez temel bir eylem senaryosu aykırı durumlarının kapsanması ile karmaşık yapıya bürünmektedir [15]. Proje kavşak tabanlı olduğundan bu bölümde örnek bir kavşak senaryosu problemleri tanıtılacaktır. Genel olarak sürüş senaryoları arasında şunlar sayılabilir: trafik engelsiz yol sürüşü, haraketli aracı geçme, duran bir engeli ya da aracı geçme, kavşak geçişleri, trafiği yoğun yola girme/yoldan çıkma, trafik nedeni ile yolda duraksama ve devam etme. Bu senaryolarda örneğin bir kavşakta 2 sürücüsüz aracın geçişlerinin başarılmasının sağlanmasının yanı sıra sürücüsüz aracın araçlardan birinin sürücülü olduğu durumda da kurallarının üretilmesi gerekir. Bu durumda sürücünün kurallara uygun davrandığında hata yapılmamasının ve kurallara uygun davranmadığında ise mümkün olduğunca sürücüsüz aracın kaza olmasını engelleyecek uygun

davranışlarda bulunmasının sağlanması söz konusudur.
Kavşaklarda yolun şerit sayısına göre senaryolarda çeşitlemeler olacaktır. Kavşak senaryoları bir sonlu durum makinesi olarak modellenir ve gerçeklenir [15].

5.1. T kavşak senaryoları

T kavşak senaryosunun türevleri Şekil 5'de gösterilmektedir. Birinci senaryo adımında (Şekil-5a) iki aracın bir T kavşakta karşılaşıp farklı yönlere doğru hareketleri söz konusudur. Buradaki hedef araçların sorunsuz şekilde kendi mevcut şeritlerini takip ederek birbirlerini çarpmadan ilgili yönlere yönelmeleridir. Bu senaryoda temel sorun geçiş sırasında araçlar birbirine çarpmamasına rağmen tam köşede geçiş sırasında çok yaklaşmaları nedeni birbirilerini engellemeleridir. Araçlardan biri sürücülü ise sürücü bu durumdan kendini kurtarabilir. Ancak 2 otonom araç birbirini kilitleyecektir. Senaryoda iki aracın karşılıklı durumunu bilmesi şüphesiz kurtulmayı kolaylaştıracaktır. Ancak senaryo gerçeklemede bu tür bilgilerden kaçınmaya özen göstereceğiz.



Şekil 5*:* T kavşakta geçiş türleri

Şekil-5b'de gösterilen senaryoda, T kavşakta iki aracın da aynı dönüş noktasını kullanarak diğerinin geldiği yöne doğru hareketinin sağlanmasıdır. Buradaki sorun önceki örnekteki karşılıklı engelleme sorunun tüm geçiş boyunca yer almasıdır. Kavşak geçişlerinde uygun ışın kesme açıları kullanılarak sorunun giderilmesine çalışılır. Şekil-5c'de ise araçların geçiş şeritleri kesişmektedir. Yola önce girenin geçiş üstünlüğünü alarak kaza ve karşılıklı engellenme olmaması sağlanmalıdır.

6. Oyunun Genel Yapısı

Gerçeklenen çalışmanın birincil hedefi bir ciddi oyun olarak eğitimi kapsamasıdır. Bu amaca ne kadar yaklaşıldığını ölçmek için deney gruplarına aşağıdaki düzende oyun seviyeleri uygulanması planlanmıştır. Oyuncuların farklı başarım ölçütlerine ilişkin topladığı puanlar sonraki bir araştırmanın verisini oluşturacaktır. Oyunlarda; oyuncunun oynama süresini arttırma amacına yönelik ilgisinin sürekli kılınması için genelde basitten karmaşık yapılara göre seviye tasarımları yapılır. Seviye içinde oyun mekanikleri ile bir oynama düzeni kurulur. Çalışmamızda ise oyuncunun edinebileceği tecrübenin geliştirilmesi ve bu gelişmenin ölçülmesi de hedeflendiğinden oyun mekanikleri buna göre tasarlanmıştır.

