YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÖNDER DENETİMLİ TAKIM CANLANDIRMA SİSTEMLERİ

Kontrol ve Bilgisayar Yük. Müh. Levent CUHACI

FBE Bilgisayar Mühendisliği Anabilim Dalında Hazırlanan

DOKTORA TEZİ

Tez Savunma Tarihi : 8 Temmuz 2008
Tez Danışmanı : Prof.Dr. B. Tevfik AKGÜN (YTÜ)
Jüri Üyeleri : Prof.Dr. Eşref ADALI (İTÜ)
: Prof.Dr. Oya KALIPSIZ (YTÜ)
: Prof.Dr. A. Coşkun SÖNMEZ (YTÜ)
: Yrd.Doç.Dr. A. Şima ETANER UYAR (İTÜ)

İSTANBUL, 2008

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

di	i. etmenin diğer etmenlere olan ortalama uzaklığı
d_{ort}	Etmenler arası ortalama uzaklık
n	Toplam etmen sayısı
N	Toplam küme sayısı
$O(\ )$	Zaman karmaşıklık fonksiyonu
p_i	i. etmenin konum vektörü
r	Etmenin algı bölgesi yarıçapı
S	Takım başarım puanı
x	Etmenlerin aldıkları ortalama yol
α	İki etmen arası yönelim farkı
θ_t	i. etmenin diğer etmenlerle olan ortalama yönelim farkı
θ_{ort}	Etmenler arası ortalama yönelim farkı
Δi	i. test eğrisinin ortalama eğri ile olan hata kareleri toplamı

KISALTMA LİSTESİ

ANOVA	Analysis of Variance
CPU	Central Processing Unit
OYD	Ortalama Yer Değiştirme
SPSS	Statistical Package for the Social Sciences
TBP	Takım Başarım Puanı
TDK	Türk Dil Kurumu
UML	Unified Modeling Language

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

ÖNSÖZ

Bilgisayarda canlandırma konusu gelişen teknolojiye paralel olarak başta eğitim, eğlence ve benzetim sistemleri olmak üzere pek çok sektörde kendine kullanım alanı bulabilmektedir. Özellikle son yıllarda davranışsal canlandırma teknikleri üzerinde yapılan çalışmalar ile canlandırma ortamında bulunan nesnelerin canlandırılmasında çok daha gerçekçi ve etkileyici sonuçlar elde edilmeye başlanmıştır.

Bu tezde gerçeklediğim önder denetimli takım canlandırması konusu ile, gerçek hayatta sürü davranışı sergileyen canlıların bir önder denetiminde hareket etmelerini sağlayıp sürü canlandırmalarında oluşabilecek sorunlara çözüm getirmeye; bunun yanı sıra canlandırmada çoklu takımların oluşması, bu takımların bölünmesi, tekrar birleşmesi ve takımlar arasındaki etkileşim gibi konulara da çözüm getirmeye çalıştım. Yaptığım bu çalışmanın, benzer konularda yapılacak diğer çalışmalara ışık tutmasını diliyorum.

Tez konusunun seçimi ile başlayıp tez teslimine kadar geçen süreçte bana aktardığı bilgi birikimi ve çalışma sürecinde bana aşıladığı olumlu motivasyon ile tezin bu noktaya gelmesinde büyük katkıları olan danışman hocam Prof.Dr. B. Tevfik AKGÜN'e; özellikle tez ilerleme aşamalarında yaptıkları değerlendirmeler ve verdikleri fikirler ile tezin yönünün belirlenmesini sağlamış olan sayın Prof.Dr. Eşref ADALI ve sayın Prof.Dr. Oya KALIPSIZ'a; tezin sonuçlanma aşamasında yaptıkları değerli katkılar için sayın Prof.Dr. Coşkun SÖNMEZ ve sayın Yrd.Doç.Dr. Şima Etaner UYAR'a; tezde elde edilen sonuçların istatistiksel analizi konusundaki yardımları için çalışma arkadaşım sayın Yrd.Doç.Dr. Hikmet ÇAĞLAR'a ve bu zorlu süreçte her zaman yanımda olup bana desteklerini esirgemeyen sevgili eşim Sibel Akpınar CUHACI'ya teşekkürlerimi bir borç bilirim.

Levent CUHACI

İstanbul, 2008

ÖNDER DENETİMLİ TAKIM CANLANDIRMA SİSTEMLERİ

Levent CUHACI
Bilgisayar Mühendisliği, Doktora Tezi

Bilgisayarda canlandırma konusu üzerinde son yıllarda yapılan çalışmalar, canlandırma ortamında bulunan etmenlerin hareketlerinin canlandırıcı tarafından önceden belirlenmesi yerine, çevresini algılayan ve edindikleri bilgilere göre akılcı bir karar verip buna ilişkin hareketleri uygulayan akıllı etmenlerin gerçeklenmesi üzerine yoğunlaşmıştır. Davranışsal canlandırma olarak da bilinen bu teknik sayesinde hem birey hem de sürü bazında gerçekçi canlandırmalar elde edilebilmektedir.

Sürü canlandırması konusunda yapılmış olan çalışmalarda gelişmiş bir sürü bireyi – önder ilişki modelinin bulunmaması ve bu konuda görülen boşluk, tez çalışmasının motivasyonunu oluşturmaktadır. Önerdiğimiz önderlik modeli kullanılarak sürüdeki bireylerin bir önder denetiminde hareket etmeleri sağlanmış ve böylelikle önder denetimli takım modeli geliştirilmiştir. Sürü canlandırmalarında karşılaşılabilen tıkanıklık problemlerinin çözümü için değişken önderlik kavramı önerilmiş ve devreye giren önder değişim algoritmaları sayesinde bu tıkanma problemlerinin çözülmesi sağlanmıştır. Ayrıca canlandırma süresince mevcut takımların dış etkenler ile bölünmesi ve bunların tekrar birleşmesi sonucu oluşabilecek farklı takımların etkileşimi, geliştirilmiş olan takım demetleme algoritması ile desteklenmiş ve takım etkileşimlerinin önderler tabanında da gerçeklenmesi sağlanmıştır.

Anahtar Kelimeler: Sürü canlandırması, akıllı etmen, önderlik, takım, tıkanma, takım demetleme.

JÜRİ:

1. Prof.Dr. B. Tevfik AKGÜN
2. Prof.Dr. Eşref ADALI
3. Prof.Dr. Oya KALIPSIZ
4. Prof.Dr. A. Coşkun SÖNMEZ
5. Yrd.Doç.Dr. A. Şima ETANER UYAR

Kabul tarihi: 08.07.2008
Sayfa Sayısı: 117

LEADER-CONTROLLED GROUP ANIMATION SYSTEMS

Levent CUHACI
Computer Engineering, Ph.D. Thesis

Recent works in computer animation field are focused on the use of intelligent agents – which can percieve their environment, make sensible decisions and act as a result of these decisions – rather than using of the pre-scripted animation techniques. Realistic animation sequences can be achieved with the use of intelligent agents and these techniques are also known as behavioral animation techniques.

The lack of an efficient group member-leader relationship model in the recent works on flock animation subject and the potential need in this field forms the motivation for this thesis. A new leader controlled group model has been developed with introducing a new leadership model and the leader controlled movement of group members has been achieved. With the suggested flexible and changeable leadership model, it's possible to detect and solve congestion problems of flock animations by triggering leadership change algorithms. Also disintegration of groups into smaller groups and reintegration of these groups during the animation have been supported with introducing a new grouping algorithm and interaction between these different groups has also been achieved by means of group members and group leaders.

Keywords: Flock animation, intelligent agent, leadership, group, congestion, group clustering.

JÜRİ:

1. Prof.Dr. B. Tevfik AKGÜN
2. Prof.Dr. Eşref ADALI
3. Prof.Dr. Oya KALIPSIZ
4. Prof.Dr. A. Coşkun SÖNMEZ
5. Yrd.Doç.Dr. A. Şima ETANER UYAR

Kabul tarihi: 08.07.2008
Sayfa Sayısı: 117

1. GİRİŞ

TDK canlandırma (*animation*) sözcüğünü "Tek tek resimleri veya hareketsiz cisimleri gösterim sırasında hareket duygusu verebilecek biçimde düzenleme ve filme aktarma işi" şeklinde tanımlar. Canlandırma sektörü günümüzde özellikle eğlence, sinema, eğitim vb. gibi alanların vazgeçilmez araçlarından biri durumuna gelmiştir. 1930'lu yıllardan itibaren Walt Disney Stüdyolarının öncülüğünde çizgi canlandırma şeklinde sinema alanında başlayan gelişmeler, bilgisayar teknolojisinin devreye girdiği yıllardan itibaren bilgisayar canlandırması adı altında büyük bir sektöre öncülük eder. Çizgi canlandırmada hikayeyi oluşturan her kare çizerler tarafından çizilip boyanırken, bilgisayarla canlandırmada tüm görüntüler bilgisayarın hesaplama gücünün de yardımıyla nesnelerin üç boyutlu modelleri kullanılarak üretilir. Çizgi canlandırmada kullanılan temel kurallar sonraki yıllarda bilgisayarla canlandırma tekniklerine de uygulanmıştır (Lasseter, 1987).

Bu bölümde önce bilgisayarla canlandırma konusundaki farklı yaklaşımlar kısaca ele alınacak; daha sonra ise tez çalışmasının dayanak noktası olan davranışsal canlandırma ve takım canlandırma konuları ile bu konularda yapılan temel çalışmalar daha detaylı olarak incelenecektir.

1.1 Bilgisayarla Canlandırma Yöntemleri

Bilgisayarla canlandırmadaki temel konu hareketin tanımlanmasıdır. Hareket denetim yöntemleri genel olarak sınıflandırmak gerekirse geometrik, fiziksel ve davranışsal yöntemler olmak üzere 3 ana grupta incelenebilir (Magnenat-Thalmann ve Thalmann, 1996).

1.1.1 Geometrik tabanlı canlandırma

Canlandırıcı kişi eğer geometrik bilgilerden yararlanacak ise, nesnelerin ve bu nesnelerin referans noktalarının koordinatları, açıları ve hızları hareket denetimindeki temel paramatreleri teşkil eder. Geometrik yaklaşımda kullanılan geleneksel programlama tekniği anahtar kare oluşturma olarak bilinen tekniktir (Burtnyk ve Wein, 1971; Magnenat-Thalmann ve Thalmann, 1990). Bu teknikte, nesnelerin canlandırmanın belirli noktalarındaki konumları ve durumları canlandırıcı tarafından belirlenir ve bilgisayar canlandırma anında ara boşlukları aradeğerleme ile doldurarak canlandırmayı sağlar. Canlandırma sinemasının önemli örneklerinden olan 1995 yapımı *Toy Story* adlı film, bu teknikle geliştirilmiştir. Fakat bu yöntem hiç kuşkusuz üzerinde oldukça dikkat ve emek gerektiren bir yöntemdir. Çünkü

canlandırma devam ettiği sürece bu nesneler belirli sınırlar dışına çıkmamalı, çarpışma olasılıkları düşünülmeli ve tüm bu denetimler canlandırılan bütün nesneler için tek tek ele alınmalıdır.

Geometrik yaklaşımda kullanılan yöntemlerden bir başkası olan hareket yakalama tekniğinde bir insanın (veya bir canlının) üç boyuttaki hareketleri manyetik ve optik cihazlar ile kaydedilir bu hareketler sonucu oluşan koordinat bilgileri bilgisayarda canlandırılacak olan sanal karaktere adapte edilir (Bruderlin ve Williams, 1995).

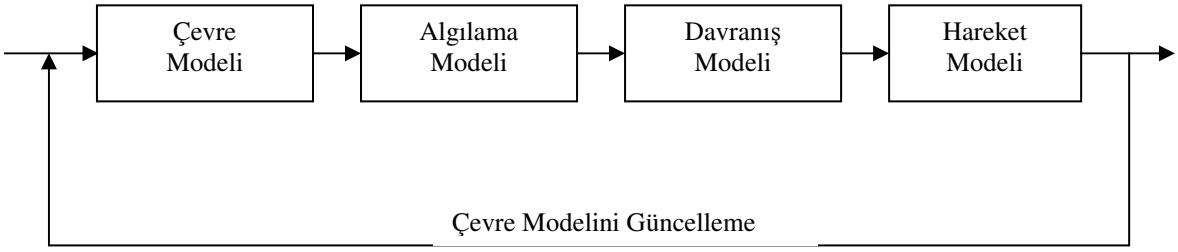
Bir diğer geometrik yaklaşım olan ters kinematik yönteminde ise (Badler vd., 1985) canlandırıcı, eklemli bir yapıdaki bir uç noktanın konumunu tanımlar ve bu konumu sağlayacak eklem açılarını bulma yoluna gider.

1.1.2 Fiziksel tabanlı canlandırma

Fiziksel tabanlı canlandırma yönteminde nesnelerin hareketleri birtakım fiziksel denklemler ile belirlenmektedir. Bu modelde her bir nesnenin hareketine ait kurallar ve kısıtlamalar tanımlanır. Gerçeğe daha yakın bir canlandırma için eğer varsa yerçekimi, rüzgar vb. gibi dış etkenler de dikkate alınır (Witkin vd., 1997). Ardından nesnelerin hareketleri fizik kuralları dahilinde bilgisayar tarafından hesaplanır. Fiziksel tabanlı yaklaşımlarda canlandırıcının canlandırma üzerindeki denetimi, diğer yöntemlere göre daha azdır.

1.1.3 Davranışsal canlandırma

Canlandırma konusunda son yıllardaki çalışmalar, geleneksel canlandırma teknikleri yerine davranışsal canlandırma (*behavioral animation*) tekniklerinin kullanımı üzerine yoğunlaşmıştır. Bir davranışsal canlandırma sisteminde her sanal karakter bir nevi özerk etmen (*autonomous agent*) olarak tasarlanır. Böylelikle sanal karakterler canlandırma süresince özerk olarak davranabilir: bulunduğu ortam modelinden anlık bilgileri algılar, bu bilgileri kendi iç durumunu da dikkate alarak yorumlar, yapacağı hareketi belirler ve bu hareketi uygular (Şekil 1.1). Bu çevrim canlandırma süresince kısa aralıklarla tekrarlandığında ortaya önceden planlanmış bir canlandırma yerine, çalışma anında şekillenen ve değişebilen ortam modeline göre farklılaşabilen bir canlandırma çıkmaktadır. Dolayısıyla davranışsal canlandırma, bilgisayarda grafik ve yapay zeka konularının kesişiminde yer alan bir ara disiplin olarak da düşünülebilir.



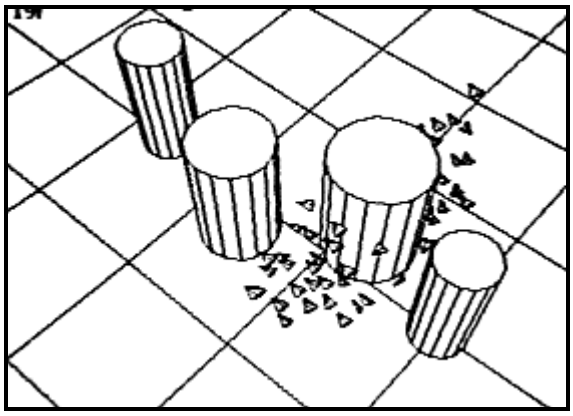
Şekil 1.1 Genel bir davranışsal canlandırma sistemi (Millar vd., 1999).

Çevre modeli, canlandırmanın gerçeklendiği sanal dünyanın içinde yer alan tüm nesneleri kapsayan bir modeldir. Belirli bir karakterin çevre modeli ise, o karakterin içinde bulunduğu sanal dünyadaki tüm nesneler ile tıpkı kendisi gibi algılama, davranış ve hareket modellerine sahip olan diğer karakterleri içerir. Tüm bu canlandırılan karakterler, çevresindeki durumdan haberdar olabilmek için farklı algılama teknikleri kullanabilirler. Buradan alınan bilgiler karakterin davranış modeline aktarılır. Davranış modelinde karakterin çevresine ne şekilde tepki vereceği belirlenir ve bu çıktı hareket modelinin girişi olur. Esas hareketi sağlayan model, hareket modelidir. Canlandırılan karakter aynı zamanda çevre modelinin bir parçası olduğundan bu işlemlerin sonucunda çevre modeli de güncellenmelidir (hareketlerin sonucunda oluşan yeni durumlar, yeni konumlar vb.).

1.2 Davranışsal Canlandırma Konusunda Yapılan Çalışmalar

Davranışsal canlandırma konusundaki ilk temel çalışma Reynolds'a (1987) aittir. Reynolds doğadaki hayvanların sürü hareketlerini gözlemlemiş ve buna ilişkin örnek bir bilgisayar simülasyonunu o ana kadar kullanılan geleneksel tekniklerden farklı olarak ele almıştır. Reynolds'un geliştirdiği uygulamada bir kuş sürüsünün hareketleri canlandırılmıştır. Yöntem olarak her bir kuşun canlandırma süresince izleyeceği yol önceden belirlenmemiş; bu etmenler bağımsız birer aktör olarak düşünülmüş ve böylece her bir kuşun canlandırma süresince kendi rotasını çizmesi sağlanmıştır (Şekil 1.2). Reynolds'un bu çalışmasında kuşların hareketini 3 basit temel kural belirlemektedir: Sürüye bağlılık, sürüden ayırma ve sürü ile hizalama.

Renault ve Magnenat-Thalmann (1990), bir davranışsal canlandırma sistemindeki aktörlerin bulundukları ortamı algılamasında kullanılabilecek yapay görüş yöntemini geliştirmişlerdir. Renault ve Thalmann uygulama olarak bir koridor geçiş problemini ele almış ve bu ortamda aktörlerin birbirlerine ve sabit duran nesnelere çarpmadan geçebilmeleri için bulundukları ortamı görebilmelerini bu yapay görüş yöntemi ile sağlamıştır.



Şekil 1.2 Reynolds'un sürü modeli (Reynolds, 1987).

Reynolds (1987) modelinde sürüdeki her bir kuş için tek bir algılama bölgesi kullanılmıştır. Oysa ki doğada algılama işleminin kesin veya belirsiz olduğu alanlar vardır (Takeuchi vd., 1992). Takeuchi, canlandırılan karakterin algı karakteristiğini modellerken, algılanan nesne ile ilgili karakterin arasındaki uzaklığı da göz önüne almıştır.

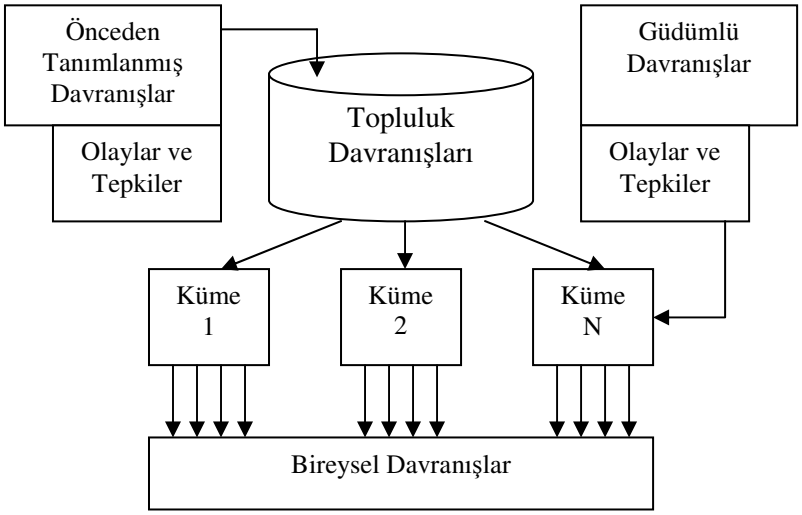
Terzopoulos ve Tu (1994), sanal bir akvaryum oluşturarak farklı tip ve davranıştaki balıkları biyomekanik olarak modellemiş ve bu şekilde gerçeğe çok yakın hareketler elde etmişlerdir (Şekil 1.3). Algılama yöntemi olarak Reynolds'un önerdiği yapay görüş tekniği kullanılmış ve ortam bilgisi incelenerek sanal balığın davranış modelindeki bir kuralın tetiklenmesi sağlanmıştır. Daha sonra bu tetikleme balığın yapması gereken hareketi denetleyen motor kontrolörüne iletilmiş ve balığın gerçeğe yakın canlandırması sağlanabilmiştir.

Şekil 1.3 Sanal akvaryum ve yapay balıklar (Tu ve Terzopoulos,1994).

Becheiraz ve Thalmann (1998), özerk etmenlere zihinsel ve duygusal parametreler tanımlayarak, bu parametrelerin etmenlerin karar verme modeline etki ettiği bir davranışsal canlandırma sistemi tasarlamışlardır.

Reynolds (1999), özerk etmenler için bireysel ve takım yönlendirme davranışlarını detaylı bir şekilde tanımlamıştır. Bayazit ve arkadaşları (2002), gerçekleştirdikleri canlandırmalarda kural tabanlı yol harita modelini kullanarak eve dönme, keşif, hedef arama ve dar geçitten geçme gibi davranışlar için yeni teknikler önermişlerdir.

Son dönemde sanal topluluk konusunda yapılan çalışmalar, daha çok sanal insan takımlarının oluşturulması ve denetimi üzerine yoğunlaşmıştır. Musse ve Thallman (2001) yaptıkları çalışmada, sanal topluluk kavramını sanal etmenlerden oluşmuş takımlar topluluğu olarak tanımlamışlardır. Önerdikleri hiyerarşik model (Şekil 1.4) ile sanal topluluk davranışlarının denetiminde; önceden tanımlı davranışların kullanımı, olay ve tepkilere dayalı davranışların kullanımı ve gerçek zamanlı dışarıdan müdahaleler olmak üzere 3 farklı yöntem sunmuşlardır.



Şekil 1.4 Hiyerarşik topluluk modeli (Musse ve Thallman, 2001).

Ulicny ve Thalmann (2001), etkileşimli sanal ortamlarda yer alan insan topluluklarının, özellikle panik durumlarındaki davranışlarını modellemişler ve gerçeğe yakın sonuçlar elde etmişlerdir. Panik ve bunun sonucu olarak bulunulan ortamı terketme durumlarında topluluğu oluşturan sanal etmenlerin bireysel davranışları ayrıca Braun ve arkadaşları (2003) tarafından da incelenmiştir. Treuille ve arkadaşları (2006) ise hareket halindeki farklı toplulukların süreklilik dinamikleri üzerinde çalışmışlardır.

Lebar Bajec ve arkadaşları (2005) sürü hareketlerinin canlandırılmasında ilk kez bulanık mantık yaklaşımını kullanmışlardır.