Tasarlanan oyun seviyeleri şu şekilde sıralanabilir: Araç denetime alıştırma turları, boş ve bir veya iki sürücüsüz araç bulunan yollarda serbest gezinme, sadece sürücülü araç trafiğinde verilen güzergahları tamamlama, sadece sürücüsüz araç trafiğinde bir sürücüsüz aracı takip ederek verilen güzergahları tamamlama, sadece sürücüsüz araç trafiğinde verilen güzergahları tamamlama, sürücü ve sürücüsüz araç trafiğinde verilen güzergahları tamamlama ve son olarak 3. adımda yapılan sürücülü araç trafiğinde verilen güzergahları tamamlama. 3. Seviye ile son seviye gelişimi karşılaştırma amacı için kullanılır. Burada tanımlanan seviyelerin kendi içinde az kavşaklı, zor kavşaklı gibi önceki bölümlerde anlatılan alt seviyeleri olacaktır. Çok sürücülü ortamlarda diğer sürücülerin çalışmasının aksatılmaması ya da amaç dışı davranışları engellemek için belli sayıda ya da türde hata yapan sürücünün denetimi elinden alınır ve aracı sürücüsüz bir araç olarak başlangıç yerleşkesine gönderilir ve oyuncu bu sürüşü izler ve o seviyeye yeniden başlar.

7. Sonuçlar

Bu çalışmada araç ve trafik simülasyonunun bir ciddi oyun olarak uygulanmasındaki aşamalar irdelenmiştir. Bir oyun hedefi belirlenerek ve bu hedef doğrultusunda simülasyon gereksinimleri azaltılarak seçilen bir oyun motoru ile gerçeklemenin ilk aşamaları başarılmıştır. Bundan sonraki genişletme olarak sırası ile yayaların, trafik levhalarının, trafik ışıklarının ve bir trafik yönetim sisteminin sisteme eklenmesi gözetilebilir. Bu genişletmelerinin oyun hedefi üzerinde ve sistem gereksinimlerinin artması nedeni ile oynanma yaygınlığı üzerinde olumlu olumsuz etkileri olacaktır. Ele alınan problemin, bu çalışmadaki sunulan seviyesinde dahi, istenen detayda ve işlev kalitesinde gerçeklenmesi önemli güçlükleri barındırmaktadır. Çalışmada simülasyon karşılığı isterler karşılanmış ve gerçeklenmiştir. Simülasyon yönü ile ele aldığımızda özellikle senaryo tabanında çalışmaların genişletilmesi sağlanacaktır. Çalışmanın deney grupları üzerinde denenmesi ve sonuçların irdelenmesi hedeflenmektedir. Diğer bir aşama olarak, çalışmanın bir oyun olarak kabul edilmesini sağlayacak yeteneklerini ve yaygınlığını arttıracak özelliklerle genişletmek hedeflerin arasında vardır.

8. Kaynaklar

[1] Bayarri, S., Fernandez, M., Perez, M. "Virtual reality for
driving simulation." ACM 39, 1996, 72-76

[2] http://unity3d.com/ (Ziyaret tarihi Temmuz 2014)

[3] Pursula, M. "Simulation of traffic systems-an overview."
Journal of Geographic Information and Decision Analysis,1999, 1-8.

[4] http://sumo-sim.org/ (Ziyaret tarihi Temmuz 2014)

[5] http://www.carnetsoft.com/(Ziyaret tarihi Temmuz2014)

[6] http://citycardriving.com/ (Ziyaret tarihi Temmuz 2014)

[7] http://www.unity3dgames.eu/traffic-talent.html (Ziyaret tarihi Temmuz 2014)

[8] http://drivesim.osu.edu/about/ (Ziyaret tarihi Temmuz 2014)
[9] http://citr.osu.edu/CrIS/?page_id=362 (Ziyaret tarihi Temmuz 2014)

[10] Lee,S., Cho, J., and Young Ju, D. "Autonomous Vehicle
Simulation Project." International Journal of Software
Engineering and Its Applications Vol.7, 2013, 393-402

[11]http://u3d.as/content/bone-cracker-games/smart-ai-car-2- 1/6gu (Ziyaret tarihi Temmuz 2014)

[12] http://www.sick.com/ (Ziyaret tarihi Temmuz 2014)

[13] Akgün, T., Koç, Z., Güner, Ş., Öztürk, B., Özkan, B., Üstün, Ö., Tuncay, N., Özgüner, Ü., "A Study on Autonomous Vehicle Development Process at Okan University", 2012 IEEE International Conference on Vehicular Electronics and Safety, 2012, 369-374

[14] DARPA Urban Challenge Route Network Definition
File and Mission Data File Formats, 2007,1-14