1.2.1 Önderlik konusunda yapılan çalışmalar

Sürü hareketlerinde önderlik konusu bilgisayar canlandırması dışındaki farklı alanlarda da araştırma konusu olmuş ve incelenmiştir. Kelly (1997) tarafında robotik alanında yapılan bir çalışma, fiziksel robotlarla sürü davranışının gerçeklenmesi üzerinedir. Kelly, Reynolds'un sürü dinamiklerinden yola çıkarak birbiriyle çarpışmadan bir takım şeklinde hareket eden ve deneyimlerinden öğrenen robotlar geliştirmiştir. Robotların sürü hareketlerini öndersiz ve önderli olarak iki farklı şekilde inceleyen Kelly, ortam engelleri ve fiziksel birtakım zorluklar nedeniyle önderliğin dinamik olması gerektiğini, yani zorunlu anlarda sürüde bir önder değişiminin gerekliliğini ortaya koymuştur. Robotik alanında yapılan bazı diğer çalışmalarda ise robotlar için merkezi olmayan sürü hareketi algoritması geliştirilmesi (Yan vd., 2008), belleksiz ve asenkron hareketli araçların sürü hareketine önder izleme modelli çözüm (Gervasi ve Prencipe, 2003) ve robot ağlarında önder seçimi (Canepa ve Gradinariu, 2007) gibi konular ele alınmıştır.

Li ve arkadaşları (2001), sanal insan topluluklarının hareketleri için bir önder izleme modeli geliştirmişlerdir. Her sanal topluluğa bir önder atanmış ve topluluk üyelerinin yapay yaşam ilkelerini kullanarak önderi takip etmesi gerçeklenmiştir. Çalışmada önderlerin rotalarının çakışması ise bir yol planlama yaklaşımı ile ele alınmıştır.

Aubé ve Shield (2004), gerçek hayatta mimar ve güvenlik sorumlularının karşılaştığı ciddi bir problem olan kısıtlı alanlarda kalabalık insan topluluklarının güvenli bir şekilde yönetilmesi konusunu önder tabanlı bir yaklaşımla ele almışlardır. Geliştirdikleri yazılımla, sınırları tanımlanabilen alanlarda farklı türden önderlerin topluluğun çevresinde farklı yerlerde konumlandırılmasına olanak sağlamışlar ve problemin çözümünde tatmin edici sonuçlar elde etmişlerdir.

Pelechano ve Badler (2006) tarafından yapılan bir çalışmada bir binanın tehlike anında boşaltılması eğitimli önder davranışının modellenmesi ile gerçeklenmiştir. Benzer konuda yapılmış olan bir başka çalışmada (Ji ve Gao, 2007) panik anında bulunulan ortamdan kaçış için önder-izleyici modeli benimsenmiş ve A* algoritması kullanılarak bireylerin çevredeki en uygun öndere yönlendirilmesi sağlanmıştır.

1.3 Tezin Kapsamı

Yapılan bu tez çalışmasında, önder denetimli takım canlandırma sistemi olarak yeni bir davranışsal canlandırma sistemi önerilmiştir.

Davranışsal canlandırma sistemleri konusunda yapılmış olan çalışmalar genel olarak birey bazında ve sürü (*flock*) bazında olmak üzere iki alanda yoğunlaşmıştır. Sürü canlandırması üzerinde yapılmış olan çalışmalarda, bireylerin yanındakini veya önderini izlemesi dışında özellikle üzerinde çalışılmış ve gelişmiş bir sürü bireyi-önder ilişki modelinin olmaması ve bu konuda görülen boşluk, tez çalışmasının motivasyonunu oluşturmaktadır. Genel olarak önceki çalışmalardaki önderlik kavramı; bulunulan ortamı tanıyan eğitimli bir bireyin diğer hiçbir bireyi takip etmemesi şeklinde gerçeklenen *sürü öncüsü* olarak seçimi ve diğer bireylerin seçilmiş olan bu öncüyü takip etmeleri şeklinde gerçeklenmiştir.

Reynolds (1987) tarafından önerilmiş olan *sürü* kavramı, bireylerin sadece yakın çevrelerinde bulunan diğer bireyler ile etkileşimi sonucu ortaya çıkan ve öndersiz hareket eden bireyler topluluğu olarak tanımlanmıştır. Öte yandan önerdiğimiz sistemde adı geçen *takım* kavramı ile bir önder denetiminde hareket eden bireyler topluluğu anlatılmaktadır.

Önceki bölümlerde incelenmiş olan çalışmalarda motivasyon, daha çok önderin akıllı yapılması konusunda yoğunlaşmıştır. Önder; ortamı tanıyan, gidilecek hedefi bilen, gerektiğinde birtakım akıllı algoritmaları kullanan (yol planlama, en kısa yolu bulma, diğer önderler ile çakışmama vb) ve kendisini izleyen "acemi" bireyleri bu hedefe yönlendiren bir etmendir. Kısacası önder-izleyici (*leader-follower*) modelinde ağırlık önder tarafındadır.

Önderin akıllı yapılması konusu, bu tez çalışmasının kapsamında değildir. Modelimizde önder, sürü içinden herhangi bir etmen olabilir; sürüdeki diğer etmenlerden farkı ise varılacak hedefi bilmesi ve oraya yönlenmesidir. Dolayısıyla ortak amaç, önder üzerinden önderi takip eden sürü etmenlerine de verilmiş olmaktadır.

Bu tezde önerilen önder denetimli takım canlandırma sistemi ile önceki uygulamalar arasındaki fark için kısaca şunları söyleyebiliriz: Önerdiğimiz önderlik modellerinden olan değişken önderlik modeli ile canlandırma süresince takım hareketlerinde oluşabilecek tıkanma durumlarında önder değişim algoritmalarının devreye girmesi sağlanmıştır. Bunun sonucu olarak ise sistemde bir önder değişimi yapılarak canlandırmanın sürekliliği sağlanmıştır. Takımı oluşturan bütün bireylerin canlandırmanın her adımında akıllı davranış üretmesi yerine sadece önderin seçenekli davranması hedeflenmiş ve böylelikle canlandırmada sürekliliğin sağlanmasının yanı sıra sistem başarımında da iyileşme elde edilmiştir. Ayrıca canlandırma süresince mevcut takımların dış etkenler ile bölünmesi ve bunların tekrar birleşmesi sonucu oluşabilecek farklı takımların etkileşimi, geliştirilmiş olan takım demetleme algoritması ile desteklenmiş ve takım etkileşimlerinin önderler tabanında da gerçeklenmesi sağlanmıştır.

Özetle takım canlandırmasında etkin önderlik modeli geliştirilmiş ve önderi takip eden bireylerin, dolayısıyla da takım canlandırmasının daha gerçekçi olması hedeflenmiştir.

Tezin bundan sonraki bölümlerinin kısa birer özeti aşağıda verilmiştir.

- 2. bölümde, etmen tabanlı davranışsal canlandırma benzetim sistemleri incelenmiş ve gerçeklemede temel olarak seçilen Buckland benzetim sistemi (Buckland, 2005) detaylı olarak tanıtılmıştır.

- 3. bölümde, önerdiğimiz sistemin temel birimleri olan takım, önder ve takım bireyi modelleri tanıtılmıştır.

- 4. bölümde, sürü canlandırmasında karşılaşılan en önemli sorun olan tıkanıklık durumu incelenmiş; tıkanıklığın sezilmesi ve canlandırmadaki akıcılığın sağlanması yönünde çözüm olarak önerdiğimiz önder değişim algoritmaları tanıtılmıştır.

- 5. bölümde, önerdiğimiz önder denetimli takım canlandırma sisteminin gerçekleme aşamaları ele alınmıştır.

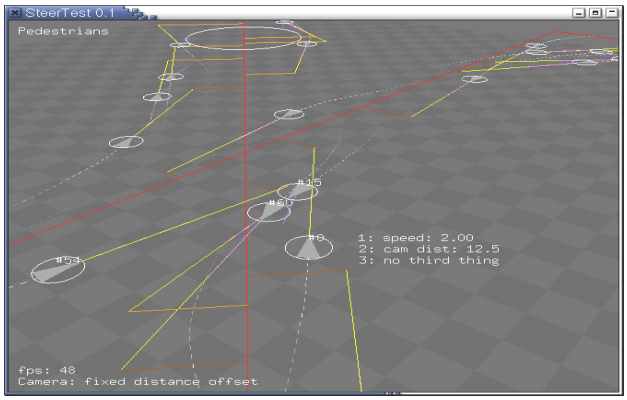
- 6. bölümde, önerdiğimiz önder değişim algoritmaları engelsiz ve engelli ortamda gerçekleştirilmiş canlandırmalar üzerinde sınanmış ve elde edilen verilerin istatistiksel analizleri yapılmıştır. Önder değişimini tetikleyen önemli parametrelerin duyarlılık analizleri de yine bu bölüm içinde ele alınmıştır.

- 7. bölümde, önerdiğimiz sistemin bir uygulaması olarak çoklu takımların oluşumu ve bu takımların denetimini sağlamak için önerilmiş olan takım demetleme algoritması tanıtılmış; gerek aynı tür, gerekse farklı türden bireylere sahip takımların birbirleriyle olan etkileşimleri incelenmiştir.

- 8. bölümde tez çalışmasından çıkan sonuçlar ve önerilen ileri çalışma konuları özet olarak açıklanmıştır.

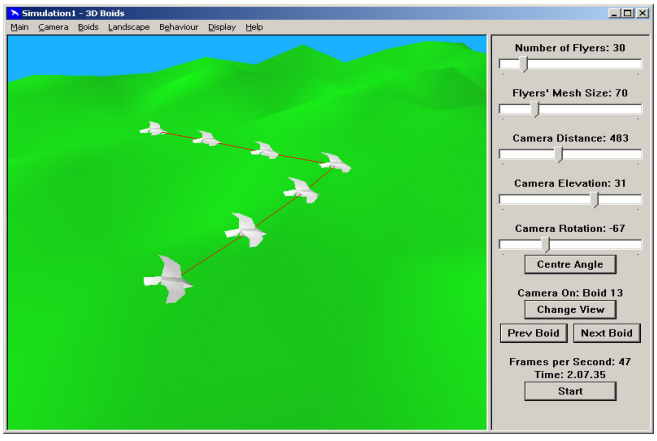
2. DAVRANIŞSAL CANLANDIRMA BENZETİM SİSTEMLERİ

Davranışsal canlandırma konusundaki öncü çalışmanın sahibi olan Reynolds, özerk etmenlerin yönlendirilmesine ihtiyaç duyan uygulamalarda yararlanılabilecek *OpenSteer*[*] adında açık kaynak kodlu C++ kütüphanesi ve bu kütüphaneyi kullanan örnek bir benzetim sistemi geliştirmiştir (Şekil 2.1). *OpenSteer* programcıya soyut bir hareketli etmen modeli sunar ve programcının bu model üzerinde çeşitli yönlendirme davranışlarını uygulayabilmesine olanak tanır.



Şekil 2.1 OpenSteer benzetim sistemi

Robert Platt tarafından geliştirilmiş OpenGL tabanlı *3D Boids*[**] uygulaması, parametreleri kullanıcı tarafından girilen bir kuş sürüsünün hareketlerini kullanıcıya farklı bakış açılarından sunabilmektedir (Şekil 2.2). Bu benzetim sistemi de sürü canlandırmasında Reynolds'un ortaya koyduğu kuralları esas almaktadır.



Şekil 2.2 3D Boids benzetim sistemi

[*] http://opensteer.sourceforge.net/ , 2003.
[**] http://www.navgen.com/3d_boids/, 1999-2004.

Bir başka davranışsal canlandırma benzetim sistemi, etmen topluluklarının canlandırılması için Java programlama dili kullanılarak geliştirilmiş, açık kaynak kodlu *Crowd Simulation Framework*[*] uygulamasıdır.

Bunların dışında pek çok ticari amaçlı canlandırma uygulaması davranışsal canlandırma desteği de sağlamaktadır. Canlandırma konusundaki en yaygın ticari yazılımlardan biri olan *Maya*[**] programı için geliştirilmiş olan *Crowd Make*r[***] davranışsal canlandırma motoru, bunlara bir örnek olarak verilebilir.

Tez çalışmasında belirlenen hedefler doğrultusunda ilk çalışmalara başlayabilmek için etmen tabanlı bir benzetim sisteminin gerekliliği ortaya çıktığında, Reynolds'un (1999) ortaya koyduğu temel yönlendirme davranışlarının gerçeklendiği bir canlandırma programı (Buckland, 2005) çatı olarak seçildi. Buckland benzetim sistemi esas olarak *OpenSteer* kütüphanesini temel almakta ve nesneye yönelik programlama tekniğini kullanması sebebiyle geliştirilmeye açık esnek bir davranışsal canlandırma modeli sunmaktadır. Sistemin uygulama temeli olarak seçilmesinin bir diğer sebebi de, tez çalışmasında üzerinde yoğunlaşılacak olan algoritmaların sınanması ve başarım ölçümleri için uygunluğudur.

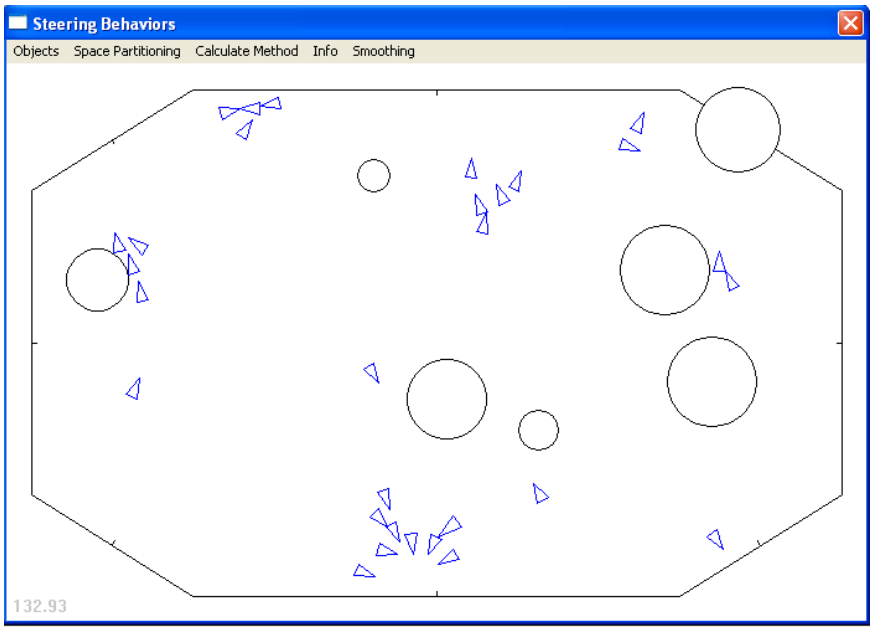
Bu bölümde, Buckland benzetim sisteminde var olan etmen bazındaki temel yönlendirme davranışlarının kısa birer özeti verilmiş ve yine etmen bazında gerçeklenip sisteme eklenmiş olan diğer yönlendirme davranış modelleri tanıtılmıştır.

2.1 Buckland Benzetim Sistemi

Tez çalışması için çatı olarak alınan sistem, bağımsız bir programcı ve aynı zamanda "*AI Interface Standards"* komitesinin bir üyesi olan Mat Buckland (2005) tarafından geliştirilmiştir. Sisteme ilişkin genel bir ekran görüntüsü Şekil 2.3'de verilmiştir. Ortamda bulunan mavi üçgenler özerk etmenleri, daireler engelleri ve düz çizgiler ise canlandırma alanını sınırlayan duvarları temsil etmektedir. Etmenler duvar, engeller ve birbirleri ile çarpışmadan hareket etmekte ve aynı zamanda Reynolds'un (1987) sürü davranışı için ortaya koyduğu temel kuralları işletmektedirler. Böylece rastgele gezinirken birbirleri ile karşılaşan her etmen sürüye dahil olmakta ve ortaya gerçekçi bir sürü davranış görüntüsü çıkmaktadır. Örnek olarak Şekil 2.3'de görülen senaryoda birbirinden ayrık kümelerin oluşmasının sebebi, bu kümelerin canlandırma ortamında henüz birbirleriyle karşılaşmamış olmasından

[*] http://sourceforge.net/projects/crowd, 2004.
[**] http:// www.autodesk.com/maya
[***] http://web.tiscali.it/maya_tutorial/

kaynaklanmaktadır. Canlandırmanın ilerleyen adımlarında bu kümelere ait etmenlerin birbirlerinin algı bölgelerine girmesi sonucu küme birleşmeleri oluşabilmektedir. Benzer şekilde küme içindeki bir etmenin farklı nedenlerden dolayı kümeden ayrılabilmesi de olağan bir durumdur.



Şekil 2.3 Buckland benzetim sistemi

2.1.1 Sistemin temel birimi olarak etmen

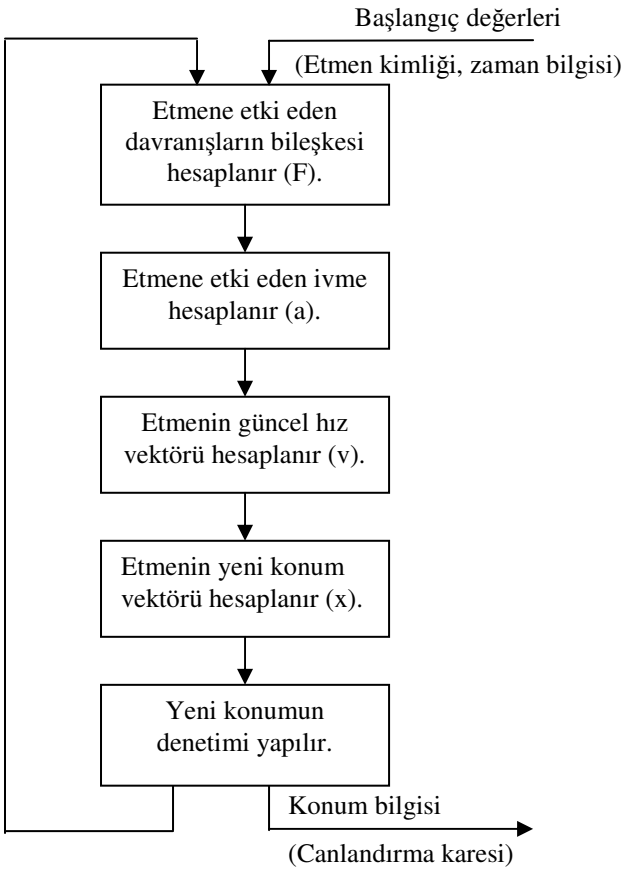
Gerçek hayatta rastlanabilen farklı sürü oluşumlarında bir birey, çevresinden durum bilgisini algılayan ve bu bilgiyi kullanarak hedefi doğrultusunda bir sonraki hareketine karar veren akıllı bir varlık olarak düşünülebilir. Bu akıllı bireylerin bilgisayarla canlandırma sistemlerinde modellenmesinde, önceden planlanmış hareketler yerine çalışma anında değişen ortam koşullarına göre farklı hareketler sergileyebilen, dolayısıyla bir algı-karar mekanizmasına sahip nesneler olarak tasarlanmaları gerekir.

Pek çok davranışsal canlandırma sisteminde olduğu gibi Buckland benzetim sisteminde de etmen olarak adlandırılan ve çevresinden aldığı bilgileri değerlendirip senaryodaki hedefi doğrultusunda bir sonraki adımda uygulayacağı en uygun harekete karar veren birimler canlandırmanın yapı taşlarını oluşturmaktadır. Sürü canlandırması ise bu etmenlerin birbirleri ve çevresiyle olan etkileşimleri sonucu ortaya çıkar.

Sistemde bulunan tüm nesneler *BaseEntity* isimli ana sınıfın alt sınıflarından türetilmiştir. *MovingEntity* hareket eden nesnelere ilişkin genel sınıfı, *Vehicle* ise etmenlere ilişkin özel

sınıfı teşkil etmektedir. *Behaviors* sınıfı ise her bir etmene uygulanabilecek yönlendirme davranışlarına ilişkin yöntemlerin gerçeklenmiş olduğu sınıftır.

Bir etmenin canlandırmanın bir sonraki adımındaki konumunu güncelleyen ve dolayısıyla her çerçevede çalıştırılan yöntem, *Vehicle* sınıfına ait *Update* yöntemidir. Bu yöntemde asıl olarak fiziksel modelleme ve temel fizik kurallarına göre hesaplama yapılmaktadır.



Şekil 2.4 Etmen güncelleme çevrimi

Bir davranışsal canlandırma modelinde yer alan bir etmene pek çok kuvvet etki edebilir. Örneğin etmen belirli bir hedefe doğru yönlenmişken, yoluna çıkan engellerden kaçınması gerekebilir. Bu örnekte iki farklı vektörel büyüklükten söz edebiliriz; etmeni hedef noktaya yönelten vektörel büyüklük ve önündeki engele çarpmasını engelleyecek olan vektörel büyüklük. Bu iki büyüklüğün bileşkesi her zaman vektörel toplam olmayabilir. Engelden kaçınma daha öncelik verilmesi gereken bir davranış modeli olduğu için buradaki bileşke kuvvet hesabında ağırlıklı olarak yerini alacaktır.

Bir etmenin kütlesi ve etmene etki eden bileşke kuvvet biliniyorsa, Newton'un hareket yasaları kullanılarak etmenin yapacağı yer değiştirme – dolayısıyla bir sonraki adım için etmenin yeni konumu – bulunabilir.

Buna göre bir etmenin konumu için Şekil 2.4'de de verilmiş olan güncelleme çevrimi şu şekildedir:

1. Etmene etki eden yönlendirme kuvvetlerinin bileşkesi bulunur ($\vec{F}$).
2. Etmene etki eden ivme vektörü ($\vec{a} = \vec{F} / m$) eşitliği kullanılarak hesaplanır.
3. Etmenin hız vektörü ($\vec{v} = \vec{a}t$) eşitliği kullanılarak hesaplanır.
4. Etmenin yeni konum vektörü, ($\vec{x} = \vec{v}t$) eşitliği ile bulunur.
5. Hesaplanan yeni konumun canlandırma ortamı sınırları içinde olması sağlanır.

Etmenlerin sergileyebilecekleri her davranış modeli, *Behaviors* sınıfına ait ayrı birer yöntemdir ve her birinin sonucu olarak vektörel bir büyüklük (etmenin yanıtı) üretilmektedir.

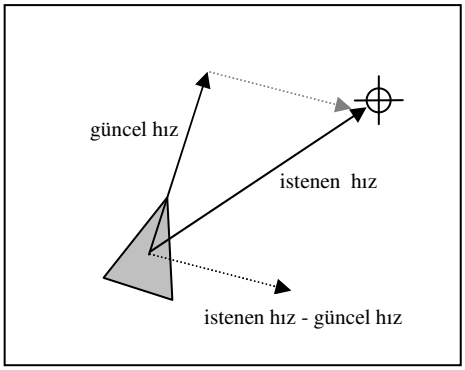
2.2 Temel Yönlendirme Davranışları

Kullanılan benzetim sisteminde yer alan ve her bir etmene ayrı ayrı uyarlanabilecek temel yönlendirme davranışları aşağıda listelenmiştir.

- Yönelme (*Seek*)
- Kaçınma (*Evade*)
- Varma (*Arrive*)
- Takip (*Pursuit*)
- Kaçma (*Flee*)
- Rastgele Gezinme (*Wander*)
- Engelden Kaçınma (*Obstacle Avoidance*)
- Duvardan Kaçınma (*Wall Avoidance*)
- Araya Girme (*Interpose*)
- Saklanma (*Hide*)
- Yol İzleme (*Path Following*)
- Mesafeli Takip (*Offset Pursuit*)

2.2.1 Yönelme davranışı

Yönelme davranışına ilişkin yöntem, bir etmeni hedeflenen bir konuma yöneltecek şekilde bir kuvvet geri döndürür (Şekil 2.5). Öncelikle istenen hız vektörü hesaplanır. Bu vektör, ideal bir dünyada etmenin hedef konuma ulaşması için gerekli olan vektörel büyüklüktür ve etmenden hedefe doğru, etmenin erişebileceği maksimum hızı geçmeyecek şekilde çizilen vektördür.



Şekil 2.5 Yönelme davranışı

Bu yöntem sonunda geri döndürülen yönlendirme kuvveti, etmenin o andaki hız vektörüne eklendiğinde hedefe doğru istenen hız vektörünü oluşturacak olan kuvvettir. Dolayısıyla bu kuvveti hesaplamak için istenen hız vektöründen etmenin o anki hız vektörünün farkını almak yeterli olacaktır.

Yönelme davranışı, diğer pek çok davranış modelinde kullanılan temel bir davranış tipidir.

2.2.2 Kaçınma davranışı

Kaçınma, bir hedefe yönelme davranışının tam tersidir. Etmeni bir hedefe doğru yönlendiren kuvveti üretmenin tersine, etmenin o hedefin tam ters yönünde kaçmasını sağlayan bir kuvvet üretilir. Kaçınma yöntemi ile yönelme yöntemi arasındaki tek fark, istenen hız vektörünün hesaplanması sırasında yapılan çıkarma işleminde bileşenlerin yer değiştirmesidir. Bu da, yönelme ve kaçınma davranışları sonucunda ortaya çıkan vektörler arasındaki farkın 180º olması anlamına gelir.

Kaçınma davranışını daha gerçekçi yapmak için, etmenin görüş alanı kavramı yönteme dahil edilebilir. Buna göre kaçınma yönteminin sonunda geri döndürülecek olan vektörel büyüklük, kaçılacak hedef etmenin görüş alanı içinde ise belirli bir büyüklükte olacaktır; aksi taktirde geri dönecek olan vektörel büyüklüğün değeri sıfır olacaktır.

2.2.3 Varma davranışı

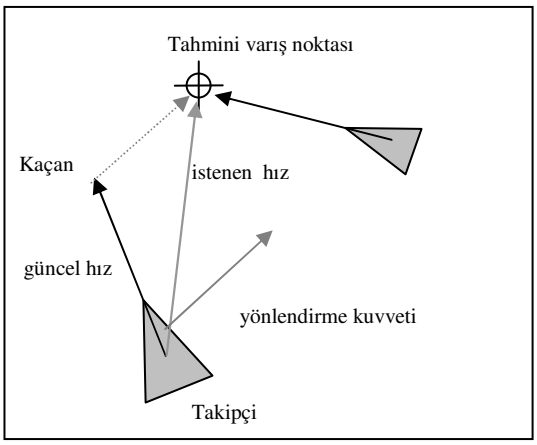
Bölüm 2.2.1'de bahsedilmiş olan yönelme davranışı, bir etmeni belirli bir istikamete yöneltmek için çok kullanışlı olmasına rağmen, bazı durumlarda etmenin belirli bir hedefe yaklaştığında yavaşlaması ve bu hedefe ulaştığında ise hızının sıfır olması (yani durması) istenebilir. Bunun için kullanılması gereken yönlendirme davranışı, varma davranışıdır.

Varma davranışı etmen hedeften uzak bir konumdayken yönelme davranışı gibi çalışır; etmen hedefe yaklaştıkça (aradaki uzaklık azaldıkça) gittikçe azalan hızlarda yaklaşmaya devam edilir ve hedefe varıldığında ise aradaki uzaklık sıfırlandığı için büyüklüğü 0 olan bir vektörel kuvvet geri döndürülür.

2.2.4 Takip davranışı

Herhangi bir davranışsal canlandırma modelinde yer alan en önemli davranışlardan biri, takip davranışıdır. Bu davranışa gerçek hayatta da oldukça sık rastlarız. Bir hayvan sürüsünü güden bir çoban veya bir köpek, annesini takip eden bir çocuk, avının peşinden koşan bir yırtıcı, bu davranışı sergileyen canlılara verilebilecek en temel örnekler arasındadır.

Takip davranışının gerçeklenmesinde izlenebilecek ilk yöntem, takipçi etmenin kaçan etmen istikametinde yönelme davranışını gerçekleştirmesidir. Buradaki temel fark, yönelme hedefinde olduğu gibi hedefin sabit değil, yer değiştiren bir hedef olmasıdır. Fakat bu yöntem, teoride kolay gerçeklenmesine rağmen, pratikte görülebileceği gibi çok akıllıca bir yöntem değildir. Daha iyi bir yöntem, takipçinin kaçan etmenin belirli kısa bir zaman dilimi sonunda varacağı noktayı tahmin ederek, bu noktaya yönelmesidir. Bu durum Şekil 2.6'da gösterilmektedir.



Şekil 2.6 Takip Davranışı

Takip davranışının gerçeklendiği yöntemin başarısı, takipçinin kaçan etmenin gideceği konumu ne kadar iyi tahmin edildiğine bağlı olarak değişebilir. Özel bir durum olarak; eğer takip edilen etmen takipçinin ilerleme yönünde ise (takipçinin hareket yönünün önünde ve bu vektörün en fazla 20 derece sapmasında ise) takipçi, kaçan etmeni hedef olarak alıp bu hedefe doğru yönelme davranışını sergilemelidir. Böylece takip davranışı, fazla bir hesaplamaya gerek duymadan yönelme davranışına dönüştürülmüş olmaktadır.

Yöntemin başarısını belirleyen en önemli faktör, takipçi etmenin ne kadar ilerisini tahmin edeceğinin belirlenmesidir. Bu süre, her iki etmen arasındaki uzaklık ile doğru orantılı (etmenler birbirine yaklaştıkça takipçi etmenin hareket yönünü değiştirme açısı artacaktır, dolayısıyla çok daha yakın bir zaman sonrasını tahmin edebilmelidir), etmenlerin hızları ile de ters orantılıdır.

2.2.5 Kaçma davranışı

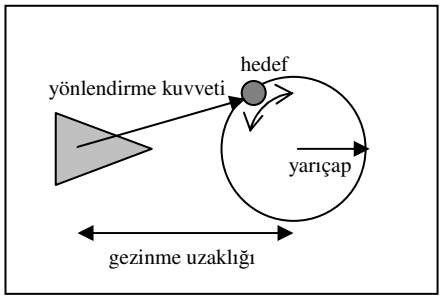
Kaçma davranışı, takip davranışı ile tek bir fark dışında birebir aynıdır. Bu fark; takip edilen etmenin takipçinin gideceği konumu tahmin etmesi ve bu hedeften kaçınma davranışını sergilemesidir.

2.2.6 Rastgele gezinme davranışı

Bir davranışsal canlandırma modelinde en çok kullanım alanı bulan etmen yönlendirme davranışı rastgele gezinme davranışı olarak bilinen davranıştır. Çoğu zaman etmenin belirli bir hedefi olmadığı durumlarda içinde bulunduğu ortamda amaçsızca gezinmesi istenir. Bunun için akla gelebilecek ilk yöntem, rastgele bir yönlendirme kuvveti üretilip etmenin bu kuvvet doğrultusunda hareket ettirilmesidir. Fakat bu durumda etmen kararsız hareketler, keskin dönüşler gibi göze hoş görünmeyen ve gerçekçilikten uzak davranışlar sergileyebilir. Bunu önlemek için Reynolds (1999) tarafından geliştirilen bir çözüm, etmenin hemen önünde sanal bir çember oluşturup, etmeni bu çember üzerinde kısa yer değiştirmeler yapan bir hedef noktasına doğru yöneltmektir.

Şekil 2.7'den de görülebildiği gibi, rastgele gezinme davranışının gerçekçiliğini belirleyen 3 temel parametre bulunmaktadır. Bunlar:

- Oluşturulan sanal çemberin yarıçapı,
- Etmenin sanal çembere olan uzaklığı,
- Hedef noktanın sanal çember üzerinde yapabileceği maksimum yer değiştirmedir.

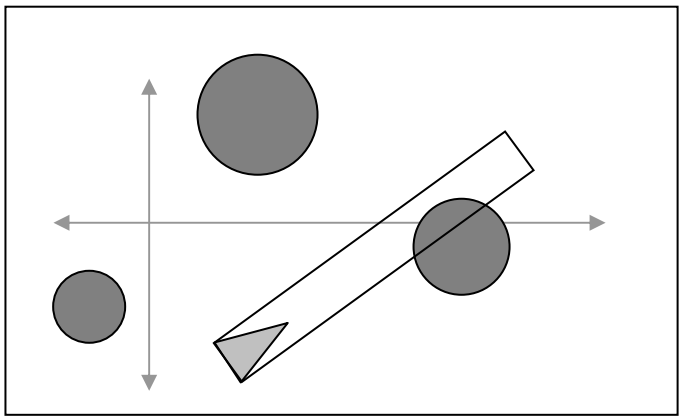


Şekil 2.7 Rastgele gezinme davranışı

Oluşturulan sanal çemberin etmenden olan uzaklığı, etmenin rastgele gezinirken yapabileceği dönüşleri belirlemedeki en önemli unsurlardan biridir. Sanal çember ne kadar uzakta olursa, etmenin yapabileceği dönüş açıları o kadar küçülecek ve dolayısıyla etmen doğrusala yakın bir hareket gerçekleştirecektir. Oluşturulan sanal çemberin etmene yakın olması durumunda ise, etmenin gezinirken daha büyük açılarla dönüş yapabilmesine (dolayısıyla daha keskin dönüşlere) olanak tanıyacaktır. Benzer şekilde sanal çemberin yarıçapı ve hedef noktanın yapabileceği maksimum yer değiştirme de, rastgele gezinme davranışının gerçekçiliğini belirleyen parametrelerdendir.

2.2.7 Engelden kaçınma davranışı

Engelden kaçınma davranışının gerçeklendiği yöntem, bir etmen hareket ederken önüne çıkabilecek çeşitli sabit nesneler ile çarpışmasını engelleyecek şekilde bir yönlendirme vektörü geriye döndürür. Engel olarak tanımlanan hareketsiz nesne, kendisini çevreleyen bir daire ile temsil edilebilen herhangi bir cisim olabilir. Etmenin olası bir çarpışmayı önceden sezebilmesi için bu etmenin hareket ettiği yön doğrultusundaki sanal bir dikdörtgenden yararlanılır. Bu durum Şekil 2.8'de gösterilmektedir.



Şekil 2.8 Engelden kaçınma davranışı

Etmenin de içinde olduğu sanal dikdörtgenin eni, etmenin eni ile aynıdır. Dikdörtgenin boyu ise canlandırma süresince etmenin hızı ile doğru orantılı olacak şekilde değişir. Zira etmen hızlı hareket ederken olası bir çarpışmayı belirli bir süre önceden sezebilmek için oluşturulan bu hayali dikdörtgenin boyu etmenin yavaş hareket ettiği durumdakine göre daha uzun olmalıdır. Buna göre olası bir çarpışma sezildiğinde öncelikle hayali dikdörtgen ile engellerin kesişim noktaları arasından en yakın olanının bulunması gereklidir. Olası çarpışma noktalarını bulmak için çizgi-daire kesişim testini uygulamak yeterli olacaktır.

En yakın kesişim noktası bulunduktan sonra, bu noktaya en yakın olan engelden uzaklaştırıcı bir yönlendirme kuvvetinin üretilmesi gerekir. Bu kuvvetin iki bileşeni bulunmaktadır : yavaşlatıcı kuvvet ve yanal kuvvet. Yavaşlatıcı kuvvet, etmenin hareket yönünün tersine oluşan bir kuvvet olup etmenin yavaşlamasını sağlar; yanal kuvvet ise engel ile çarpışmayı önleyecek olan ve engelin aksi yönünde oluşan bir kuvvettir. Bir başka deyişle, üretilecek yönlendirme kuvvetinin x bileşenini yavaşlatıcı kuvvet, y bileşenini ise yanal kuvvet belirler.

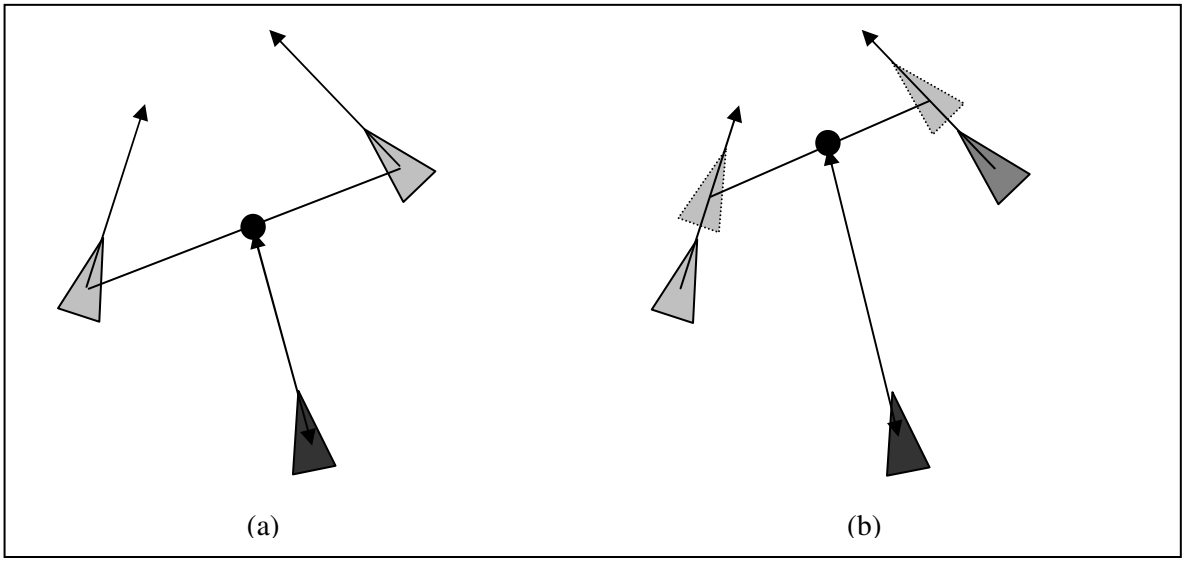
2.2.8 Duvardan kaçınma davranışı

Bir canlandırma sisteminde duvar, bir dizi doğru parçasının birleşimi olarak düşünülebilir. Duvardan kaçınma davranışı, etmenin duvarın herhangi bir parçası ile olası çarpışmasını önceden denetlenerek, bu çarpışmayı önleyici yönde bir yönlendirme kuvveti oluşturulması esasına dayanır. Bunun için etmenin önünde ve yanlarında sanal algılayıcılar oluşturulur. Aslında bu sanal algılayıcılar birer doğru parçası olup her aşamada bu doğru parçaları ile duvarı oluşturan doğru parçalarının kesişim testi yapılır. Herhangi bir kesişim olması durumu ilerde oluşabilecek olası bir çarpışmayı işaret ettiği için, ilgili doğru parçasının normal vektörü yönünde (yani duvardan uzaklaşılacak yönde) bir yönlendirme kuvveti geri döndürülür.

2.2.9 Araya girme davranışı

Araya girme davranışı, bir etmeni arasına girilecek olan diğer iki etmenin orta noktasına doğru yönlendirir. Yani arasına girilecek olan iki etmeni birleştiren çizginin orta noktasına işaret edecek şekilde bir vektörel büyüklük oluşturulur. Bu davranış canlandırma modelinde gütme davranışı olarak da ele alınabilir. Örneğin bir çobanın veya bir köpeğin, iki koyunu denetleyebilmesi için bu koyunları birleştiren çizginin orta noktasına doğru yönelmesi, böylece her iki koyunu da denetimi altına alması mümkün olacaktır.

Tıpkı takip davranışında olduğu gibi, araya girecek olan etmenin diğer iki etmenin belirli bir *t* süre sonra bulunacağı noktaları tahmin edip bu noktalar arasındaki orta noktaya yönelmesi daha gerçekçi ve başarılı bir davranış modeli olacaktır. Buradaki problem, *t* zamanının değerinin ne olması gerektiğidir. Bu parametrenin kesin bir değeri olmamakla birlikte, hesaplanmasındaki en uygun yöntemlerden biri Şekil 2.9'da gösterilmektedir.



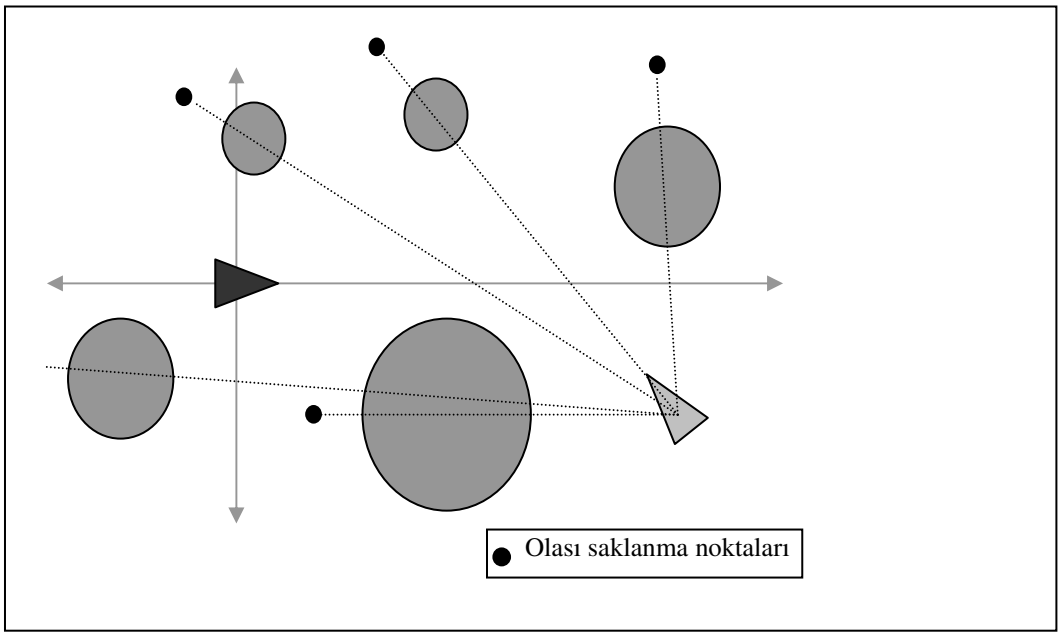
Şekil 2.9 Araya girme davranışı

Bulunulan zaman çerçevesinde hedef iki etmen arasındaki nokta hesaplanır ve araya girecek olan etmenin bu noktaya olan uzaklığı, etmenin maksimum hızına bölünerek *t* süresi bulunur. Daha sonra öndeki iki etmenin bu *t* süre sonra alacağı yol ve bulunacakları konum hesaplanır, bu son konumları birleştiren doğru parçasının orta noktası, hedef nokta olarak belirlenir. Son aşama olarak ise araya girecek olan etmenin bu noktaya varış davranışını gerçeklemesidir.

2.2.10 Saklanma davranışı

Saklanma davranışını sergileyen bir etmen, avcı olarak nitelendirilebilen diğer bir etmen ile arasında mutlaka bir engel olmasını sağlayabilecek en uygun noktaya yönlenir. Bir davranışsal canlandırma modelinin gerçekçiliğini sağlayan en önemli davranışlardan birisidir ve pek çok senaryoda kullanım alanı bulabilir; yırtıcıdan kaçan bir hayvan, hedefine sessizce yaklaşmaya çalışan bir avcı bu senaryolar için başlıca örnek olarak verilebilir.

Algoritmik açıdan incelendiğinde bu davranışın gerçekleştirilmesi için öncelikle bakılması gereken, ortamda bulunan engellere ilişkin saklanma noktalarıdır. Bu noktaların bulunma yöntemi, Şekil 2.10'da görülebilmektedir.



Şekil 2.10 Saklanma davranışı

Parametre olarak bir engelin konumu, bu engelin yarıçapı ve saklanılacak etmenin konum bilgilerini alan bir yöntem, bu engele ilişkin saklanma noktasının konum bilgilerini içeren bir vektör geri döndürür. Bunu yaparken öncelikle avcı etmenden engele doğru oluşturulan vektör hesaplanır, bu vektör normalize edilir ve saklanılacak noktanın engelden ne kadar uzakta olabileceğini içeren bir sabit ile çarpılır ve sonuç vektör engelin konum vektörüne eklenir.

Bu yöntem, saklanma davranışının gerçeklendiği ana yöntem içinden ortamda bulunan her engel için çağrılmaktadır. Sonuç olarak bulunan olası saklanma noktalarından en yakın olanına varma davranışı gerçekleştirilir. Eğer ortamda herhangi bir engel yoksa, bu durumda saklanacak olan etmenin sergileyeceği davranış, kaçma davranışı olacaktır.

2.2.11 Yol izleme davranışı

Yol izleme davranışını gerçekleştiren bir etmen, bu davranışı sürdürdüğü sürece belirli bir rota üzerinde yer alan noktalara sırayla yönelme hareketini gerçekleştirir. Burada rota veya yol olarak nitelendirilen bilgi, aslında belirli sayıda noktanın koordinatlarının bulunduğu bir sınıftır. Yol normal veya kapalı çevrim olabilir; normal yollarda son noktada durma işlemi gerçekleştirilir; kapalı çevrim yollarda ise son nokta ilk nokta ile birleştirilir, dolayısıyla etmenin bu noktaları ziyareti sürekli olarak devam eder.

2.2.12 Mesafeli takip davranışı

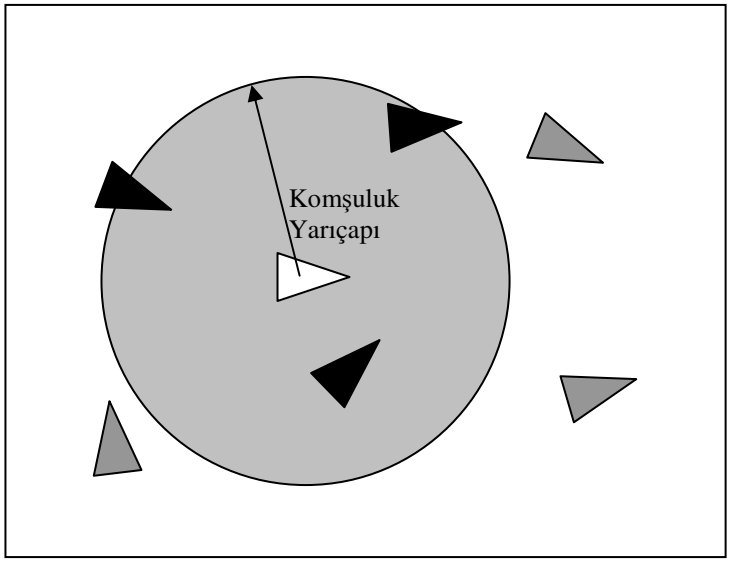
Mesafeli takip, takip eden etmen ile hedef etmen arasında her zaman belirli bir uzaklığın korunması demektir. Bu davranış modeli, özellikle sürü canlandırmalarında belirli bir formasyonun oluşması için kullanılabilecek uygun davranışlardan birisidir. Belirli bir öncü etmen ve bu öncüyü belirli bir uzaklıktan takip eden sürü, göze hoş gelen bir canlandırma modeli sergiler.