[15] Özgüner, Ü., Acarman, T., Redmill, K., Autonomous Ground Vehicles, Artech House, USA, 2011

EK-B

Sürücü Davranışı İyileştirmeye Yönelik Bir Oyun ve Yol Tanımlama

Kader Nikbay[1], Kutay Ata Şen[1], B.Tevfik Akgün[1]
[1] Okan Üniversitesi, Bilgisayar Mühendisliği Bölümü, İstanbul
kader.nikbay@okan.edu.tr,kutayatasen@gmail.com,tevfik.akgun@okan.edu.tr

Özet: Bu çalışmada, bir 3 boyutlu oyun (OKANOM') oluşturularak sürücüsüz araçların varlığı ile zenginleştirilmiş bir sanal trafikte sürüş deneyimleri ile sürücü davranışlarının ölçülmesi, iyileştirilmesi hedeflenmiştir. Oyunun ikinci evre geliştirilmesinde ABD Savunma Bakanlığı İleri Araştırma Projeleri Ajansı (DARPA) tarafından üretilen yol ağ tanımı (RNDF) ile hedef (MDF) veri yapısı uyarlanacaktır. Bildiride; birinci bölüm olarak çalışmanın gerçeklenmesi, ve oyun seviyeleri özetlenmiştir. İkinci bölümde ise DARPA yol ağ tanımları ve hedef veri yapıları özetlenmiştir.

Anahtar Sözcükler: 3B Oyun, Araç Sürüş Simülasyonu, Sürücüsüz Araç, RNDF, MDF.

A Game For Improvement Of Drivers' Attitudes And Methods Of Route Recognition

Abstract: In this paper, the main goal is scaling the drivers' attitudes and driving experiences in a virtual traffic by inclusion of autonomous vehicles and improvement of the drivers' attitudes by constructing a three dimensional game (OKANOM).In the second stage of the development of the game , the mission data file (MDF) by the route network definition file (RNDF) which was generated by DARPA(TheDefense Advanced ResearchProjectsAgency) will be adapted. In the article,the first part is about the implementation of the study and game levels and in the second DARPA route network definition file and the mission data file are explained.

Keywords: 3B Games, Simulation of Vehicle Driving, Autonomous Vehicle, RNDF, MDF

1. Giriş

Bu çalışmada 3 boyutlu bir oyun kullanarak; sürücülü ve sürücüsüz (otonom) araçların birlikte bulunduğu bir sanal trafik ortamında sürücülerin (oyuncuların) verilen hedefleri gerçeklemesi sırasındaki sürücü davranışlarının incelenmesi ve iyileştirilmesi hedeflenmektedir. OKANOM olarak adlandırılan oyun çok oyunculu bir yapıda tasarlanmıştır. OKANOM ortamında oyuncular kendilerine atanan hedeflere (yerleşkelere) ulaşırken yaratılan bir trafikte yer almaktadır. Bu trafikte diğer oyuncuların denetlediği araçların yanı sıra sürücüsüz araçlar da yer almaktadır. Sürücüsüz araçlar kurallara uygun hareket etmektedirler. Sürücüsüz araçların varlığının ve sayısının mevcut trafiği ne derece de etkilediği ve sürücülerin sürücüsüz araçlara ve diğer sürücülü araçlara olan davranışlarının nasıl değiştiği araştırılacaktır. Beklenen ya da

istenen sonuç; trafikte kurallara uygun seyreden sürücüsüz araçların sayısının belli bir oranda artmasının trafiğin daha iyi akmasına ve kullanıcı davranışlarının gelişmesine destek vermesidir. Ancak sürücülerin sürücüsüz araç davranışlarını öğrendikçe bunları kendi lehlerine istismar etmeleri de mümkündür.

Çalışmanın eğitim ve araştırma amaçlı bir ciddi oyun olarak tasarlanmasının temel amaçları: oyun biçiminde sunulan proje hedeflerinin kullanıcılar tarafından benimsemesini kolaylaştırması ve gerçeklemeyi destekleyen oyun motorlarının varlığıdır. Çalışmanın bir oyun olarak tasarlanmasının olumsuz yönü oyuncuların düzgün araç kullanma konusunda özenli olmayabilecekleridir. Oyunun ödüllendirilerek özendirilmesi ya da kötü puanlanmanın oyuncuya olumsuz geri dönüşü bu oyunun belli bir sorumlulukla oynanmasını sağlayabilir. Çalışmanın yararının deney grupları üzerinde ölçülmesi hedeflenmektedir.