Mesafe bilgisi, iki boyutlu düzlemde vektörel bir büyüklük olup, her zaman için takip edilen öncünün yerel koordinat uzayında tanımlıdır. Dolayısıyla yapılması gereken ilk iş, bu mesafe vektörünü yerel koordinat düzleminden genel koordinat düzlemine dönüştürmektir. Yöntemin bundan sonraki kısmı takip davranışının gerçeklenme yöntemi ile oldukça benzerdir. Hedef nokta için (burada farklı olarak hedef nokta takip edilen etmenin konumu değil, bu konuma verilen mesafedeki noktadır) belirli bir süre sonundaki konumu belirlenir ve takip eden etmen tarafından bu konuma varma davranışı gerçeklenir.

2.3 Sürü Yönlendirme Davranışları

Sürü davranışları, canlandırma ortamında bulunan etmenlerin bir kısmının veya tamamının birtakım kurallar dahilinde gerçekleştirdiği bireysel davranışlar sonucunda oluşan, fakat genel perspektifte incelendiğinde bir bütünlük oluşturan davranışlardır. Her ne kadar karmaşık görünse de aslında bu türden davranışların oluşumunu sağlayan ve sürüdeki tüm bireylerin uyması gereken üç temel kural vardır (Reynolds, 1987) : ayırma, hizalama ve bağlılık. Bu üç kuralın sürüdeki her birey için işlemesi durumunda ortaya doğada da sıkça rastlayabileceğimiz türden (örneğin; sürü olarak uçan kuşlar, yiyecek arayan balık sürüleri vb.) sürü davranışları çıkabilmektedir.

Bir sürü davranışı için yönlendirme kuvvetlerini bulmadan önce, sürüyü oluşturan etmenler için komşuluk tanımını ele almak gerekir. İki boyutlu uzay için bir etmenin komşuluğu, bu etmenin kurallarının çalıştığı algı bölgesidir. Yani bir anlamda o etmenin algılama sınırlarını belirleyen bir bölgedir. Etmen merkez olmak üzere belirli bir yarıçapa sahip dairesel alan olarak düşünülebilir. Burada yarıçapın büyüklüğü, etmenin algı sınırı ile doğru orantılıdır. Çok büyük yarıçaplı bir algı bölgesinin olması durumunda etmen daha uzak bölgelerdeki diğer etmenleri sezebilecek; kısa yarıçaplı algı bölgelerinin olması durumunda ise etmenin algı bölgesi daha sınırlı olacaktır. Buna göre, herhangi bir etmenin algı bölgesi içine giren diğer etmenler, o etmenin komşuları olarak kabul edilir. Bu durum Şekil 2.11'de gösterilmektedir.



Şekil 2.11 Etmen komşuluğu

Örnek olarak verilen Şekil 2.11'de beyaz renkli etmenin komşuluğuna giren etmenler siyah renkli olarak gösterilmişlerdir. Gri renkli etmenler ise komşuluk çemberinin dışında kaldıklarından, beyaz etmenin komşuları olarak işaretlenemezler.

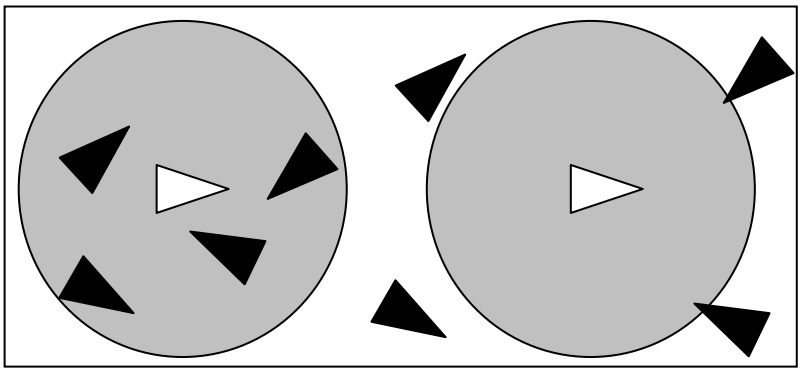
Buna göre, herhangi bir etmen için bir yönlendirme kuvveti belirlenmeden önce, o etmenin komşuları belirlenmeli ve bu komşular işareletlenerek, söz konusu etmenin hareketi için yapılacak olan tüm hesaplamalara dahil edilmelidir.

Komşuluk alanı, canlandırılacak olan nesnenin doğası göz önüne alınarak daha gerçekçi yapılabilir. Örneğin modellenecek olan bir kuş sürüsünün hareketleri ise, bu komşuluk alanı tam bir daire yerine 270º'yi tarayan bir daire parçası olabilir. Çünkü kuşlar tam arkalarında kalan bir bölgeyi sezemeyecekleri için bu bölgeye giren diğer kuşlardan haberdar

olamayacaklardır. Dolayısıyla hareketini bu arkada kalan kuşları dikkate almadan gerçekleştirecektir.

2.3.1 Ayırma kuralı

Bu kural, etmenin komşuluğu içinde bulunan etmenler ile olası bir çarpışmasını engelleyecek şekilde bir yönlendirme kuvveti üreten bir kuraldır. Yani bir nevi çarpışma denetim kuralıdır. Kural komşulukta bulunan tüm etmenlere de uygulandığında, bu etmenler birbirleri arasındaki uzaklığı maksimum yapacak şekilde birbirlerinden uzaklaşacaklardır. Bunun örneği Şekil 2.12'de görülebilir.

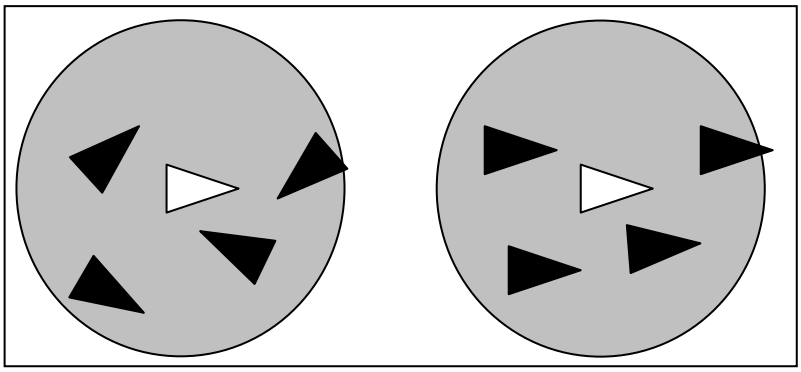


Şekil 2.12 Ayırma kuralı

Kuralın gerçeklenmesi oldukça basittir. Öncelikle etmenin komşuluğunda bulunan komşu etmenler belirlenir. Daha sonra bu işaretlenmiş etmenlerin ele alındığı bir döngüde, her komşu etmene olan konum vektörü bulunup normalize edilir, iki etmen arası uzaklığa bölünür ve ortaya çıkacak genel yönlendirme kuvvetine eklenir.

2.3.2 Hizalama kuralı

Hizalama kuralı, etmenin hareket doğrultusunu komşuluğu içindeki etmenlerin hareket doğrultularının ortalamasına uydurmaya çalışır. Bir anlamda sürüde yönelim birliğini sağlayan kuraldır. Şekil 2.13'de verilen örnekten de görülebileceği gibi bu kural komşuluktaki diğer etmenlere de uygulandığında bütün etmenler aynı yöne doğru hizalanırlar.



Şekil 2.13 Hizalama kuralı

Kural gerçeklenirken öncelikle etmenin komşuluk alanı içinde yer alan komşu etmenler belirlenir. Daha sonra bu etmenlerin ele alındığı bir döngüde, her etmenin yönelim vektörü bulunur ve bu yönelimlerin ortalaması alınır. Bu ortalama, istenen yönelim değeridir; dolayısıyla her etmene uygulanacak olan yönlendirme kuvvetini bulmak için bu ortalamadan etmenin yönelim vektörünü çıkarılması gerekmektedir.

Bir otoyolda hareket eden araçlar bu kuralın uygulandığı bir model olarak düşünülebilir. Araçlar yolda ilerleyebilmek için aynı hizada hareket ederler; üstelik birbirleriyle çarpışmamak için ayırma kuralını da işletip, her zaman aralarında belirli bir uzaklığın kalmasını da sağlarlar.

2.3.3 Bağlılık kuralı

Bağlılık kuralı, bir etmeni komşuluğunda bulunan etmenlerin ağırlık merkezini oluşturan noktaya doğru (dolayısıyla komşuluğun merkezine doğru) yönlendiren bir vektör üretir. Bu kural bir anlamda ayırma kuralı sonucunda aralarında belirli bir uzaklık oluşan sürü elemanlarının dağılmasını engelleyen ve birlikteliği sağlayan bir kural olarak düşünülebilir. Sürüden uzak düşen bir koyunun sürüye katılması için sürü merkezine doğru koşması, bu kuralın gerçek hayattaki yansımalarına bir örnek olarak verilebilir.

Kuralın gerçeklenmesi, hizalama kuralının gerçeklenme yöntemine çok benzer; fakat bu kez yönelimlerin ortalamasının alınması yerine komşu etmenlerin konumlarının ortalaması alınır. Bu ortalama konum, gidilmesi istenen konumdur; dolayısıyla kuralı işleten etmen bu noktaya doğru yönelim kuralını işletir.

Modelde yer alan bütün etmenler için bu üç kural birlikte işletildiğinde, ortaya göze hoş gelen bir sürü hareketi çıkacaktır. Her kural, bu hareketin önemli bir parçasını denetlemektedir.

Ayırma kuralı ile etmenlerin birbiriyle çarpışması önlenir; hizalama kuralı bireylerin aynı yönde hareket etmelerini sağlar; bağlılık kuralı ise bu etmenlerin birbirleriyle yakın hareket etmesini sağlar.

2.4 Yönlendirme Kuvvetlerinin Birleştirilmesi

Bir davranışsal canlandırmada genellikle etmenlerin birden fazla davranışı gerçeklemeleri istenir. Örneğin belirli bir noktaya yönlenen bir etmenin tek amacı sadece bu noktaya gidebilmek değildir; önüne çıkacak engeller ve duvarlar ile çarpışmamalı, yine önüne çıkabilecek yırtıcılardan saklanmalı veya kaçabilmelidir. Eğer bu etmen, sadece yönelme davranışını işletir ve diğer davranışları işletmezse sadece hedef noktaya doğru hareket eden ve çevresini algılamayan bir etmen; dolayısıyla gerçekçilikten uzak, yapay bir hareket modeli ortaya çıkar.

Uygulamada tanımlanmış her davranış modeli, *Behaviors* sınıfına ait farklı yöntemler olarak gerçeklenmişlerdir. Hareket halindeki her bir etmenin bu sınıftan türetilmiş bir örnek değişkeni bulunmaktadır ve her bir davranış, ilgili etmen için aktif/pasif konumuna alınabilmektedir. Örnek olarak, bir sürü ile birlikte gezinip engellerden ve duvardan kaçınan bir etmen modeli için eş zamanlı olarak uyarlanabilecek davranış modelleri şunlar olabilir :

- Ayırma
- Bağlılık
- Hizalama
- Rastgele Gezinme
- Engelden Kaçınma
- Duvardan Kaçınma

Buradaki ilk 3 kural, etmenin sürü hareketine uyumunu sağlamaktadır. Rastgele gezinme davranışı ile amaçsızca gezinme sağlanabilmekte; engelden kaçınma kuralı ile canlandırma ortamındaki engeller ile çarpışma önlenebilmekte ve duvardan kaçınma kuralı ile de etmenin algı bölgesi içinde ve hareket doğrultusunda belirebilecek bir duvar ile çarpışması önlenebilmektedir.

Vehicle sınıfına ilişkin *Update* yönteminde çağrılan *Calculate* yöntemi ile tüm aktif davranışlardan gelen kuvvetler birleştirilerek tek bir bileşke kuvvet elde edilmektedir. Bu bileşke kuvvetin hesaplanmasında 3 farklı yöntem kullanılabilir:

- Ağırlıklı toplam yöntemi
- Öncelikli toplam yöntemi
- Olasılıklı hesap yöntemi

2.4.1 Ağırlıklı toplam yöntemi

Bir etmene etki eden bileşke kuvvetin hesaplanmasındaki en basit yöntemdir. Her bir davranıştan gelen kuvvetler, o davranışlara ilişkin belirli ağırlık değerleriyle çarpılır, vektörel toplam alınır. Eğer bulunan toplam, etmene etki edebilecek maksimum kuvveti aşıyorsa, bu kuvvetten küçük kalacak şekilde bir kesme işlemi yapılır.

Bu yöntemin gerçeklenmesi her ne kadar kolay olsa da, çalışmada bazı problemleri içinde barındırır. Öncelikle her zaman çerçevesinde her bir davranışa ilişkin hesaplamalar yapılmaktadır. Bu durum, etmen sayısı arttığında sistem başarımında bazı sorunlara yol açabilir. Diğer bir sorun ise birbirine zıt kuvvetlerin bileşkesi alındığında ortaya çıkmaktadır. Bir etmenin, çok sayıda etmen tarafından duvara sıkıştırıldığı durum buna örnek olarak verilebilir. Burada duvardan sakınma yöntemi ile üretilen itme kuvveti, etmenlerden ayrılma yöntemiyle gelen itme kuvvetlerinin toplamından daha düşük kalabilir; bu ise etmenin duvarın diğer tarafına geçmesine neden olur. Duvardan sakınma davranışına ilişkin ağırlık değerini çok büyük yapmak buna bir çözüm gibi görünse de, bu kez etmen duvara yaklaştığında gerçek dışı davranışlar sergileyebilmektedir.

2.4.2 Öncelikli toplam yöntemi

Öncelikli toplam yönteminde her bir davranışa ilişkin bir öncelik değeri tanımlanır. Bazı davranışların, diğer davranışlar üzerinde öncelikleri bulunur. Örnek olarak çarpışma denetimleri (engelden veya duvardan kaçınma), öncelikli davranışlardandır. Aynı şekilde bir yırtıcıdan kaçma davranışının pek çok davranışa göre önceliği bulunmaktadır.

Öncelikli toplam yönteminde, her zaman çerçevesinde davranışlara ilişkin sonuç kuvvetler öncelik sıralamasına göre çalıştırılır ve her aşamada ara bileşke kuvvet hesaplanır. Herhangi bir anda bu kuvvet, etmene etki edebilecek maksimum kuvveti geçmiş ise, diğer davranışlar

göz ardı edilir ve sonuç bileşke kuvvet hesaplanır. Böylelikle başarım ve doğruluk arasında iyi bir denge kurulabilmektedir. Zira en öncelikli davranışlar kesinlikle işlenmekte, fakat maksimum kuvveti aşan, öncelik değeri düşük davranışlar işlenmeyebilmekte, bu ise başarım arttırımını sağlamaktadır.

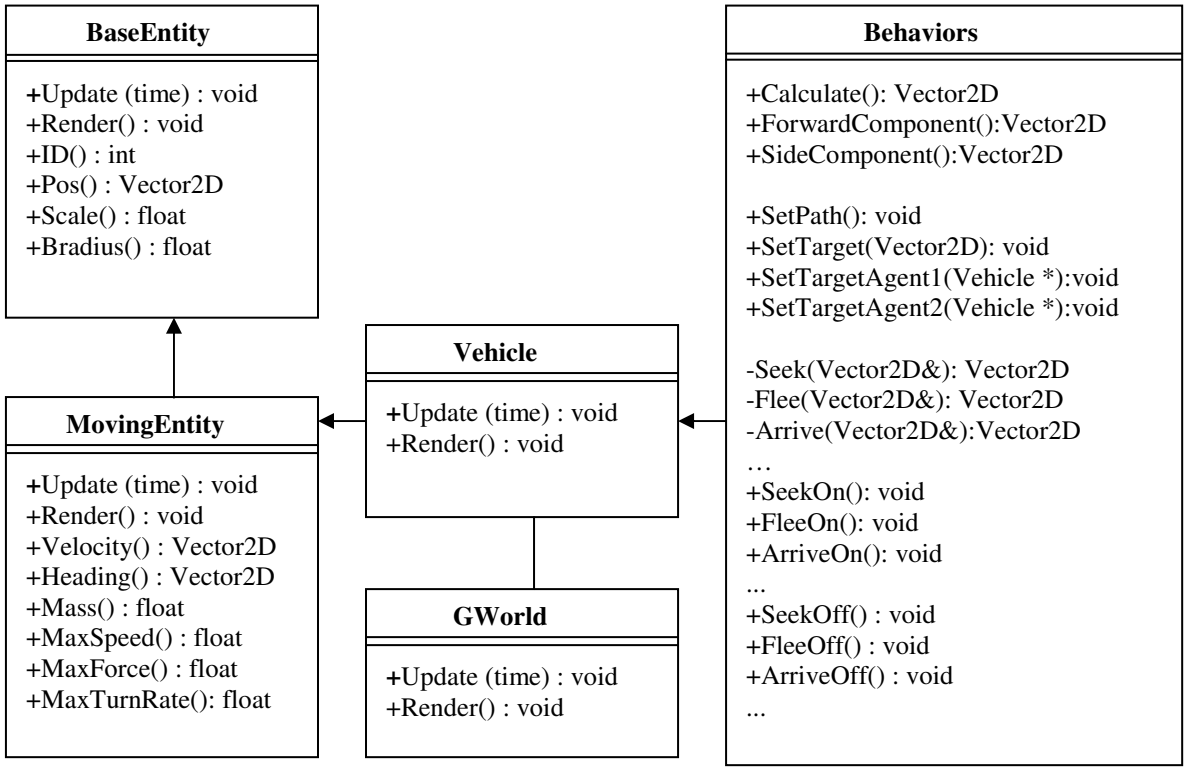
2.4.3 Olasılıklı hesap yöntemi

Bu yönteme göre, belirli bir zaman çerçevesinde etmene etki eden davranışlardan sadece bir tanesi hesaplanır ve buradan gelen kuvvet, sonuç kuvvet olarak işletilir. Bu hesap için her bir davranışın çalıştırılma olasılığı belirlenir; 0-1 arasında bir sayı olarak belirlenebilecek bu değer, bir çeşit davranış öncelik değeri gibi düşünülebilir. Yani çalıştırılma ihtimali yüksek olan davranışlara yüksek değerler verilmelidir. Olasılıklı hesap yöntemi ile en iyi başarım sağlanır fakat bu kez gerçekçilikten ödün verilmektedir.

Bu üç yöntem başarım ve gerçekçilik açısından karşılaştırıldığında, bu ikisi arasında en iyi dengeyi sağlayan yöntem öncelikli toplam yöntemidir. Ağırlıklı toplam yöntemi, bazı özel durumlar dışında en gerçekçi hareketleri sağlayan yöntemdir fakat çok fazla CPU dostu değildir; zira her zaman çerçevesinde her etmen için etki eden tüm davranış yöntemleri çalıştırılır ve bileşke kuvvetler hesaplanır. Olasılıklı hesap yöntemi ise, bu yöntemin tam tersine çok az işlem gücü harcar; fakat sonuçlar her zaman için gerçekçi olmayabilir. Öncelikli toplam yöntemi ise belirli öncelik değerlerine göre davranışları sıralamakta, her davranışı bu sıra ile işletip bileşke kuvvetin ara değerlerini hesaplamakta ve bu değer maksimum kuvveti geçtiği anda ise hesaplamayı bitirmektedir. Böylece başarım ve gerçekçilik arasında iyi bir denge kurulabilmektedir.

2.5 Temel Sınıf Hiyerarşisi

Benzetim sisteminde modellenmiş olan temel sınıflar arasındaki ilişki Şekil 2.14'de verilen UML diyagramında basit olarak gösterilmektedir.



Şekil 2.14 Buckland benzetim sistemi temel sınıf hiyerarşisi (Buckland, 2005)

3. ÖNERİLEN TAKIM CANLANDIRMA SİSTEMİ

Davranışsal canlandırma benzetim sistemlerinde, birbirleri ile etkileşim içinde olmadan tamamen bağımsız hareket eden etmenlerin olduğu ortamdan sadece "*etmenler topluluğu*" şeklinde bahsedebiliriz. Bu topluluğun etmenleri her ne kadar rastgele gezinme, engelden kaçınma, duvardan kaçınma gibi davranışları gerçekliyor olsalar da, ortada ortak bir amaç yoktur. Eğer bu ortak amaç (davranış) tanımlanırsa o zaman etmenler topluluğu artık "*sürü*" olarak adlandırılabilir.

Sürü davranışının modellenebilmesi için canlandırma ortamındaki tüm etmenlere üç temel sürü yönlendirme davranış kuralının (*ayırma*, *bağlılık*, *hizalama*) uygulanması gerekir. Sürünün gözlenen hareketi tüm etmenlerin yaptıkları hareketlerin bir sonucu olduğundan, sürüye genel bir hedef belirlemek için bu hedefin sürüdeki tüm etmenlere tanıtılması gerekmektedir. Hedefin gerçekleştirilmesi için uygulanması gereken davranışlar ne derece karmaşık olursa, sistemin genel başarımı o derece düşer. Bunun dışında sürünün önüne çıkabilecek çeşitli engeller ve bazı etmenlerin olası denetimsiz davranışları, tıkanma ve sürünün dağılması gibi sorunlara sebep olacak; sürünün tekrar bir araya gelebilmesi için dağılmış olan etmenlerin yeniden birbirlerinin algı bölgelerine girmeleri gerekecektir.

Yaptığımız çalışmada, sürü canlandırma sistemlerinin olumsuzluklarını giderme ve bu alanda yeni çalışmalara imkan tanıma amacıyla yeni bir davranışsal canlandırma sistemi olarak önder denetimli bir model sunulmaktadır. Bu noktada önerdiğimiz önder denetimli modelin önceki modellerden farkını ortaya koymak için var olan *sürü* kavramının yanı sıra ayrıca *takım* kavramı da önerilmektedir. *Sürü (flock)* kendi başına hareket eden birbirine yakın *etmenler* topluluğu iken, *takım (group)* kavramının bir önderi takip eden *bireyler* topluluğu olarak tanımlanması önerilmiştir. Önerdiğimiz yapıda sürünün elemanı olan *etmen* ile önderli bir takımın elemanı arasındaki farkı vurgulamak için birey (*individual*) sözcüğünün kullanılması uygun görülmüştür. Doğal olarak yazılım sisteminde bireyler de birer etmen (*agent*) olarak modellenmiştir.

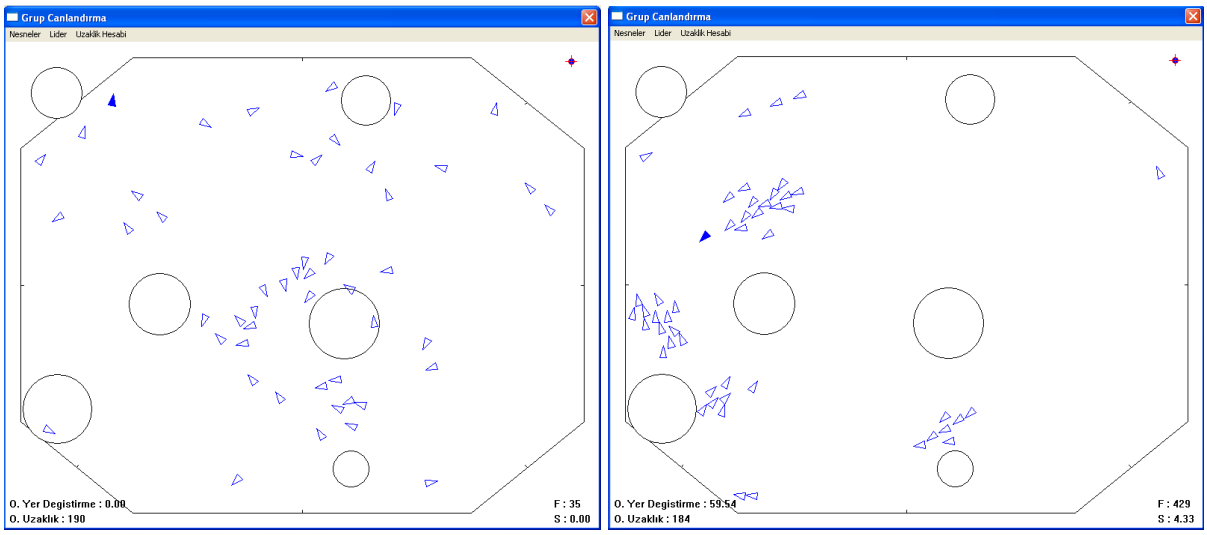
Önder modeli geliştirilme sürecinde, önder ve diğer etmenler tarafından kullanılmak amacıyla ilk defa Reynolds (1999) tarafından tanımlanan ve Buckland benzetim sisteminde olmayan birtakım bireysel yönlendirme davranışlarının da geliştirilmesi gerekmiştir. Bu davranışlar; Bölüm 5.1'de tanıtılmış olan önderi takip davranışı, sıralı izleme davranışı, duvar izleme davranışı ve saldırı davranışıdır.

Bu bölümde, önerdiğimiz takım canlandırma sisteminin temel birimleri olan takım ve önder kavramlarının modellenmesi açıklanmıştır. Takım modeli ile daha önce var olan sürü modelinin karşılaştırılabilmesi amacıyla takım başarım ölçütlerinin belirlenmesi ise Bölüm 3.5'de ele alınmıştır.

3.1 Canlandırmada Takım Modeli

Gerçek yaşamdaki takım kavramının bilgisayar canlandırmasındaki karşılığı, bir önder ve bu önderi izleyen bireyler tarafından oluşturulmuş topluluktur. Bu topluluktan uzakta bulunan diğer bireyler, algı bölgelerinin içine takım elemanlarından birinin girmesi durumunda takıma dahil olacak ve takım önderini izlemeye başlayacaktır. Dolayısıyla takım, bir önder denetiminde ortak hedefe varmaya çalışan bireyler topluluğu olarak modellenmiştir.

Bölüm 2.1'de açıklandığı gibi önerdiğimiz sistemde de önder ve bireyler, yazılım ortamında birer etmen olarak modellenmiştir. Ancak doğal olarak bu durumda hepsi özdeş etmen olmayacaktır.



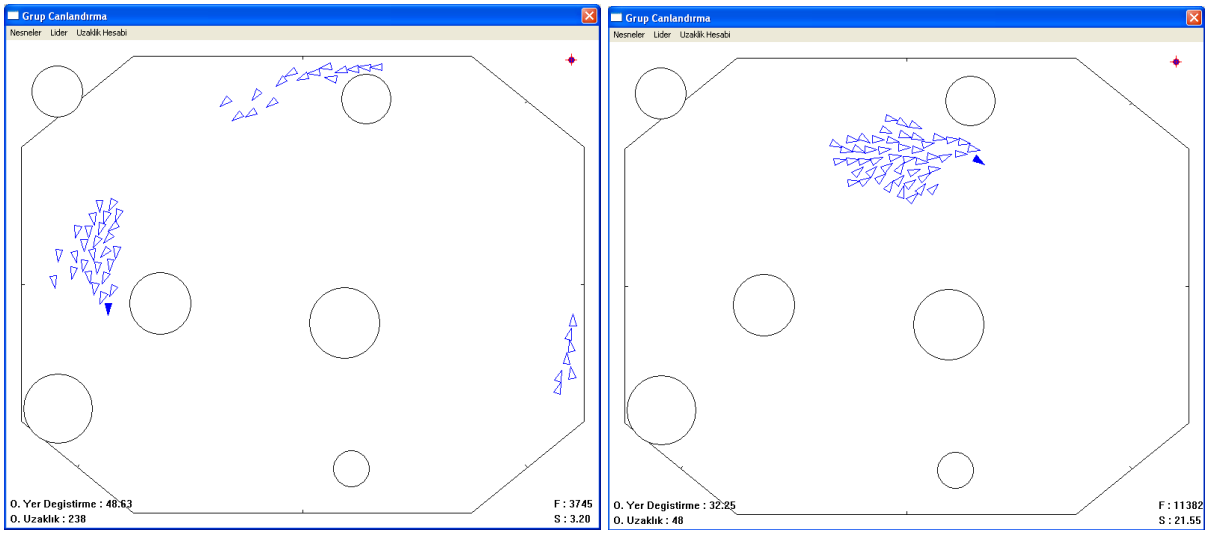
(a) (b)

Şekil 3.1 Canlandırmada takım oluşumları ve önder - I

Canlandırma sürecini açıklamak üzere örnek bir canlandırma ele alınmıştır. Şekil 3.1'de örnek canlandırmanın ilk anlarından alınmış ekran görüntüleri verilmektedir. Canlandırma başlangıcında (Şekil 3.1.a) koyu renkle gösterilmiş olan önder ve diğer tüm bireyler, rastgele konumlarda ve rastgele yönlere yönelmiş olarak yaratılmışlardır. Daire ile gösterilen nesneler ortamdaki engelleri, düz çizgiler ise duvarları temsil etmektedir. Senaryoda önder rastgele gezinme davranışı sergilemekte; diğer bireylere ise sürü davranışı ve algı bölgelerine girdiği

anda önderi izleme davranışı uygulanmıştır. Canlandırma ilerledikçe birbirleri ile karşılaşan bireyler sürü davranışı içinde gruplaşmaya başlamış; önderi algı mesafesi içinde gören bireyler ise önderi takip etmeye başlamışlardır. Bu durum Şekil 3.1.b'de görülebilmektedir. Burada dikkat edilmesi gereken en önemli nokta, her bir bireyin önderi izlerken kendine özgü hareketler sergilemesidir. Çünkü etmenler canlandırma süresince çok farklı yönlendirme kuvvetlerinin etkisi altında kalır ve bu kuvvetlerin bileşkesi olan etmen yönlendirme kuvveti her bir etmen için farklı olur. Örneğin önderi takip eden bazı bireyler birbirleri ile çarpışmamak için yön değiştirebilir; bazıları ise önlerine çıkabilecek bir engelin yanından geçerek önderi takibe devam edebilir. Sonuç olarak; önderi takip eden tüm bireyler için tek bir hesaplama yapılıp tüm etmenlere uygulanmamaktadır. Ortak hesaplama yöntemi her ne kadar sistem başarımının oldukça yüksek olması anlamına gelse de, ortaya çıkacak olan canlandırma gerçekçilikten oldukça uzak olacaktır.

Takım içindeki her bireyin takım hareketini şekillendiren üç temel davranışı ayrı ayrı çalıştırması yerine önderi takip etmesi ve sadece çarpışmadan kaçınma denetimi davranışını gerçeklemesi, sistem başarımını arttırıcı yönde bir etki yapacaktır. Öte yandan bir tıkanıklık durumunda bunun sezilip önder değişim algoritmalarının devreye girmesi ile de, canlandırmanın sürekliliği sağlanacaktır.



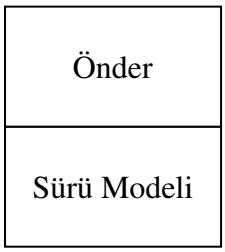
(a) (b)

Şekil 3.2 Canlandırmada takım oluşumları ve önder - II

Örnek canlandırmanın ilerleyen aşamaları Şekil 3.2'de yer alan iki ekran görüntüsünde verilmektedir. Şekil 3.2.a'da artık üç ana küme meydana gelmiştir. Resmin solunda önderi takip eden küme asıl olarak bir *takımdır*; diğer iki küme ise öndersiz kendi başlarına hareket ettiği için *sürü* olarak değerlendirilir. Bu sürülerdeki bireylere her ne kadar önderi takip etme

davranışı atanmış olsa da, önderin içinde yer aldığı takım bu iki farklı sürünün algı bölgesine henüz girmemiştir. Şekil 3.2.b ise kümelerin kaynaştığı bir ekran görüntüsüdür. Tüm bireyler artık önderi izleyen tek bir takımın içinde yer almaktadır. Takımın yönü ve hedefi önder tarafından belirlenmektedir.

Sonuç olarak önerdiğimiz sistemdeki takım modeli, Şekil 3.3'de görüldüğü gibi önder denetiminde hareket eden bir sürü modeli olarak gerçeklenmiştir.



Şekil 3.3 Takım Modeli

3.2 Önder Modeli

Gerçek yaşamda takım davranışı sergileyen canlılar bireysel farklılıklar gösteren hareketler yapmakla birlikte, çoğunlukla kendi aralarından seçtikleri veya kendilerine önder olarak atanmış bir bireyi izlerler. Önder, belirlenen hedefe ulaşmaya çalışan ve takımın genel davranışını yönlendirici özellikli bir bireydir. Gerçek yaşamda önder aynı zamanda takımındaki bireylerden sorumlu olabilmekte, onlardan aldığı bilgilere göre bu bireyleri yönlendirebilmektedir. Çok özel uygulamalarda bir senaryo karşılığında uygulanabilecek olan bu özellik, önder modelimizde kapsanmamıştır. Aslında önder olarak tanımlanan etmen, sürü modelindeki herhangi bir etmen özelliği taşımaktadır. Modelimizdeki ağırlık, önderi izleme görevi verilen takım bireyleri üzerindedir. Ancak ileri bir uygulama olarak, örneğin hiyerarşik önderlik gibi uygulamalarda önder sayısının birden fazla olması ve aralarında eşgüdüm olması gerektiğinden önder olarak tanımlanan etmenler doğal olarak herhangi bir sürü etmeni özelliğinde olmayacaklardır.

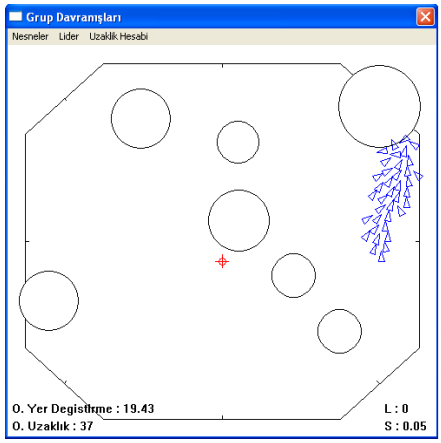
Önderlik farklı uygulamalarda el değiştirebilmektedir. Örneğin göçmen kuşlarda sürünün en önü olan önderlik yerinin sırayla diğer bireylerce paylaşıldığı bilinmektedir. Gezinmekte olan bir balık sürüsünün, bir yırtıcı tehdidi karşısında buna ilk tepki veren bireyleri izledikleri ve genel kaçışma hareketinin bu bireylerin önderlikleri tarafından yönlendirildiği bir başka örnektir. Özellikle tıkanma sorunun giderilmesi ve farklı uygulamalara imkan tanımak için

modelimizde “Önder Değişimi” konusu da kapsanmıştır. Önerdiğimiz sistemde bir takım için önder seçimi dört farklı şekilde yapılabilmektedir:

- Takımın ilk bireyi,
- Takımın merkezine en yakın olan birey,
- Takımın merkezine en uzak olan birey,
- Takım içinden rastgele seçilen bir birey.

Önderin takımın merkezine en yakın birey olması, takımın kolay dağılmamasını sağlar. Buna karşın tüm takım bireyleri öndere yönelecekleri için canlandırmada aşağıdaki paragrafta da açıklanmış olan tıkanma olasılığı artar. Takımın en uzağında bulunan birey önder olarak seçilirse, bir önce anlatılan durumun tersi oluşacaktır. Önder değişimi konusu Bölüm 4’te ele alınmıştır.

Sürü canlandırmalarında karşılaşılan temel sorun, canlandırma ortamında engellerin bulunması durumunda tıkanıklıkların oluşabilmesidir. Önderi takip eden bireyler, önderin önünün tıkanması durumunda önderi sıkıştırabilmekte ve takım canlandırmasındaki akıcılık kesilebilmektedir. Örnek olarak Şekil 3.4’de böyle bir durum görülmektedir. Her bir etmenin yakın çevresinde bulunan diğer etmenlerin hareketsiz kalması durumunda sistem ancak uzun zamanda çözülebilen bir tıkanmaya girebilmektedir. Bu tıkanıklığın çözümü ancak bir çeşit zincirleme hareket sonucunda olur. Tıkanıklık merkezine uzakta bulunan bir veya daha fazla etmenin rastgele açık bölgeye yönlenmeleri sonucu bu etmenlere yakın diğer etmenler sırasıyla sürü hareketine uyacak ve bu şekilde tıkanıklık çözülebilecektir. Tıkanıklık konusu Bölüm 4’te ele alınmıştır.



Şekil 3.4 Tıkanıklık durumu

Önerdiğimiz sistemdeki önder modeli, Şekil 3.5'te de görülebileceği gibi sadece belirlenmiş olan hedefe ulaşmaya çalışacak şekilde takımı yönlendiren bir birey olmayıp, aynı zamanda tıkanıklık durumlarında önder değişim algoritmalarını tetikleyerek önderliğin değişmesini, dolayısıyla tıkanıklığın çözülmesini sağlayan bir modeldir.

Şekil 3.5 Önder modeli

3.3. Takım Bireyi Modeli

Sürü canlandırma sistemlerinde sürünün oluşumunu sağlayan temel unsur, her sürü bireyinin bağlılık, hizalama ve ayırma davranışlarını çalıştırarak hesaplanan bir bileşke kuvvet yönünde hareket etmesidir. Sadece bu üç davranışın değil, bireye etki eden tüm davranışların bir bileşkesi olan bu kuvvet, bireyi sürüye uyumlu bir yönde ilerlettiği için bir çeşit *sözde önder* olarak da düşünülebilir. Aslında her sürü bireyinin sözde önderi, o bireyin kendi iç dinamiklerinin bir sonucu olarak bireyi yönlendiren kuvvettir.

Bir sürüyü sözde önderler denetiminde canlandırmak yerine sürüden seçilen bir bireyi önder olarak belirlemek ve diğer bireylerin bu önderi izlemesini sağlamak; sürünün artık bir önder denetiminde hareket eden bir *takım*, sürü bireylerinin de önderi izleyen *takım bireyleri* olması anlamına gelir. Takım bireyi modeli için ihtiyaç duyulan önderi izleme davranışının benzetim sistemindeki eksikliği, bu davranışın da gerçeklenip sisteme eklenmesini gerektirmiştir. Bu amaçla, izleyen paragraflarda tanıtılan iki farklı tipteki önder izleme davranışı gerçeklenip sisteme eklenmiştir.

Önderi izleme davranışı, canlandırma ortamında bulunan etmenlerin sistem tarafından belirlenen bir önder etmeni izleme davranışıdır. Yöntemin gerçeklenmesi, esas olarak Bölüm 2.2.12'de tanıtılan mesafeli takip yönteminin özel bir durumu olarak da incelenebilir. Önder dışındaki tüm bireyler, her canlandırma adımında önderin hemen arkasındaki sanal bir noktaya yönlenerek takip davranışını gerçekleştirmektedir. Böylelikle takımın genel davranış

modelini önder belirlerken diğer bireyler önderi takip ederek genel davranış modelini gerçekliyor görüntüsü verir.

Önder takibinin özel bir durumu olan *sıralı izleme davranışı*, takımdaki etmenlerin birbirleri arkasında sıralı olarak dizilmeleri şeklinde gözlemlenen bir davranış modelidir. Bir askeri birliğin tören düzeninde yürümesi, doğada bir fil sürüsünün ilerleyişi ve yavrularını peşine takmış bir anne ördeğin hareketleri bu davranışın sıkça rastlanan örnekleri arasındadır.

Önderi izleme davranışını gerçekleyen takım bireylerinin öndere yönelmeleri sonucunda önderin önünün tıkanıp hareketsiz kalma olasılığı ortaya çıkabilir. Bu durumda takım hareketsiz kalacak ve sistem tıkanmaya gidecektir. Tıkanıklığı önlemek yolunda gerçeklenebilecek bir yöntem, takım bireylerinin akıllı davranıp önderin önünü her zaman açık tutmalarıdır (Reynolds, 1999). Bu sebeple gerçeklenen ve takım bireylerine bir davranış modeli olarak atanabilen "önderin önünden kaçınma algoritması" Bölüm 4.4'te detaylı olarak açıklanmıştır.

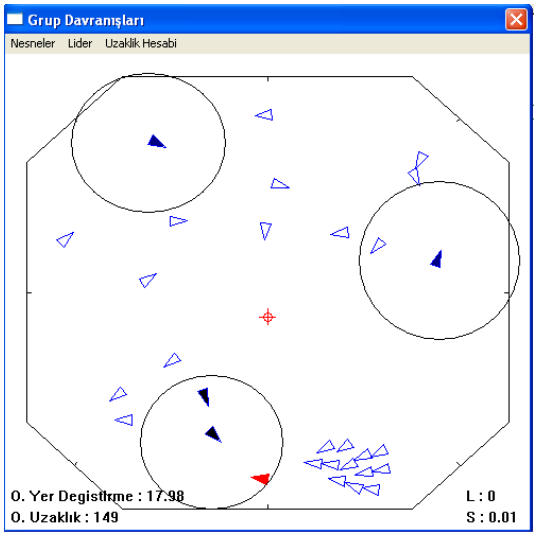
Sonuç olarak, önerdiğimiz sisteme ilişkin bir takım bireyi, sürünün içinde yer alan ve önderi takip eden bir birey şeklinde modellenmektedir (Şekil 3.6). Tüm takım bireylerinin önderi izlemesi, bir bakıma önderin davranış modellerinin bu bireylerce de dolaylı şekilde uygulanması anlamına gelmektedir.

Şekil 3.6 Takım Bireyi Modeli

3.4 Etmenlerin Algı Bölgeleri

Gerçek yaşamda takımı oluşturan her bir bireyin bir algı bölgesinin olduğu bilinmektedir. Her bir birey, algı bölgesi içindeki olaylardan haberdardır. Örneğin eğer takımın önderi bu alanın içindeyse birey önderi takip edecektir. Takım önderinin bu alanın dışında olması, bireyin tanımlanmış ise sürü oluşumuna imkan tanıyan davranışları sergilemesi anlamına gelir. Eğer sürü oluşum davranışları da tanımlanmamış ise, bu durumda takım bireyi tamamen bağımsız bir birey olarak rastgele gezinme, engelden kaçınma gibi olağan davranışları sergileyecektir.

Böyle bir durumda tek bir takımın oluşması uzun zaman alabilir. Çünkü her birey önderi sezdiği anda takıma dahil olacak; önderin uzağında kalan bireyler ise farklı davranışlar sergilemeye devam edeceklerdir.



Şekil 3.7 Etmenlerin algı bölgeleri

Örnek olarak Şekil 3.7'de verilen senaryoda bu durum açıkça görülmektedir. Bu senaryoda önder, görüntünün en altında koyu renk ile ifade edilmiş olan etmendir. Önderi algı bölgesi içinde sezmiş olan iki birey, öndere doğru yaklaşmakta ve takıma dahil olmak üzeredirler. Öte yandan önderden uzak olan bireyler rastgele gezinmektedir. Önder bu bireylerin algı bölgeleri içine girdikçe takım büyüyecektir. Araya herhangi bir engelin girmesi durumunda önderi görüş alanından kaybeden bireyler ise tekrar rastgele gezinme hareketine devam ederler.

Algı bölgesi kavramını başta önderi izleme davranışı olmak üzere diğer davranış modellerine eklemek için, öncelikle ilgili yöntemlerin başında birey etmeni ile önder etmeni arasındaki uzaklık vektörünün boyu bulunur. Eğer bulunan bu değer bireyin algı bölgesini belirleyen sanal dairenin yarıçapından büyük ise, önder bireyin algı alanı dışında kabul edilir. Bu durumda önderi izleme davranışı herhangi bir kuvvet üretmeyecek, dolayısıyla bireyin uygulayacağı bileşke davranış içinde herhangi bir etkisi olmayacaktır. Diğer yandan bireyden öndere olan uzaklık vektörünün boyu birey algı alanının yarıçapından küçük ise, birey önderi algılayabilecek ve bu durumda bireyden öndere doğru bir vektör üretilecektir.

3.5 Takım Başarım Ölçütleri

Önerilen önder denetimli takım canlandırma sistemi, yapılan bir uygulamanın değerlendirilmesine yönelik olarak takım başarım ölçütlerininin önerilmesini de

kapsamaktadır. Sistemde önderin varlığı, bize başarım ölçütü ortaya koyma konusunda fırsat tanımıştır. Burada önder ortak bir amacı temsil etmektedir. Başarı ölçütü, bu ortak amacın belirlenmesi ve ölçülmesine yönelik olmaktadır. Örneğin bir başarı ölçütü olarak canlandırmanın sürekliliği, herhangi bir tıkanıklık olmadan bireylerin birbirlerine yakın ve yüksek bir ortalama hızda hareket edebilmeleri şeklinde tanımlanabilir. Bu noktada, birbirlerine yakın ve aynı yönde hareket eden bireyler için başarımı belirleyen bir takım başarım ölçümü söz konusu olmalıdır. Temel olarak yüksek başarımı belirleyen ölçütler :

- Bireylerin birbirine olan ortalama uzaklıklarının az olması,
- Bireylerin aldıkları ortalama yolun fazla olması,
- Bireylerin yön bilgisi ile ortalama yön değeri arasındaki farkların az olması

şeklinde tanımlanır. Başarım puanı, ikinci ölçüt ile doğru orantılı, birinci ve üçüncü ölçüt ile ters orantılıdır. Yani bireylerin aldıkları ortalama yolun yüksek olması, bir tıkanıklık olmadığı anlamına gelir. Öte yandan bireylerin birbirlerine olan ortalama uzaklıklarının yüksek olması, takımın dağınıklığını belirleyen bir ölçüttür. Belli bir t anında n tane birey, yüksek hızlarda hareket ediyor ve birbirlerine yakın bir konumda bulunuyor olabilir. Fakat bu bireylerin yönlerinin farklı olması, bu birlikteliğin rastlantısal olduğunu belirler. Dolayısıyla yüksek bir takım başarımından bahsedebilmemiz için genel olarak bireylerin aynı yöne doğru hareket ediyor olmaları gerekir.