Trafik ve sürücü simülasyonları ve araç simülatörleri; ekipman ve gerekse araç ve ortam modelleri, ışıklar, trafik yönetim sistemi varlığı gibi içeriği açısından gerçek yaşama oldukça yaklaşabilir. Hedefimiz konunun bir oyun olarak ele alınması ile yaygınca kullanılması olduğundan özel ekipmanlardan ve güçlü bilgisayar gereksinimlerinden uzak durulmuştur. Bu indirgemenin diğer bir nedeni geliştirilen oyunun mobil aygıtlarda oynanabilirliğini ve dolayısı ile oynanma oranını arttırmaktır.

OKANOM oyunu geliştirilmesinin ilk aşamasında trafik ışıklarının yer almadığı kavşaklar konu edilmekte, ikinci aşamada ise yine trafik ışıkları yer almadan yaya geçitleri ve yayalar eklenmesi planlanmaktadır. Otonom ya da sürücü destekli araçlardaki üstün özellikler ve desteklerden, bu oyunda sürücüye verilmesi zor olmamasına rağmen, amaç sürücü becerisi geliştirmek olduğundan özellikle kaçınılmıştır.

OKANOM oyununun birden fazla özelliklere göre düzenlemiş seviyeleri mevcuttur. Bu seviyeler; Araç denetime alıştırma turları, boş ve bir veya iki sürücüsüz araç bulunan yollarda serbest gezinme, sadece sürücülü araç trafiğinde verilen güzergahları tamamlama, sadece sürücüsüz araç trafiğinde bir sürücüsüz aracı takip ederek verilen güzergahları tamamlama, sadece sürücüsüz araç trafiğinde verilen güzergahları tamamlama, sürücü ve sürücüsüz araç trafiğinde verilen güzergahları tamamlama.

OKANOM'daki sürücüsüz araçlar için yol tanımlama ve hedef güzergahı tanımlama problemlerinin çözümü için DARPA tarafından geliştirilen RNDF ve MDF yapılarının kullanılması hedeflenmiştir.

OKANOM Oyun Projesi, Okan Üniversitesi'nin Bilgisayar Araştırma ve Uygulama Merkezi ile Ulaştırma Teknolojileri ve Akıllı Otomotiv Sistemleri Uygulama ve Araştırma Merkezi tarafından desteklenmektedir.

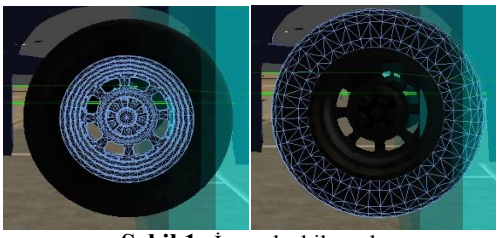
2. Oyunun Gerçeklenmesi

Bu bölümde oyunun önemli gerçeklenme bölümleri olarak 3 temel bölüm ele alınabilir. Bu bölümler; araç modeli , yerleşke modeli ve arayüz 'dür

2.1. Araç modeli

Araç modeli geliştirme aşamasında yapılmış Smart AI Car gibi örnekler incelenmiş, açık kaynak özellikleri olanlardan yararlanılmış ve uygulamanın gerektirdiği yeterlilikte bir araç modeli geliştirilmiştir [1]. Unity3D de bir nesne modeli 3 bölümden oluşmaktadır. Bu bölümler; katı cisim modeli, dönüşüm modeli, çarpışma modelidir. Bir karmaşık yapılı oyun nesnesi oluşturmak için parçalardan bütüne doğru bir tasarım aşaması uygulanır[2]. Bu yöntemde, oyun nesnesinin tüm bileşenleri teker teker oluşturulur daha sonra bu bileşenler birbiri ile ilişkilendirilir. Örnek olarak arabanın bir alt bileşeni olarak gerçeklenen tekerleklerin yine aynı yöntemle kendisinin oluşturulmasını gösterirsek; bir tekerleğin öncelikle iç bileşenleri yani mekanik kısmı katı cisim olarak gerçeklenir, daha sonra dış bileşeni yani lastiğin kaplanması aşaması gerçeklenir (Şekil 1). Oluşturulan bu tekerlek için tasarlanan dönüşüm modeli (tekerleğin dönerek yol alması) ile yukarıda tanıtılan katı cisim modeli birleştirilerek bir oyun nesnesi oluşturulmaktadır. Bu nesne diğer bir nesne ile

çarpıştığında davranışı programlanarak işlem tamamlanır.