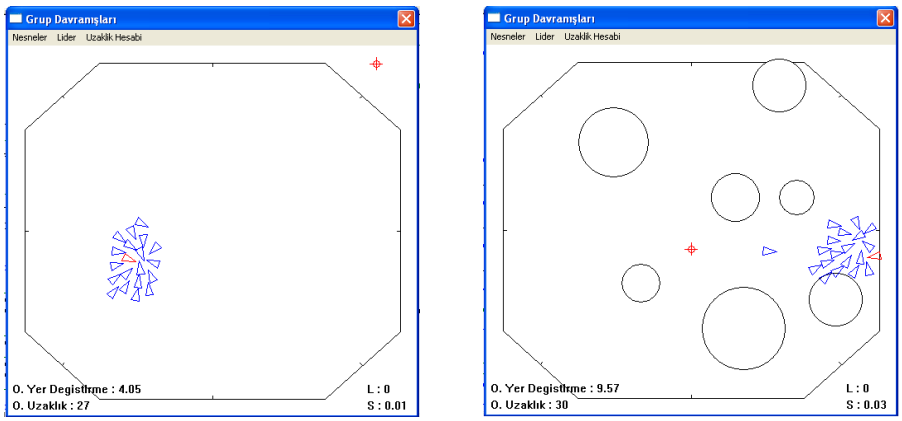
4. TIKANMALARIN ÇÖZÜLMESİ ve ÖNDER DEĞİŞİM ALGORİTMALARI

Takımı oluşturan her bir etmenin, sistemin belirlediği bir önderi takip etmesi ve takımın genel hareketini bu önderin belirlemesi, sistemde oluşabilecek tıkanma problemlerini çözebilir ve sistem başarımını olumlu yönde iyileştirebilir. Bu durumda takım önderi, diğer bireylerden farklı olarak ulaşılacak hedefi bilen (örneğin yol takip etme, yiyecek alanı gibi belirli bir noktaya ulaşma vb.) ve bu hedef doğrultusunda ilerleyen bir etmendir. Takımın belirli bir hedefi olmadığı durumlarda rastgele gezinme hareketini sergileyecek olan yine takım önderidir. Bir sürüyü oluşturan her bir bireyin fazla işlem içeren sürü içinde rastgele gezinme davranışını göstermesi yerine takım bireylerinin önderi izlemesi, canlandırmada dolaylı olarak yine benzer görüntüyü verecektir.

Sürü canlandırmasında bir tıkanmanın oluşması durumunda tıkanmanın kendiliğinden çözülmesi ya uzun zaman alacak, ya da hiç mümkün olmayacaktır. Önerdiğimiz önder denetimli takım canlandırma sisteminde bu olumsuzlukları gideren özgün yöntemler gerçeklenmiştir.

4.1 Takım Canlandırmasında Tıkanmanın Sezilmesi

Değişmez özellikli bir önder izleme davranışı, önderin hiçbir zaman değişmemesinden dolayı *körü körüne önderlik* olarak tanımlanabilir. Takım bireyleri tek hedef olarak öndere yönelir ve bazı durumlarda önder etrafı diğer takım bireyleri ile sarılmış ve hareket edemeyecek duruma gelebilir. Diğer bir senaryo ise önderin bir engel ve duvar karşısında geri dönmek veya yönünü değiştirmek istemesi ve diğer takım bireyleri ile karşılaşıp tıkanıklık yaşanmasıdır. Bu iki durum Şekil 4.1'de gösterilmiştir.



Şekil 4.1 Hareket edemeyen önderler

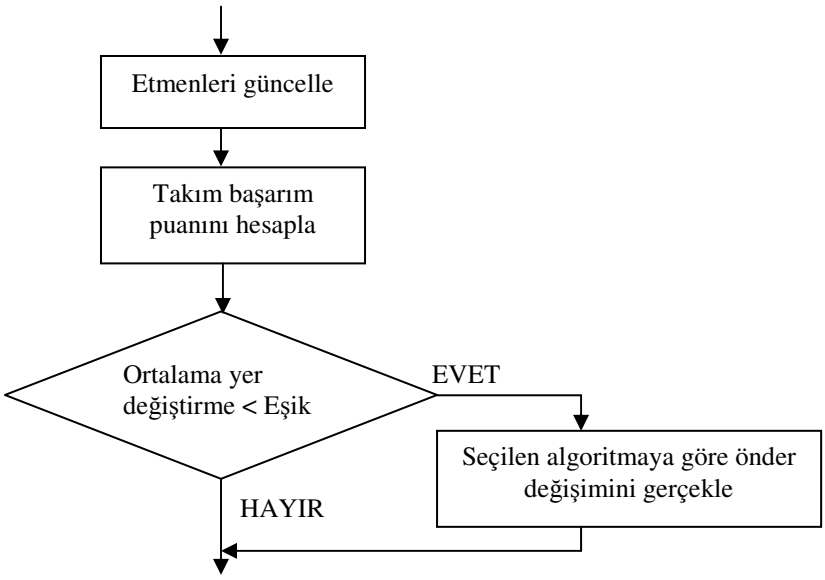
Önder denetimli takım canlandırmasının akıcılığını bozan herhangi bir tıkanmanın oluşması durumunda öncelikle bu tıkanmanın sezilmesi, daha sonra ise akışı sağlayacak çözümlerin devreye girmesi gerekmektedir. Tıkanıklık sınaması belirli aralıklarla çerçeve güncelleme çevriminde yapılır. Tıkanıklığın oluşması durumunda etmenler hareketsiz kalacak veya çok az hareket edeceklerdir. O halde tıkanıklık oluşumunu sezmek için yer değişikliklerinin ölçülmesi gerekir. Yaptığımız çalışmada bu değerlerin bir ölçüt oluşturabilmesi için "ortalama yer değiştirme parametresi" gerçeklenmiştir. Tıkanıklık durumu, ortalama yer değiştirme parametresinde ciddi bir düşüşe neden olur. Dolayısıyla belirli aralıklarla bu parametrenin bir eşik değerinin altında kalıp kalmadığı sınanarak sistemde olası bir tıkanıklığın varlığı sezilebilir. Sınanacak eşik değeri uygun bir değerde seçilmelidir. Bu değerin çok yüksek seçilmesi sistemin devamlı önder değişim algoritmasını tetiklemesine; çok düşük seçilmesi ise tıkanıklığın uzun süre devam etmesine ve önderin hiç değişmemesi gibi bir olasılığın ortaya çıkmasına sebep olur.

Bir tıkanıklık sezildiğinde önder değişimi olarak iki farklı çözüm yöntemi önerilmektedir:

- Rastgele yeni önder seçimi
- Var olan öndere en uzaktaki bireyin önder olması

Ancak yukarıdaki her iki yöntemde de önder değişmektedir. Önderi değiştirmeyen canlandırma uygulamalarında bu yöntemler kullanışsız kalmaktadır. Bu nedenle diğer bir önerdiğimiz yöntem ise tıkanıklığın önlenmesidir. Burada tıkanmanın hiç oluşmamasını sağlayan bir kural olan "önderin önünün hiçbir zaman kapatılmaması" davranışı tüm bireyler tarafından uygulanmalıdır.

Çerçeve güncelleme çevriminde tıkanıklık sınamasına ilişkin akış diyagramı, Şekil 4.2'de verilmiştir. Önder değişiminin gerçekleştiği çerçevenin hemen ardından gelen çerçevelerde ortalama yer değiştirme parametresi hızlı bir yükseliş göstermez. Zira yeni önder var olan tıkanıklığı anında çözemeyecektir. Bunun sonucunda her çerçevede ortalama yer değiştirme eşik değerin altında görünür ve devamlı yeni önder devreye girer. Bunu engellemek için her önder değişiminin ardından belirli bir süre boyunca önder değişim algoritmalarının çalıştırılmaması bir çözüm olarak sunulmaktadır (Kelly, 1997). Başka bir deyişle; her yeni öndere tıkanıklığı çözebilmek için belirli bir süre (uygulamamızda 100 çerçeve) şans tanınmıştır. Eğer bu sürenin sonunda da tıkanıklık çözülememişse, yeni bir önder seçimi için önder değişim algoritması tekrar devreye girer.



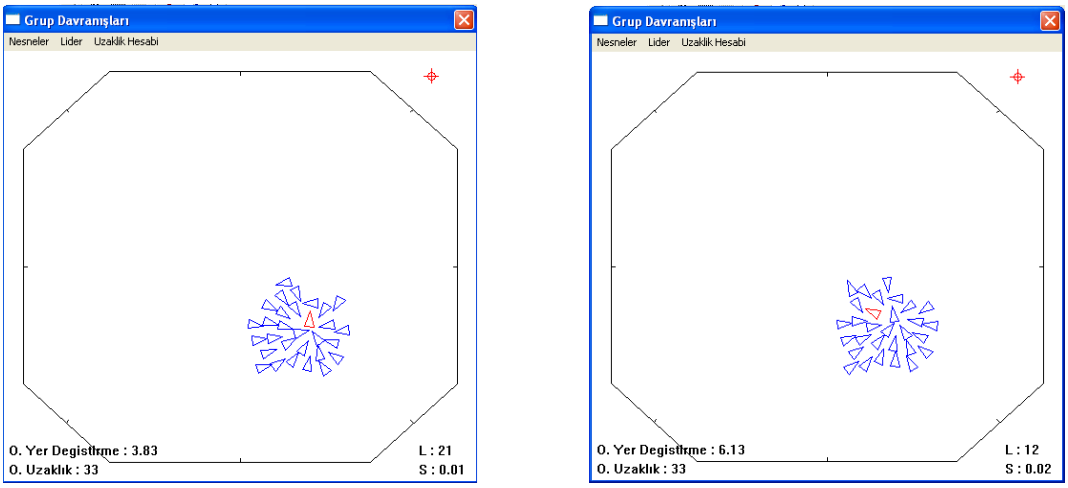
Şekil 4.2 Tıkanıklık sınaması

Aşağıdaki bölümlerde önder değişim algoritmaları ve tıkanıklığın oluşmamasını sağlayan önderin önünden kaçınma algoritması ayrıntılı olarak açıklanmıştır. Bunlara ilişkin sınamalar, hesaplanan sayısal değerler, tablolar ve grafikler Bölüm 6'da verilmiştir.

4.2 Rastgele Önder Seçim Algoritması

Tıkanıklık sezildiğinde sistem başarımını etkilemeden yapılabilecek en hızlı önder değişimi, yeni önderin rastgele seçilmesidir. Rastgele önder seçimi, uygulamada rastgele bir tamsayı üretmek şeklinde gerçeklenmiştir. Bu tamsayı, etmenlerin tutulduğu vektördeki yeni etmenin kimliğini belirlemede kullanılır. Yeni bireye önderlik verildikten sonra, bu önder takımın genel davranışını üstlenir. Eski önderi izleme davranışı yeni önderden kaldırılır, ayrıca yeni önderin hız vektörü, dolayısıyla yön bilgisi kendisinin eski yön bilgisinin tersi olacak şekilde düzenlenir. Bunun sebebi yeni önderi tıkanıklığın ters yönüne doğru hareketlendirmektir. Bu işlemlerden sonra takımın geri kalan üyelerine yeni önderi izleme talimatı verilir ve yeni öndere tanınan süre sıfırlanır. Eski önder ise takımın sıradan bir üyesi olarak düzenlenir.

Rastgele önder değişiminin olumlu yanı, işlemci gücü harcamadan hızlı bir şekilde değişimin gerçeklenmesidir. Fakat bu değişimin sonuçları her zaman tıkanıklığın hızlı çözülmesi yönünde olumlu sonuç vermeyebilir. Bu senaryoya bir örnek Şekil 4.3'de gösterilmiştir.



Şekil 4.3 Rastgele önder değişimi

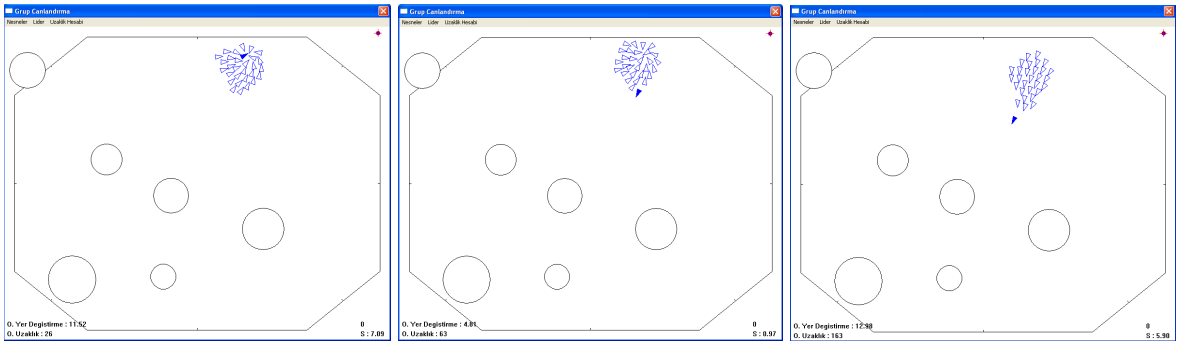
Şekil 4.3'ün sol bölümündeki canlandırma karesinde bir tıkanıklık görülmektedir. Ortalama yer değiştirme parametresi 3,83 olarak hesaplanmış ve bu değer tanımlanmış olan eşik değerinin (bu uygulamada 10 benek) altında olduğundan rastgele önder değişim algoritması çalışmıştır. Sol karede farklı renkte görülen eski önderin numarası 21'dir ve rastgele değişim sonucunda ise yeni önderin numarası 12 olmuştur. Bu yeni önder tıkanmış takımın içinde olduğundan hesaplanan ortalama yer değiştirme değeri (6,13) eşik değerinin üzerine çıkamamış ve tekrar bir önder değişimi gerekmiştir. Buradaki temel problem, rastgele olarak yapılan önder değişiminin ardından belirlenen yeni önderin, her zaman için tıkanıklığın dışındaki etmenlerden biri olamayışıdır. Eğer rastgele üretilen önder takımın dışındaki bir etmen olabilseydi, bu bireyin yönü değişecek ve tıkanıklık kısa sürede çözülebilecekti.

4.3 Uzak Bireyin Önder Olarak Seçilme Algoritması

Önder denetimli takım hareketlerinde oluşabilecek bir tıkanıklığın çözülmesi için, yeni önderin tıkanmış olan öndere en uzak mesafede bulunan etmenlerden biri olacak şekilde seçilmesi bir diğer çözüm yöntemidir. Böylece seçilen yeni önderin yönü eski yönüne ters yönde değiştirilerek önderin önünün açık olması sağlanır. Bu durumda, çerçeve güncelleme döngüsünde tıkanıklık sezildikten sonra o anki öndere en uzak konumda bulunan birey bulunur.

Bu çözümün rastgele önder değişimi çözümüne göre uygulama açısından daha iyi sonuçlar verdiği görülmüştür. Algoritmanın olumsuz yönü, her önder değişiminde var olan tüm bireylerin konumlarının taranması, dolayısıyla işlem yükünün rastgele önder değişim yöntemine göre daha fazla olmasıdır.

Şekil 4.4'te örnek bir canlandırmanın 3 farklı anına ilişkin sırasıyla tıkanıklık durumu, en uzaktaki bireyin yeni önder olarak seçilmesi ve tıkanıklığın çözülmesi aşamaları görülmektedir. İlk resimde çevresi takım bireyleri tarafından sarılan koyu renkli önderin hareket alanı kalmamış ve takımın ortalama yer değiştirmesi belli bir eşik değerin altına düşmüştür. Böylelikle önder değişim algoritması devreye girmiş ve ortadaki resimde görülebileceği gibi öndere en uzaktaki birey, yeni önder olarak seçilmiştir. Eski önder, önderlik özelliğini kaybettiği için artık sıradan bir takım bireyi olarak yeni önderi izleme davranışını sergilemeye başlamıştır. Son resimde ise önü açılan yeni önderin hareketlenmesi ve takım bireylerinin de bu öndere yönelmeleri sonucu tıkanıklık durumunun çözüldüğü görülebilmektedir.



Şekil 4.4 Uzak bireyin önder olarak seçilmesi

4.4 Önderin Önünden Kaçınma Algoritması

Olası tıkanmaların önlenmesi için önerdiğimiz bir yöntem, bireylerin önderlerinin önünü hiçbir zaman kapatmama davranışını da yürütmesidir. Bu davranışın gerçeklenmesi, tıkanmaların oluşmasını önleyecek ve dolayısıyla herhangi bir önder değişimi olasılığını da ortadan kaldıracaktır.

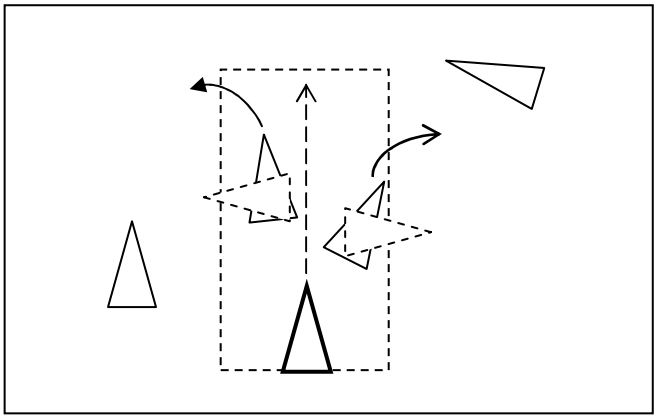
Önderin önünden kaçınma konusunda ilk düşünceler Reynolds'un (1999) önder izleme davranışı içinde gerçeklenmesine rağmen bu konu üzerinde başka bir gelişme göze çarpmamıştır. Önder izleme davranışı modelinden bağımsız olarak geliştirdiğimiz ve iki farklı yaklaşımda önerdiğimiz bu davranış modeli, davranışın gerçekleşme görüntüsünü belirleyen kullanıcı tanımlı parametreler sayesinde farklı uygulamalarda farklı şekillerde kullanılabilmektedir. Önderin önünden kaçınma davranışını önder izleme davranışı içinden ayırmamızın bir başka nedeni de, canlandırma ortamında bulunan ve herhangi bir önderi izlemeyen diğer bireylerin bu davranışı uygulayabilmesine ve ortamdaki önderlerin önlerinden kaçınabilmelerine olanak sağlamaktır.

Önderin ön bölgesi, Şekil 4.5'te görüldüğü gibi önder etmenin hareket yönünde oluşturulan sanal bir dikdörtgen alan (Reynolds, 1999) veya önder etmen merkez olmak üzere bu etmenin hareket doğrultusunda belirli bir açıyla taranmış olan bir daire dilimi (Şekil 4.6) olarak tanımlanmaktadır.

Önderin önünden kaçınma; tıpkı çarpışmadan kaçınma, bir hedefe yönelme veya yırtıcıdan kaçınma gibi bireysel yönlendirme davranışlarından biri olarak tanımlanmıştır. Canlandırmanın tasarımında bireylere bir davranış modeli olarak atanabilir. Bu durumda bireyler, önderin önünde belirlenmiş olan bölgeye girdikleri anda alandan itici yönde kaçınmaya çalışacaklardır.

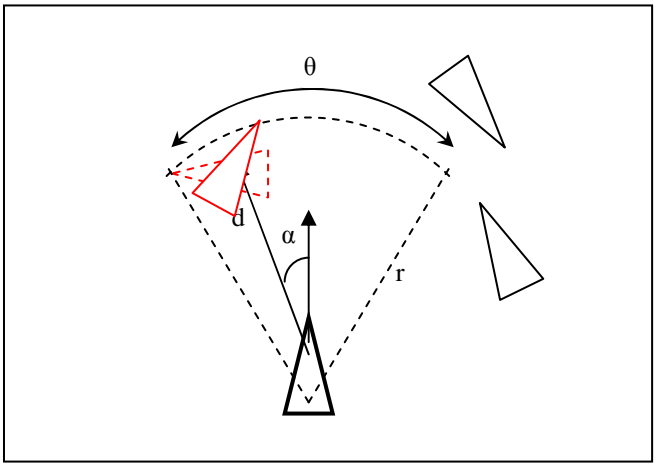
Şekil 4.5 ve Şekil 4.6'da dikdörtgen ve daire dilimi yaklaşımlarına örnek uygulamalar gösterilmiş ve ilgili paragraflarda açıklamalar yapılmıştır. Şekillerde yer alan yön değiştirmiş bireyin yeni konumu ideal bir durumu temsil etmektedir. Aslında bireyin yönü, bireye etki eden tüm kuvvetlerin bileşkesi olarak hesaplanmaktadır.

Şekil 4.5'de verilen dikdörtgen yaklaşımında, önderin önü sanal bir dikdörtgen alan olarak belirlenmiştir. Bu dikdörtgenin uzunluğu, önderin hızı ile orantılı olacak şekilde canlandırma sırasında değişir. Önderin önünden kaçınma davranışını sergileyen bireyler herhangi bir anda bu alan içine girdiklerinde, önderin solunda veya sağında olmalarına bağlı olarak en kısa yoldan bu alanı terkedecek bir vektörel kuvvetin etkisine girerler. Bu itici kuvvet, önderin yön vektörüne dik bir vektör olarak seçilir.



Şekil 4.5 Önderin önünden kaçınma (dikdörtgen)

Şekil 4.6'da ise önderin önü, bir daire dilimi olarak düşünülmüştür. Önderin önündeki alanı belirleyen 2 parametre, r ve θ parametreleridir. Burada r daire diliminin yarıçapını, θ ise bu dilimin taradığı açıyı temsil eder.



Şekil 4.6 Önderin önünden kaçınma (daire dilimi)

Herhangi bir bireyin bu alan içinde olup olmadığının denetimi önderin bu bireye olan uzaklığı (d) ve önderin hareket yönü ile bu bireyin konumu arasındaki açı (α) parametreleri ile belirlenebilir.

Buna göre;

- $d < r$ ve
- $|\alpha| < \theta/2$

koşullarının her ikisi de sağlandığı taktirde bu bireyin önderin önünde olduğu anlaşılmış olur. Bu durumda birey kendisine en yakın kenara doğru ve önderin yön vektörüne dik açı yapacak yönde hareketlenerek bu alanı terkeder.

5. TAKIM CANLANDIRMA SİSTEMİNİN GERÇEKLENMESİ

Bu bölümde, önerdiğimiz önder denetimli takım canlandırma sistemine ilişkin gerçeklemeler detaylı olarak ele alınmıştır. İlk olarak, literatürde tanımlanmış olan fakat Buckland benzetim sisteminde bulunmayan ve önerdiğimiz önder ve takım bireyi modelleri uyarınca yürütüldüğü için sisteme eklenmiş olan etmen yönlendirme davranışları incelenmiştir. Daha sonra bölüm 3.5'te sözü edilen takım başarım ölçütlerine ilişkin gerçeklemeler ele alınmıştır. Son olarak ise önerdiğimiz sisteme ilişkin yazılım tarafındaki gerçeklemeler detaylı bir biçimde anlatılmıştır.

5.1 Gerçeklenen Yönlendirme Davranışları

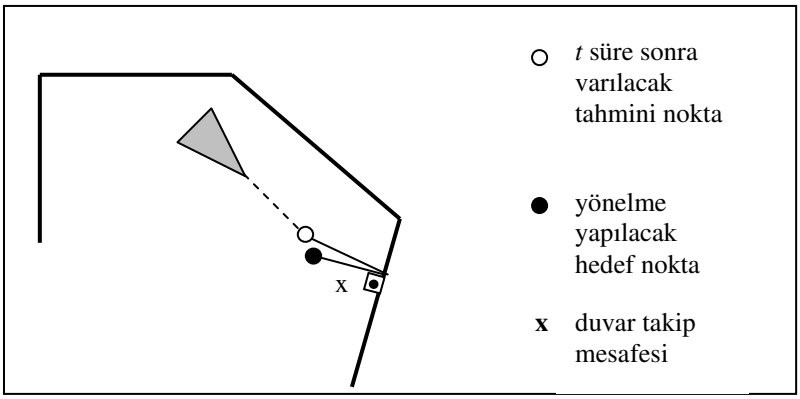
Önerilen önder denetimli takım canlandırma sistemindeki etmenler tarafından kullanılmak üzere ilk defa Reynolds (1999) tarafından tanımlanmış, ancak Buckland benzetim sisteminde bulunmayan bir dizi yönlendirme davranışı da gerçeklenmiş ve sisteme eklenmiştir. Aşağıda listelenen bu davranışlar, 3. bölümde önerdiğimiz önder ve takım bireyi modelleri uyarınca bireyler tarafından yürütülebilecek davranışlardır. Ayrıca 4. bölümde ele aldığımız takım canlandırmalarında tıkanıklık durumlarının bir çözüm yöntemi olarak önerdiğimiz önderin önünden kaçınma algoritması, önderi takip eden etmenler tarafından uygulanan bir davranış olarak geliştirildiği için bu listeye eklenmiştir. İzleyen alt bölümlerde eklenmiş olan bu davranış modelleri açıklanmıştır. Eklenen davranışlar şunlardır:

- Duvar izleme davranışı (*Wall Following*)
- Önder takip davranışı (*Leader Following*)
- Sıralı takip davranışı (*Single File Leader Following*)
- Saldırı davranışı (*Attacking*)
- Takım arama davranışı (*Looking for a Group*)
- Önderin önünden kaçınma davranışı (*Leader Obstruction Avoidance*)

5.1.1 Duvar izleme davranışı

Duvar izleme davranışı, bir etmenin canlandırma ortamında bulunan bir duvar boyunca, duvar ile arasında sabit bir uzaklığı koruyarak ilerleme davranışıdır. Bir labirentten çıkmaya

çalışırken veya bir bölgenin kapalı bir bölge olup olmadığının denetlenmesinde uygulanabilmektedir.



Şekil 5.1 Duvar izleme davranışı

Şekil 5.1'de görülmekte olan duvar izleme davranışının gerçeklenme yönteminde ilk olarak, etmenin kısa bir *t* süresi sonunda varacağı düşünülen noktanın koordinatları bulunur. Daha sonra, izlenecek olan duvarın bulunan bu noktaya en yakın olan noktasından bir *x* uzaklığında çıkılan dikme noktasına etmen yönelme hareketini gerçekleştirir. Böylelikle her adımda etmen, duvara paralel olarak belirlenmiş olan *x* uzaklığını koruyarak ilerlemiş olur.

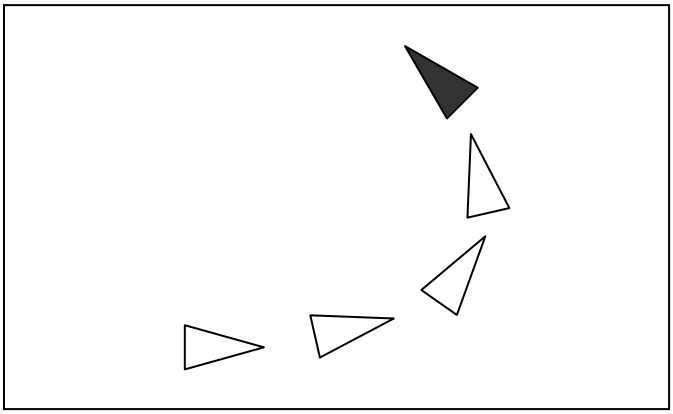
5.1.2 Önder takip davranışı

Canlandırma ortamında bulunan etmenlerin, belirlenen diğer bir etmeni - çoğu durumda önder etmeni - izleme davranışıdır. Yöntemin gerçeklenmesi, esas olarak Bölüm 2.2.12'de tanıtılan mesafeli takip yönteminin özel bir durumu olarak da incelenebilir. Önder dışındaki tüm etmenler, her canlandırma adımında önderin hemen arkasındaki sanal bir noktaya yönlenerek takip davranışını gerçekleştirmektedir. Böylelikle takımın genel davranış modelini önder belirlerken diğer etmenler önderi takip ederek genel davranış modelini gerçekliyor görüntüsü verir.

5.1.3 Sıralı takip davranışı

Önder takibinin özel bir durumu olan sıralı takip davranışı, takımdaki etmenlerin birbirleri arkasında sıralı olarak dizilmeleri şeklinde gözlemlenen bir davranış modelidir. Bir askeri birliğin tören düzeninde yürümesi, doğada bir fil sürüsünün ilerleyişi ve yavrularını peşine takmış bir anne ördeğin hareketleri bu davranışın sıkça rastlanan örnekleri arasındadır.

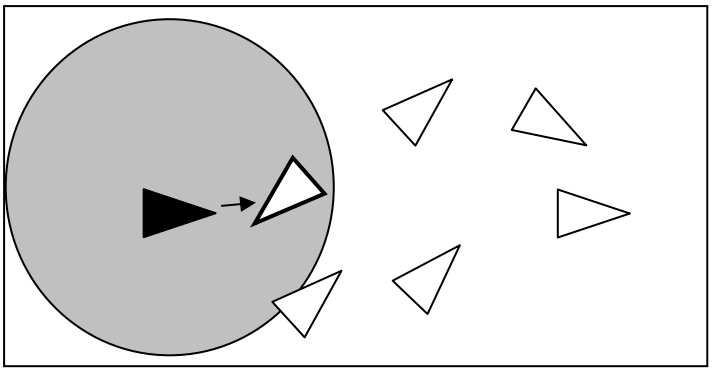
Şekil 5.2'de sıralı takip davranışı görülmektedir. Koyu renkli etmen önder olarak tanımlanmıştır ve diğer etmenler birbirleri ardına dizilerek önderi takip etmektedir.



Şekil 5.2 Sıralı takip davranışı

5.1.4 Saldırı davranışı

Canlandırma ortamında saldırgan nitelikli etmenlerin bulunması durumunda, bu tip etmenlerin uygulayabileceği bir davranış çeşididir. Yırtıcı etmenin kendi algı bölgesi içinde bulunan en yakın etmene yönlenmesi sonucunu doğurur. Gerekli şartlar sağlandığında ve yırtıcı ile av arasındaki uzaklık çok küçük bir değere indiğinde, av olan etmen canlandırma ortamından silinir. Örnek olarak Şekil 5.3'te verilen örnekte siyah renk ile gösterilen yırtıcı etmen, kendi algı bölgesindeki en yakın etmeni av olarak belirlemiş ve bu etmene doğru yönelme davranışını sergilemektedir.



Şekil 5.3 Saldırı davranışı

5.1.5 Takım arama davranışı

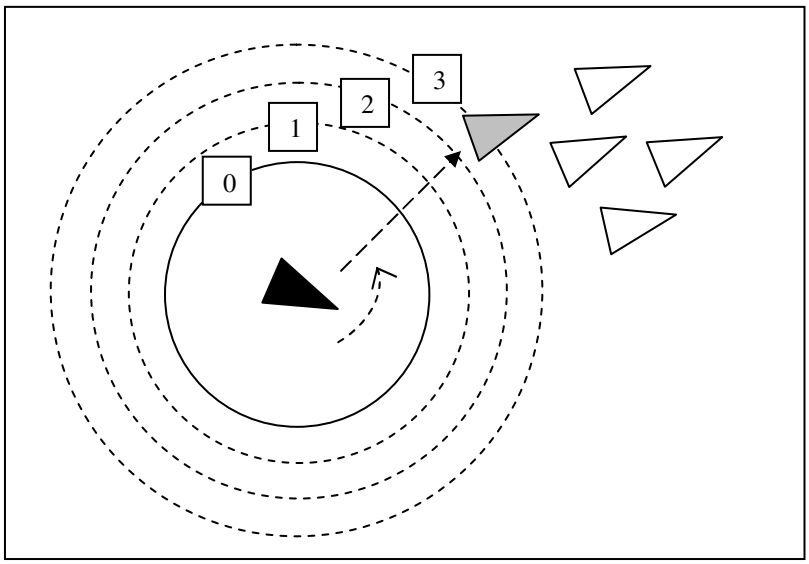
Takım canlandırmalarında bireylerin bir takımda bağımsız olarak tek başına gezinmeleri, karşılaşılabilecek durumlar arasında yer almaktadır. Bu durum, canlandırmanın ilk adımında bireylerin konumlarının rastgele belirlenmesinin bir sonucu olarak oluşabilir; veya bireyin dış

etkenler sebebiyle bir takımdan ayrılması ve takımın bu bireyin algı bölgesinin dışında kalmasının bir sonucu olarak da ortaya çıkabilir. Bu yalnız etmen her ne kadar canlandırmanın ilerleyen adımlarında diğer bir takım bireyini algı bölgesi içinde sezip takıma dahil olacak ise de, arada geçen süre tamamen canlandırmanın akışına bağlı olacaktır.

Önerdiğimiz takım arama davranışı, bir bireyin tek başına kaldığı durumlarda bu bireyin algı bölgesini kademe kademe genişleterek bir takıma dahil olabilme sürecini hızlandırmaktadır. Bu durum aslında doğada tek başına kalan bir canlının daha dikkatli davranmasının bir modellemesi olarak da düşünülebilir. Çünkü genellikle bağlı olduğu sürüden kopan bir hayvan, tekrar o sürüye ulaşabilmek için fazladan çaba sarf etmektedir.

Şekil 5.4'de takım arama davranışının çalışması aşama aşama görülmektedir. Koyu renkle gösterilen bireyin algı bölgesi içinde ilk durumda (0 durumu) herhangi diğer bir birey olmadığı için izleyen canlandırma adımlarında bu yalnız bireyin algı bölgesi bir miktar (uygulamada standart değerin %10'u ölçüsünde) genişletilmektedir. 3 adım sonunda etmen, algı bölgesi içinde bir takım bireyini sezmiş ve hareket yönünü bu bireye doğru değiştirmiştir. Bu aşamadan sonra bireyin algı bölgesi aynı şekilde kademe kademe azaltılarak ilk duruma indirilmektedir.

Canlandırmanın gerçekçiliği açısından uygulamada bir bireyin algı bölgesinin hiçbir şekilde varsayılan ilk değerinin 2 katını aşmasına izin verilmemektedir. Algı bölgesinin her aşamada değişim değeri ve üst limit, takım arama davranışının çalışmasını belirleyen temel parametrelerdir.



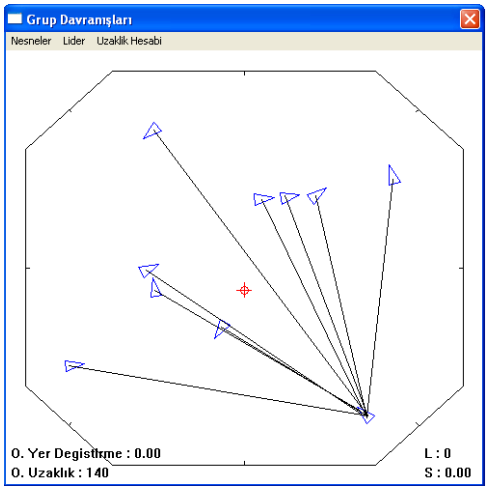
Şekil 5.4 Takım arama davranışı

5.2 Takım Başarım Ölçütlerinin Gerçeklenmesi

Bu bölümde, önerdiğimiz takım modeli ile daha önce var olan sürü modelinin karşılaştırılabilmesi için takım başarım ölçütlerinin gerçeklenmesi ve takım başarım puanının hesaplanması detaylı olarak incelenmiştir.

5.2.1 Ortalama uzaklığın hesaplanması

Yüksek başarımlı bir takım davranışını belirleyen unsurlardan birincisi, etmenlerin birbirlerine yakın olmasının gerekliliğidir. Etmenlerin birbirlerine olan ortalama uzaklığını bulmak için, canlandırmanın her karesinde her etmenin diğer etmenlere olan ortalama uzaklıkları bulunur. Bu uzaklıkların toplamının ortalaması olan tek bir değer, o andaki etmenler arası ortalama uzaklığı verir.



Şekil 5.5 Takım içi ortalama uzaklığın hesaplanması

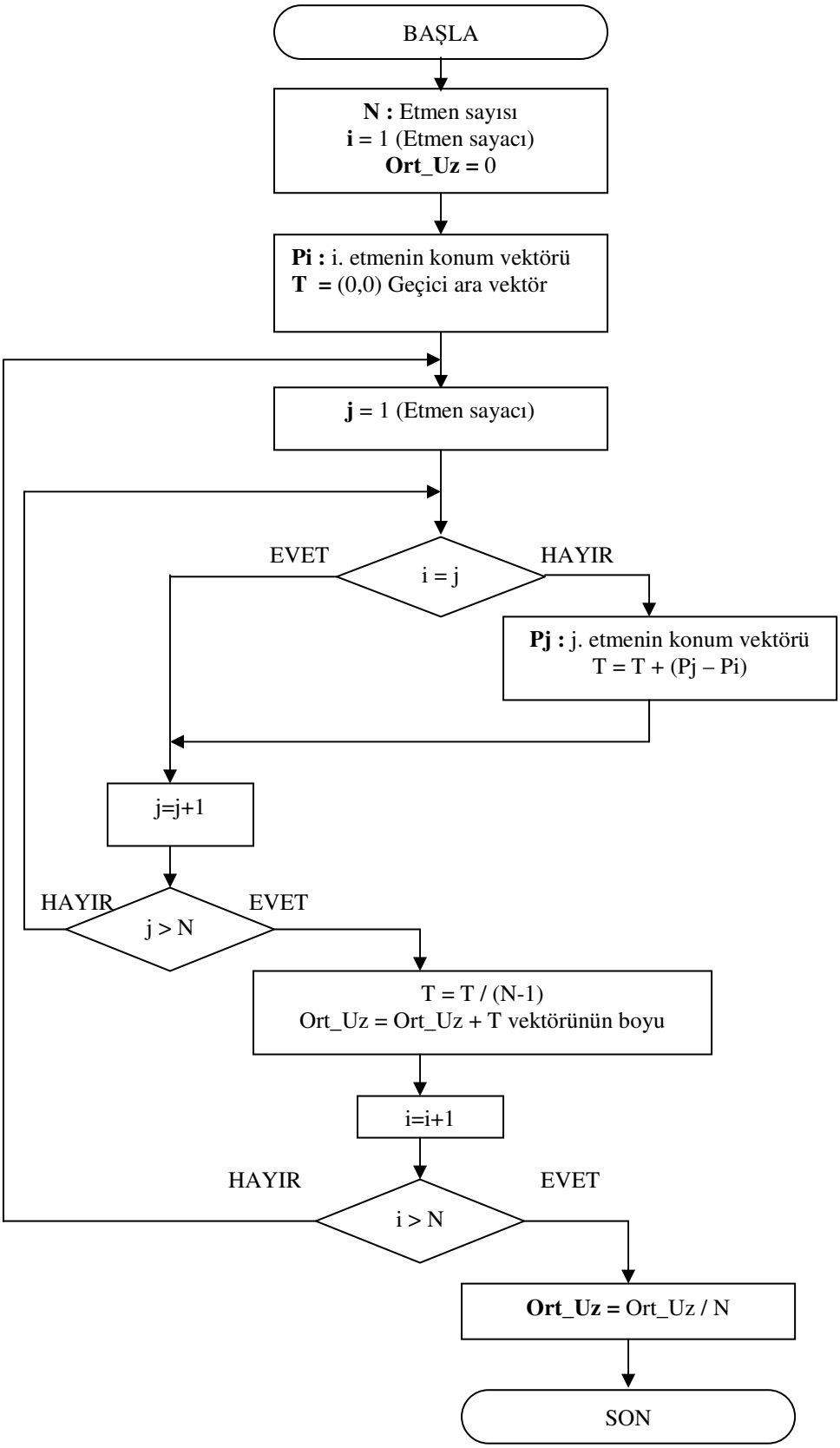
Örnek olarak n etmenden oluşan bir canlandırma ortamında belirli bir anda etmenlerin konumlarının Şekil 5.5'deki gibi olduğunu düşünelim. Takımda yer alan bir etmenin diğer etmenlere olan ortalama uzaklığı (5.1) eşitliği ile hesaplanabilir :

$$d = \frac{1}{n-1}\sum_{i=0}^{n-1}(\overrightarrow{p_i} - \overrightarrow{p}) \tag{5.1}$$

Verilen (5.1) eşitliğinde yer alan $\overrightarrow{p_i}$, i. etmenin konum vektörüdür. Buna göre canlandırmanın her çerçevesi için hesaplanan anlık ortalama uzaklık formülü ise (5.2) eşitliğinde verilmiştir.

$$d_{ort} = \frac{1}{n}\sum_{i=0}^{n} d_i \tag{5.2}$$

Bu eşitliklerin gerçeklendiği kod bloğu, genel çerçeve güncelleme yöntemi içinde yer alır ve canlandırma ortamında bulunan her etmen için ayrı ayrı hesaplanır. Şekil 5.6'da bu işlemleri içeren akış diyagramı verilmiştir.



Şekil 5.6 Ortalama uzaklık hesabına ilişkin akış diyagramı

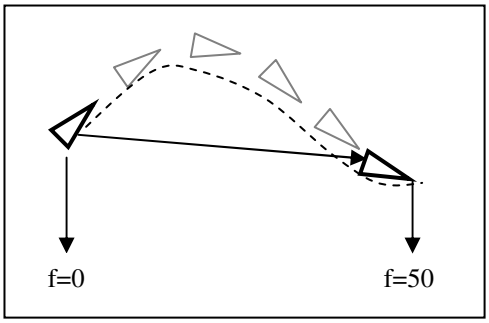
Şekil 5.6'da verilmiş olan akış diyagramı çift döngü içerdiğinden karmaşıklığı $O(n^2)$ 'dir. Yani canlandırmadaki etmen sayısı arttıkça, işlem sayısı üssel olarak artış gösterir. Dolayısıyla etmen sayısının artması sistem başarımında bir düşüşe neden olacaktır. Buna bir çözüm olarak her etmenin birbirlerine olan uzaklıklarının ayrı ayrı hesaplanması yerine, her etmenin sadece öndere olan uzaklıklarının ortalamasının hesaplanması, karmaşıklığı O(n)'e indirecektir.

Ancak önerilen bu yöntemin sorunsuz çalışabilmesi için sistemde her zaman tanımlı bir önderin olması gerekir. Ayrıca bu önderin takıma göre olan konumu, hesaplanacak ortalama uzaklık değerini doğrudan etkiler. Takımdan uzakta olan bir önder, ortalama uzaklık değerinin yüksek çıkmasına, dolayısıyla takım başarım puanının düşük olmasına neden olur.

5.2.2 Alınan ortalama yolun hesaplanması

Yüksek başarımlı takım hareketini belirleyen unsurlardan bir diğeri takımdaki etmenlerin serbest hareket edebilmeleri ve mümkün olan en fazla yolu alabilmeleridir. Alınan ortalama yolun az olması, canlandırmada oluşan bir tıkanıklığı işaret eder. Bölüm 4'de ele alınmış olan önder değişim algoritmalarının devreye girebilmesi için canlandırmadaki tıkanıklığın sezilebilmesi gerekir. Bunun için incelenmesi gereken parametre, alınan ortalama yol parametresidir.

Bir takımda alınan ortalama yol, takımdaki her bir etmenin belirli bir zaman diliminde aldığı yol değerlerinin toplamının etmen sayısına bölünmesi ile bulunur. Tek bir etmen için bu hesap, Şekil 5.7'de gösterilmektedir.



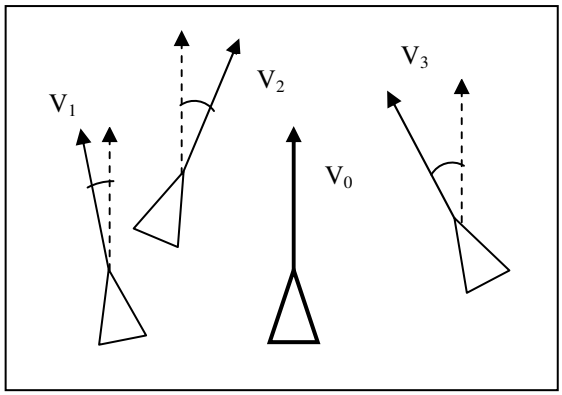
Şekil 5.7 Ortalama yol hesabı

Bir etmenin belirli bir zaman aralığında aldığı toplam yol, ölçülebilen kısa bir zaman aralığında yaptığı yer değiştirmelerin toplamına eşittir. Fakat bu hesabı canlandırmanın her adımında tüm etmenler için yapmak, sistem başarımını olumsuz yönde etkileyecektir. Bu

sebeple hesaplama zaman aralığı uygulamada 50 çerçeve olarak alınmıştır. Bu değerin çok düşük seçilmesi, sistem başarımında bir düşüşe; çok yüksek seçilmesi ise hesaplanan değerlerin ciddi hata payları içermesine sebebiyet verir.

5.2.3 Ortalama yön farkının hesaplanması

Bazı durumlarda canlandırma ortamındaki etmenlerin birbirlerine yakın olarak hızlı hareket etmeleri, bunun bir takım hareketi olduğu anlamına gelmeyebilir. Tamamen farklı yönlere hareket eden etmenler belirli bir *t* anında tesadüfen birbirlerinin oldukça yakınında bulunabilir. Bu durumda etmenlerin aldıkları ortalama yolun fazla olması ve aralarındaki ortalama uzaklık değerinin düşük olması sebebiyle yukarıda anlatılan iki ölçüt de olumlu sonuç verecektir. Fakat bu senaryo bir takım hareketi içermemektedir; bunu sezebilmek için bireylerin yönelimleri ve hız vektörleri de incelenmelidir. Sonuç olarak yüksek başarımlı takım hareketinin oluşabilmesi için aynı zamanda etmenler arası yön farklarının ortalamasının da düşük bir değerde olması gerekir.



Şekil 5.8 Ortalama yön farkı hesabı

Takımdaki etmenlerin yön farklarının ortalamasını hesaplayabilmek için her bir etmenin diğer bireylerle olan yön farklarını ayrı ayrı bulmak ve bulunan sonuçların ortalamasının alınması gerekmektedir. Tek bir etmen için, diğer bireylerle olan ortalama yön bilgisi farkı (5.3) eşitliğinde verilmiştir. Bu eşitlikte yer alan $\overrightarrow{v_0}$ ve $\overrightarrow{v_1}$ Şekil 5.8'de görüldüğü gibi etmenlerin normalize edilmiş yönelim vektörlerini ifade etmektedir.

$$\theta_i = \frac{1}{n}\sum_{k=1}^{n} arcCos(\overrightarrow{v_i}.\overrightarrow{v_k}) \tag{5.3}$$

(5.3) eşitliğinden hareketle canlandırmanın her çerçevesi için hesaplanan takım anlık ortalama yön bilgisi farkı (5.4) eşitliğinde verilmiştir.

$$\theta_{ort} = \frac{1}{n}\sum_{i=0}^{n}\theta_i \tag{5.4}$$

Tıpkı ortalama uzaklığın hesaplanmasında yapıldığı gibi bu noktada da etmenlerin kendi aralarındaki ortalama yönelim farklarının hesaplanması yerine sadece önder ile olan ortalama yön farkının hesaplanması düşünülebilir. Fakat bu yöntem uygulamada iyi sonuçlar vermeyebilir, zira önder genel yön belirleyicidir ve önderin yön değiştirmesi durumunda takım bireyleri buna hemen tepki veremeyebilecekleri için belirli bir süre içinde takım bireyleri ile önder arasında olan ortalama yön farkları yüksek çıkacak; bu da takım başarım puanını olumsuz etkileyecektir.