Şekil 1: İç ve dış bileşenler

Otonom araçlarda kullanılan sensörlerden biri olan LIDAR (Laser Imaging Detection and Ranging) sensörü sayesinde tanımlanmış bölge içinde var olan diğer nesnelerin varlığı ve bunların araçtan uzaklığı belirlenir [3][4]. Projede ise bu yöntemin daha az ışın kullanan benzeri olarak ışın kesme nesneleri (sensörleri) ile bir engel sezme yeteneği oluşturuldu. Projede aracın yanlarında ve önünde olmak üzere ışın kesme sensörleri yaratıldı. Böylelikle sürücüsüz araç belirlenen mesafede olan nesneleri algılayabilmektedir. Sürücüsüz araç geliştirilen senaryolar yardımı ile örneğin önünde bir engel var ise hızını yavaşlatıp manevralar yaparak engeli geçebilir ya da tüm hareket alanını engelliyor ise engel önünden kalkana kadar durabilir.

Araçların yol tanımları ise Defense Advanced Research Projects Agency (DARPA) tarafından oluşturulan Route Network Definition File (RNDF) ve Mission Data File (MDF) yapısı kullanılarak yapıldı. Bu konu ile ilgili detaylı açıklama Bölüm 3'de ele alınmıştır[5][6].

2.2 Yerleşke modeli

Sahneye yerleştirilen düzlem proje için tasarlanan yol modeli eklendi. Buradaki en büyük sorun sürücüsüz aracın yolu tanımasını sağlanması ile yol çizgilerini takip edip hareketi boyunca yolun dışına çıkmamasıdır. Kullanılan basit yöntemlerde araç hareketini güzergah üzerine (ayrılan şeridin ortasına) konulan işaret noktalarına göre gerçekleştirir. (Şekil 2). Araç, ışın kesme sensörü sayesinde yol kenarında ki engeli algılayarak yolun dışına çıkmamasını sağlayacak hız denetimlerini dinamik olarak yapar.

Şekil 2: Güzergah ve yol kenarı engelleri

2.3 Kullanıcı arayüzü

Kullanıcıya basit bir arayüz sunulmaktadır. Farklı ortamlarda yaygın kullanılması düşünüldüğünden oyun denetimlerinin basit tutulması istenmiştir. Ayrıca mobil ortamlar düşünüldüğünde tek ekranın kullanılması söz konusudur. Bu nedenle ekranın araç göstergeleri ile donatılmasına yerine mümkün olduğunda yol görüntüsüne ayrılması planlanmıştır (Şekil 3). Normal bir sürücünün sadece yola bakması durumu yaratılmıştır. Sürücüye hedeflerin atanması ve sürücünün güzergahta yolunu bulması için basit bir navigasyon olarak kavşak öncesi yön okları çıkmaktadır.

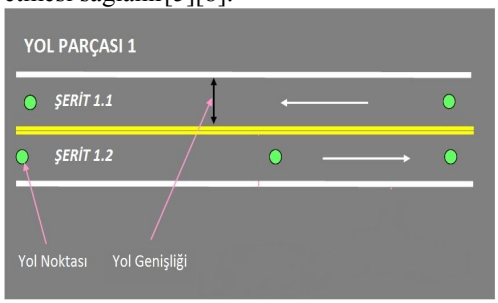
Şekil 3: Tasarım aşamasında sahnenin genel görünümü

3. DARPA yol ağ tanımları ve hedef veri yapıları

OKANOM'da öncelikle yol ağı tanımlama dosyasındaki (RNDF) veriler yol bulma algoritmaları ile yorumlanarak bir çizgeye dönüştürülmektedir.
Ardından sürücüsüz araca yol tanımlama dosyasındaki (MDF) veriler aktarılarak, dosyada belirtilen güzergah üzerindeki kontrol noktalarını ziyaret ederek, verilen hedefe ulaşması beklenir[7][9].

3.1 RNDF

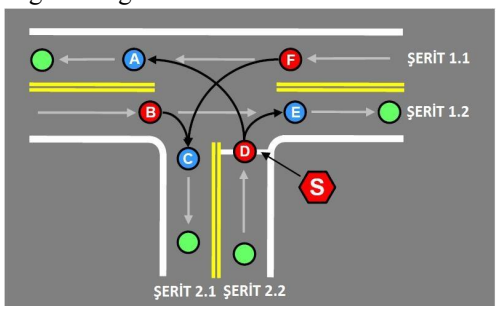
Yol Parçaları (Road Segment): Bir yol ağı bir veya birden çok şeridin bir araya gelmesi ile oluşur. Sözü edilen bu yol parçaları tek şeritli ya da çok şeritli olabilir. Bir şerit içinde şeridin genişliği, şerit çizgileri ve şerit ile ilişkili yol noktaları yer alır. Bu yol noktaları genellikle şeridin merkezine yerleştirilir. Ardışıl olarak yol noktaları takip edilerek aracın seyahat etmesi sağlanır[5][8].



Şekil 4: Yol Parçası ve Şeritler

Şekil 4 de bir yol parçası ve üzerinde sahip olduğu diğer bileşenler görülmektedir.
Şerit genişlik bilgisi zorunlu bir bilgi değildir bu sebeple her yol parçası içinde var olma zorunluluğu yoktur.

RNDF dosyası içinde yol noktalarına giriş ve çıkış bilgileri atanarak yol parçaları arasında bağlantı sağlanır.



Şekil 5: Yol Noktaları

Şekil 5'de mavi ile gösterilen yol noktaları girişleri, kırmızı ile gösterilen yol noktaları çıkışları göstermektedir. Örneğin D ile gösterilen çıkış noktası için A ve E diğer şeritlere giriş noktalarını oluşturmaktadır. Örnekte de görüldüğü gibi bir çıkış noktası birden fazla giriş noktası ile ilişkilendirilebilir. Ayrıca D noktasına "S" ile gösterilen bir "Dur" bilgisi de eklenmiştir. Bunun nedeni kavşakların kesişim noktalarında trafik durum bilgisinin kontrol edilme zorunluluğudur.

Şeritler üzerinde var olan yol noktalarından bir tanesi kontrol noktası olarak belirlenir ve bu noktalar araçlar tarafından güzergahı takip etmek için kullanılır[5].

Park Alanları : RNDF dosyası içerisinde yol parça bilgilerine ek olarak park alan bilgileri de mevcuttur. Park alan bilgilerinde giriş, çıkış noktaları, yol noktaları ve park alanın çevre nokta bilgileri yer almaktadır. Yol parçaları ve park alanlarının bağlantıları iki şekilde gerçekleşmektedir;

1.Durum: Yol parçası çıkış noktası ile park alanı giriş noktası ilişkilendirilir.
2.Durum: Park alanı çıkış noktası ile yol parçası giriş noktası ilişkilendirilir.

Şekil 6: Park Alanı

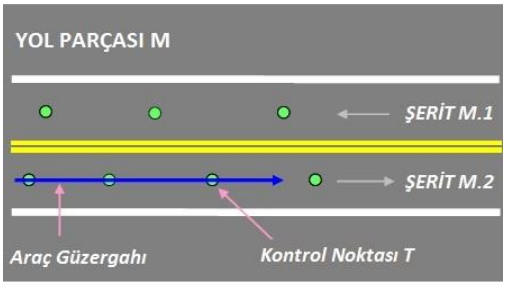
Park bölgesindeki özel olarak belirtilmiş araç park yerlerinde 2 adet yol noktası bulunur ve bu noktalardan biri aynı zamanda kontrol nokta bilgisini de taşır. Şekil 6'da bir park alanındaki belirtilen durumlar görülmektedir[5].

3.2 MDF

Aracın verilen göreve ulaşması için, ziyaret etmesi gereken kontrol nokta bilgilerinin tutulduğu dosyadır. Her MDF dosyası belirli bir RNDF dosyası tarafından yorumlanır. Birçok farklı MDF dosyası aynı RNDF dosyası ile ilişkili olabilir[5][8].

Kontrol Noktaları: Aracın verilen hedefe ulaşması için izlenecek rota üzerinde ziyaret edilmesi gereken noktalardır. RNDF dosyasında tanımlanmış olan kontrol noktaları, araç üzerinden geçtiği takdirde ziyaret edilmiş sayılmaktadır ve böylece belirlenen kontrol noktaları ziyaret edildikçe aracın doğru rotada olduğu tespit edilir. Şekil 7'de bir aracın

ziyaret etmiş olduğu kontrol noktaları görülmektedir.



Şekil 7: Araç Güzergahı

RNDF içinde tanımlanmış olan tüm kontrol noktaları MDF içerisinde yer almak zorunda değildir. Her hedef için aracın ziyaret etmesi gereken belirli kontrol noktaları vardır. Hedefe ulaşmayı sağlayan son kontrol noktası bitiş çizgisi görevini üstlenmektedir.

Hız Limitleri: MDF dosyasında aracın hedefe ulaşması sürecinde ziyaret edeceği her yol parçası için tanımlanmış olan minimum ve maksimum hız değerleri mevcuttur. Aracın mevcut olan hızını koruması için minimum ve maksimum değerlere aynı hız değerinin atanmış olması gerekir.

4. Sonuçlar

Bu çalışmada, bir oyun hedefi belirlenerek ve bu hedef doğrultusunda simülasyon gereksinimleri azaltılarak seçilen bir oyun motoru ile gerçeklemenin ilk aşamaları başarılmıştır. Bundan sonraki genişletme olarak sırası ile yayaların, trafik levhalarının, trafik ışıklarının ve bir trafik yönetim sisteminin sisteme eklenmesi gözetilebilir. Bu genişletmelerinin oyun hedefi üzerinde ve sistem gereksinimlerinin artması nedeni ile oynanma yaygınlığı üzerinde olumlu olumsuz etkileri olacaktır. Ele alınan problemin, bu çalışmadaki sunulan seviyesinde dahi, istenen detayda ve işlev kalitesinde gerçeklenmesi önemli güçlükleri barındırmaktadır.

Gerçeklenen diğer aşama ise RNDF ve MDF dosyalarındaki verilerin sürücüsüz araçlara aktarılmasıdır. Bu bölüm için bundan sonraki genişletme hedefi sürücüsüz araçların, yol tanıma ve hedef bulma çalışmalarının yürütülmesidir.
Ayrıca çalışmanın deney grupları üzerinde denenmesi ve sonuçların irdelenmesi hedeflenmektedir.

5. Kaynaklar

[1]http://u3d.as/content/bone-cracker-games/smart-ai-car-2-1(Ziyaret tarihi Aralık 2014)

[2] http://unity3d.com/ (Ziyaret tarihi Aralık 2014)

[3] http://www.sick.com/ (Ziyaret tarihi Aralık 2014)

[4] Akgün, T., Koç, Z., Güner, Ş., Öztürk, B., Özkan, B., Üstün, Ö., Tuncay, N., Özgüner, Ü., "A Study on Autonomous Vehicle Development Process at Okan University", 2012 IEEE International Conference on Vehicular Electronics and Safety, 2012, 369-374

[5] DARPA Urban Challenge Route Network Definition File and Mission Data File Formats, 2007,1-14

[6] Özgüner, Ü., Acarman, T., Redmill, K., Autonomous Ground Vehicles, Artech House, USA, 2011

[7] Fu, L., Yazıcı,A.,Özgüner,Ü., "Route Planning For OSU-ACT Autonomous Vehicle in DARPA Urban Challenge", IEEE Intelligent Vehicles Symposium, 2008, 781-786

[8] Yazıcı,A.,Özgüner,Ü., Fu, L., "OSU-ACT Otonom Otomobili için sehir içi yol planlaması" 12. Elektrik Elektronik Bilgisayar Biyomedikal Mühendisliği Ulusal Kongresi, 2007,

[9]http://www.darpa.mil/default.aspx (Ziyaret tarihi Aralık 2014)

ÖZ GEÇMİŞ

Kader NİKBAY, 2011 yılında Eskişehir Osmangazi Üniversitesi Bilgisayar Mühendisliği bölümünü tamamladıktan sonra, 2012 yılında Okan Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği programında tezli yüksek lisans eğitimine başlamıştır.

İş deneyimleri arasında 2011 ile 2013 yılları arasında Bursa'da Yalın Software firmasında Yazılım Uzmanı olarak görev yapmıştır. 2013 yılında ise Okan Üniversitesi Bilgisayar Mühendisliği bölümünde Araştırma Görevlisi olarak göreve başlamış ve halen devam etmektedir.