5.3 Takım Başarım Puanının Hesaplanması

Bölüm 3.5'de sözü edilmiş olan üç temel ölçüt, bir takım canlandırmasının başarısını ölçmek için gerekli parametreleri oluşturur. Başarımı yüksek bir takım canlandırmasında, etmenlerin birbirleriyle yakın, ortalama yer değiştirmelerinin yüksek ve ortalama yönelim farklarının düşük olması beklenir. Bu parametreler kullanılarak başarım puanını ortaya çıkaracak genel bir eşitlik yazılabilir.

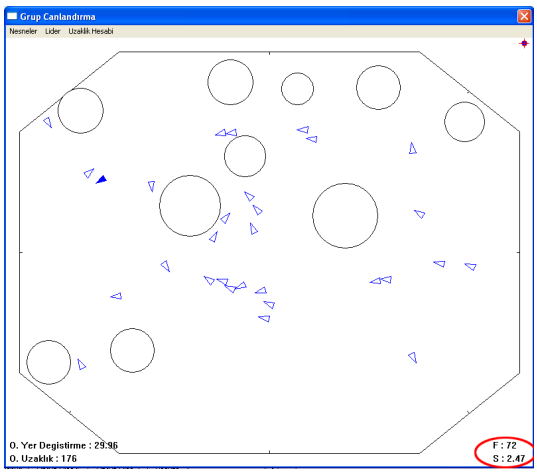
- Etmenlerin birbirine olan ortalama uzaklık parametresi (d),
- Etmenlerin aldıkları ortalama yol parametresi (x),
- Etmenlerin ortalama mutlak yönelim farklarına ilişkin parametre (θ)

olarak ele alınırsa, hesaplanacak başarım puanı (S), x değeri ile doğru orantılı, d ve θ değerleri ile ters orantılıdır. Bu durumda başarım puanı,

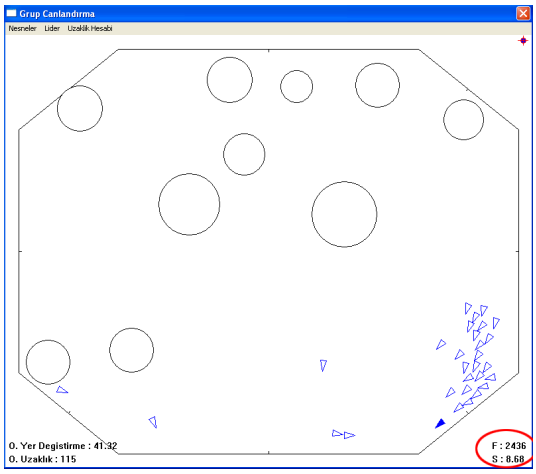
$$S = \frac{k_1 x}{(k_2 d) + (k_3 \theta)} \tag{5.5}$$

eşitliğindeki gibi hesaplanabilir. (5.5) eşitliğindeki k_i katsayıları, her bir ölçütün ağırlık değerleridir. Böylelikle takım başarımının belirli ölçütlere göre daha duyarlı olması sağlanmaktadır.

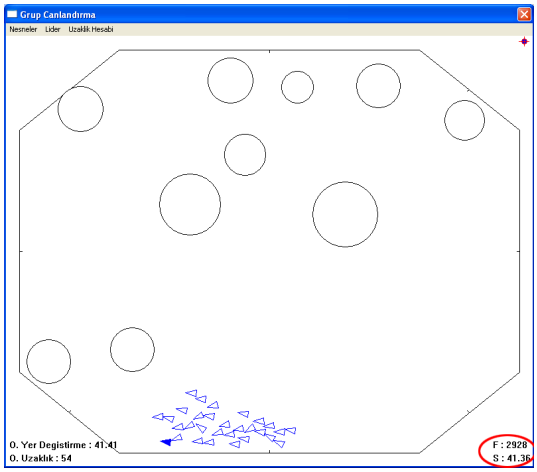
Örnek olarak Şekil 5.9'da aynı canlandırmanın farklı anlarından alınmış ekran görüntüleri yer almaktadır. Görüntülerin sağ altında elips ile işaretlenmiş bilgilerden ilki çerçeve numarası, diğeri ise o anda hesaplanmış olan takım başarım puanıdır.



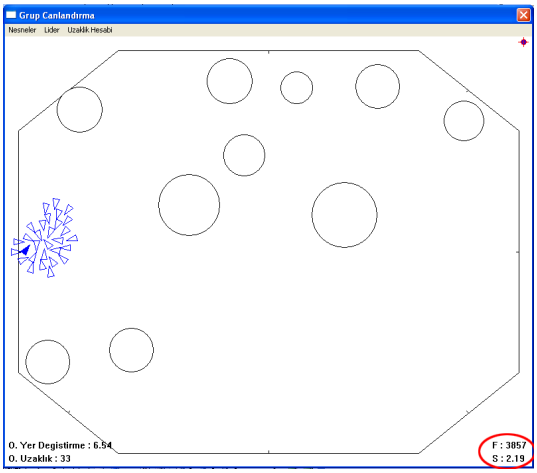
(a) 72. çerçeve



(b) 2436. çerçeve



(c) 2928. çerçeve



(d) 3857. çerçeve

Şekil 5.9 Canlandırmanın farklı anlarına ait takım başarım puanları

Şekil 5.9'da verilen dört farklı çerçevedeki takım başarım durumlarının değerlendirmesi aşağıda verilmiştir.

- (a) 72. çerçevede alınmış ekran görüntüsüdür. Canlandırmanın henüz ilk adımlarında etmenlerin başlangıç konumları rastgele seçildiğinden herhangi bir gruplaşma ve yönelim birliği olmamıştır. Bu sebeple takım başarım puanı 2.47 olarak hesaplanmıştır.

- (b) 2426. çerçevede alınmış ekran görüntüsüdür. Canlandırma ilerledikçe gruplaşma başlamış ve yönelim birliği sağlanmıştır. Takımdan ayrı kalmış etmenler ve takım arası ortalama uzaklığın biraz fazla olması nedeniyle başarım puanı 8.68 olmuştur.

- (c) 2928. çerçevede alınmış ekran görüntüsüdür. Tüm etmenler önder olarak belirlenmiş bir etmeni izlemekte, yüksek hızla aynı yöne doğru hareket etmektedirler. Hesaplanan takım başarım puanı 41.36'dır.
- (d) 3857. çerçevede alınmış ekran görüntüsüdür. Canlandırma sürdükçe bazı etmenler önderin önünü kapatmış ve önder, dolayısıyla takım hareket edemez duruma gelmiştir. Tıkanma durumunun yaşandığı bu çerçevede ortalama alınan yol çok düşük olduğundan takım başarım puanı 2.19 olarak hesaplanmıştır.

Takım başarım puanı canlandırma süresince denetlenerek takım hareketinin niteliği hakkında bilgi edinilebilir.

5.4 Yazılımın Gerçeklenmesi

Önerdiğimiz önder denetimli takım canlandırma sistemine temel oluşturan *Buckland Benzetim Sistemi*, Mat Buckland (2005) tarafından C++ programlama dilinde nesneye yönelik programlama tekniği kullanılarak gerçeklenmiştir. Buckland, sistemi oluşturan her bir temel birim için yazılımda farklı birer sınıf tanımı yapmış olup bu sınıflardan en önemlileri Çizelge 5.1'de açıklanmıştır.

Çizelge 5.1 Buckland benzetim sisteminde bulunan temel sınıflar

Sınıf Adı	**Tanımı**
Vehicle	Etmen sınıfı. *MovingEntity* sınıfından türemiş bir alt sınıftır ve bir etmenin tüm özellikleri (boyut, hız ve konum vektörleri, vb) bu sınıf içinde tanımlanmaktadır.
Behaviour	Etmenlere ilişkin yönlendirme davranış tanımlamalarını içeren sınıf. Canlandırma ortamında yer alan her etmen, bu sınıfa ait birer nesne ile ilişkilendirilmektedir.
GWorld	Canlandırma ortamına ilişkin tüm bilgileri içeren sınıf. Etmenler, duvarlar, engeller ve canlandırmanın denetiminde kullanılan tüm yöntem ve değişkenler bu sınıf içinde tanımlanmaktadır.

Önerdiğimiz sisteme ilişkin kurulan modeller ve geliştirilen algoritmik yaklaşımlar; Buckland benzetim sisteminde bulunan sınıflar içinde tanımlanıp gerçeklenen yeni üye fonksiyonlar,

yazılıma eklenen yeni sınıflar ve canlandırmanın akışı için kurulan denetim yapıları ile gerçeklenmiştir. İzleyen bölümlerde bu gerçekleme aşamaları ele alınmaktadır.

5.4.1 Yönlendirme davranışlarının gerçeklenmesi

Buckland benzetim sisteminde bulunmayan ve Bölüm 5.1'de tanıtılmış olan yönlendirme davranışları, *Behaviour* sınıfı içinde tanımlanıp gerçeklenmiş olan ve Çizelge 5.2'de açıklanan bir dizi üye fonksiyon ile sağlanmıştır. Yönlendirme davranışlarının yürütülmesi sonucunda, davranışı gerçekleyen etmenin hareketini belirleyecek olan vektör üretilmektedir. Gerçeklemeye ilişkin kodlar, Ek 1'de sunulmuştur.

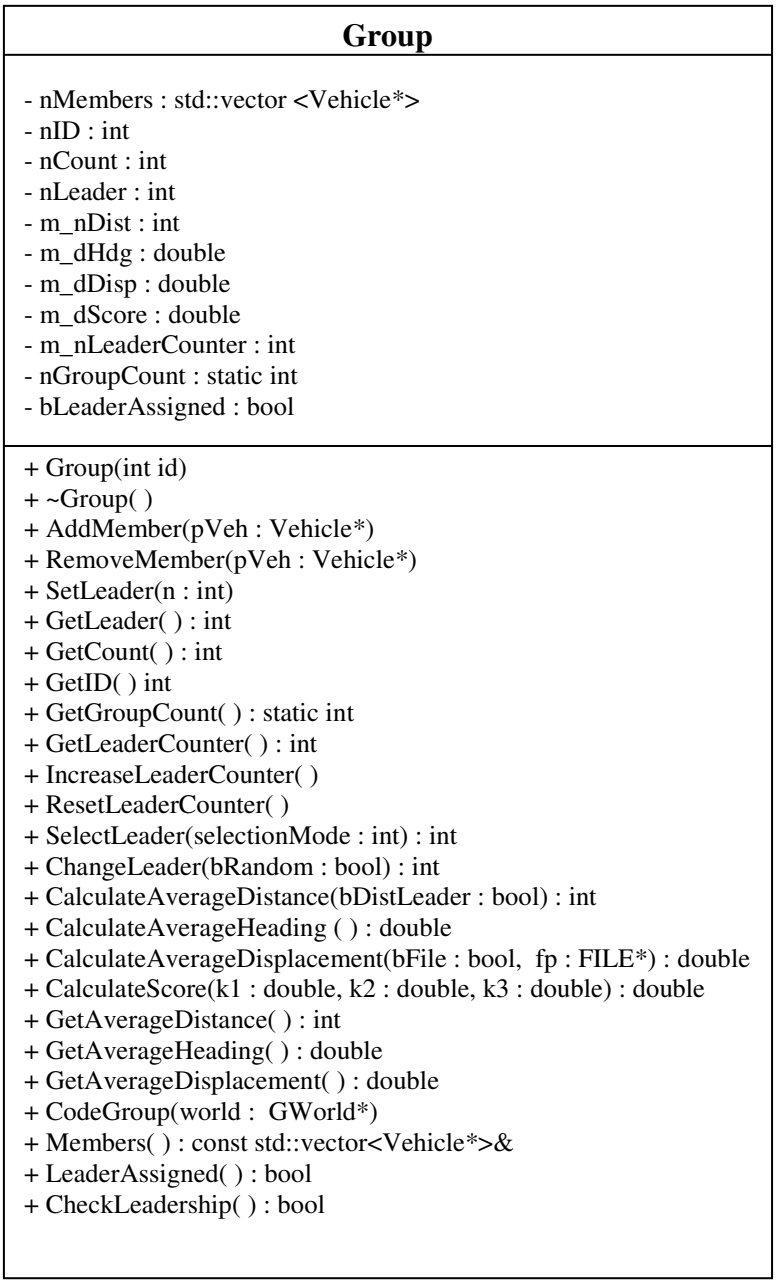
Çizelge 5.2 Gerçeklenen yönlendirme davranışları

Yönlendirme Davranışı	**Behaviour Sınıfına Eklenen Fonksiyon**
Duvar İzleme Davranışı	*Vector2D FollowWall const std::vector<Wall2D> &walls)* Parametre olarak verilen *walls*, izlenecek duvarı oluşturan doğru parçalarına ilişkin bir vektör dizidir.
Önder Takip, Sıralı Takip Davranışları	*Vector2D OffsetPursuit (const Vehicle* agent, const Vector2D offset, bool bDuckling)* Parametre olarak verilen *agent* etmeni, *offset* uzaklığı korunacak şekilde izlenir. Normal takip veya sıralı takip seçeneği, *bDuckling* parametresi ile belirlenmektedir.
Saldırı Davranışı	*Vector2D Attack( )* Saldırı davranışını yürüten etmen, algı bölgesi içinde kendisine en yakın etmene doğru yönelme davranışını gerçekleştirmektedir.
Takım Arama Davranışı	*Vector2D LookForAGroup( )* Takım arama davranışını yürüten etmenin algı bölgesi, bu bölgenin içine farklı bir etmen girene kadar kademeli olarak genişletilir. Bu noktadan sonra davranışı yürüten etmen, sezilen etmene doğru yönelir ve algı bölgesi kademeli olarak küçültülür.
Önderin Önünden Kaçınma Davranışı	*Vector2D EvadeLeader(const Vehicle* leader, bool bRect)* Parametre olarak verilen *leader,* önünden kaçınılacak olan önder etmendir. Önder etmenin önü olarak tanımlanan bölgenin dikdörtgen veya daire dilimi olması, *bRect* parametresi ile belirlenmektedir.

Ayrıca gerçeklenen her bir yönlendirme davranışına ilişkin yardımcı fonksiyonlar (davranışın etmen üzerinde aktif olup olmadığının testi, davranışı etmene atamak, davranışı etmenden silmek) ve bu davranışların etmenin hareket yanıtı üzerindeki ağırlık değerleri de tanımlanmış ve sisteme eklenmiştir.

5.4.2 Takım modelinin gerçeklenmesi

Bölüm 3.1'de önerdiğimiz takım modelinin yazılım tarafında gerçeklenmesi, oluşturulan *Group* sınıfı ile sağlanmıştır. Group sınıfına ilişkin UML diyagramı Şekil 5.10'da verilmiştir.



Şekil 5.10 *Group* sınıfına ilişkin UML diyagramı

Önerdiğimiz önder modeli için ayrı bir sınıf tanımı yapılmamış; geliştirdiğimiz *Group* sınıfı içine önder ve önderlik denetimleri ile ilgili değişkenler ve fonksiyonlar eklenmiştir. Sonuç olarak her takım önderini kendi içinde barındırmaktadır ve bu önder aslında takım bireylerinden biri olarak seçilmektedir. Önder değişim algoritmaları sonucu seçilen yeni önder de, aynı şekilde takım bireylerinden biri olacaktır.

Group sınıfının üye değişkenleri ve bu değişkenlerin kullanım amaçları Çizelge 5.3'de verilmektedir. Görüldüğü gibi önder ve önderlik denetimi ile ilgili değişkenler de bu sınıfın içine dahil edilmiştir. Böylelikle "bir önder denetiminde hareket eden bireyler topluluğu" şeklinde tanımlamış olduğumuz *takım* kavramı, yazılımda öndere ilişkin bilgileri de içeren *Group* sınıfının gerçeklenmesiyle modellenmiş olmaktadır.

Çizelge 5.3 *Group* sınıfına ait üye değişkenler

Değişken Adı	**Tipi**	**Açıklama**
nMembers	std::vector <Vehicle*>	Takım bireylerine ilişkin vektör dizisi. Dizideki her bir eleman, canlandırma başlangıcında yaratılan bir etmene işaretçidir.
nID	int	Takım kimlik bilgisi.
nCount	int	Takımdaki toplam birey sayısı.
nLeader	int	Takım önderinin indis bilgisi.
m_nDist	int	Takım bireyleri arasındaki ortalama uzaklık.
m_dHdg	double	Takım bireylerinin ortalama yönelim farkı.
m_dDisp	double	Takım bireylerinin aldıkları ortalama yol.
m_nLeaderCounter	int	Takm önderi değiştiğinde yeni öndere tanınan süreye ilişkin sayaç değişken.
nGroupCount	int	Canlandırmadaki toplam takım sayısı.
bLeaderAssigned	boolean	Takıma önder atanıp atanmadığının bilgisi.

Group sınıfının üye fonksiyonlarını yürüterek oluşturulan bir takım nesnesi üzerinde çeşitli işlemler yapmak mümkündür. Bu fonksiyonların kısa bir özeti aşağıda verilmiştir.

Group(int id), ~Group() fonksiyonları, Group sınıfına ilişkin kurucu ve yok edici fonksiyonlardır. Yapılandırıcı fonksiyonda takım nesnesinin değişkenlerine ilişkin ilk değerler tanımlanmaktadır.

AddMember(Vehicle pVeh), RemoveMember(Vehicle *pVeh)* fonksiyonları, takıma yeni bir etmen eklemek ve bir etmeni takımdan çıkarmak için kullanılan fonksiyonlardır.

Members( ) fonksiyonu ile takım üyelerine erişilmektedir.

GetCount( ) fonksiyonu, bir takımdaki toplam etmen sayısını geri döndürmektedir.

GetID( ) fonksiyonu ile takımın kimlik bilgisi öğrenilmektedir.

GetGroupCount( ) fonksiyonu, belirli bir anda canlandırma ortamındaki toplam takım sayısını öğrenmek için kullanılır. Fonksiyon; kurucu fonksiyonun her çağrılmasında bir artan ve yok edici fonksiyonun her çağrılmasında bir azalan *nGroupCount* statik değişkenine erişim sağlamaktadır.

SetLeader(int n), GetLeader( ) fonksiyonları, bir takıma önder tayin etmek ve bir takımdaki önderin numarasını öğrenmek için kullanılmaktadır.

CheckLeadership( ) ve *LeaderAssigned( )* fonksiyonları ile bir takıma önder atanıp atanmadığı sınanabilmektedir.

GetLeaderCounter( ), IncreaseLeaderCounter( ), ResetLeaderCounter( ) fonksiyonları, yeni atanmış bir öndere tanınan süre bilgisini geri döndüren ve değiştiren fonksiyonlardır. Özellikle canlandırmada bir tıkanıklık sezilmesinin sonucu olarak önder değişim algoritmalarının yürütülmesinin ardından, yeni seçilen önderin bu tıkanıklığı ilk birkaç canlandırma çerçevesi içinde çözebilmesi mümkün olamayacağından yeni öndere bir süre tanınmaktadır. Eğer önder bu süre sonunda da tıkanıklığı çözemezse yeni bir önder değişim işlemi yapılmaktadır.

SelectLeader(int selectionMode) fonksiyonu ile Bölüm 3.2'de önerilen 4 farklı yöntemden herhangi birine göre önder seçimi yapılmaktadır.

ChangeLeader(bool bRandom) fonksiyonu, canlandırmada bir tıkanıklığın sezilmesinin ardından önder değişimi yapmak için kullanılan fonksiyondur.

CalculateAverageDistance(bool bDistLeader) fonksiyonu, Bölüm 5.2.1'de ele alınmış olan takım bireyleri arasındaki ortalama uzaklığı hesaplayan fonksiyondur. Fonksiyonun

parametresi, bu hesaplamanın sadece öndere göre yapılıp yapılmayacağını belirlemek için kullanılmaktadır.

CalculateAverageDisplacement(bool bFile, FILE fp)* fonksiyonu, Bölüm 5.2.2'de ele alınmış olan takım bireylerinin belirli bir zaman dilimi içinde aldığı ortalama yol bilgisinin hesaplandığı fonksiyondur. Fonksiyonun parametreleri, hesaplanan değerlerin bir dosyaya yazdırılması ile ilgili bilgileri içermektedir.

CalculateAverageHeading( ) fonksiyonu, Bölüm 5.2.3'de ele alınmış olan takım bireyleri arasındaki ortalama yön farkının hesaplandığı fonksiyondur.

CalculateScore(double k1, double k2, double k3) fonksiyonu ile Bölüm 5.3'de anlatılmış olan takım başarım puanı hesaplanmaktadır.

GetAverageDistance( ), GetAverageDisplacement( ) ve *GetAverageHeading( )* fonksiyonları ile takım nesnesine ilişkin olarak hesaplanmış olan ortalama uzaklık, ortalama yer değiştirme ve ortalama yön farkı bilgilerinin tutulduğu değişkenlere erişilebilmektedir.

CodeGroup(GWorld world)* fonksiyonu, Bölüm 7'de ele alınacak olan çoklu takımların takım bazındaki denetimi için kullanılan bir fonksiyondur. Bu fonksiyon, takım bireyleri arasında bir bölünmenin meydana geldiği durumlarda ortaya çıkacak olan yeni takımların belirlenmesi için kullanılmaktadır.

Group sınıfında tanımlı bu üye fonksiyonlara ilişkin program kodu Ek 2'de sunulmuştur.

Canlandırma ortamında takımların oluşumu ve denetimi için yürütülmesi gereken bir dizi fonksiyon ise *GWorld* sınıfı içine eklenmiştir. İzleyen paragraflarda açıklanmış olan bu fonksiyonların *Group* sınıfı dışında tanımlanmasının temel nedeni, canlandırma bilgilerinin yer aldığı *GWorld* sınıfı değişkenlerine bağımlı olmalarıdır. Çünkü takım oluşumları ve denetimleri *GWorld* sınıfı içinde yapılmaktadır.

Bölüm 7'de ele alınacak olan çoklu takımların oluşumu ve etkileşimine ilişkin fonksiyonlar *GWorld* sınıfı içinde tanımlanıp gerçeklenmiştir. Bu fonksiyonlar kısaca aşağıda açıklanmış olup, program kodları Ek 3'de sunulmuştur.

Gro*uping( ) ve SetGroups( )* fonksiyonları, Bölüm 7.1'de önerilmiş olan takım demetleme algoritmasının gerçeklendiği fonksiyonlardır.

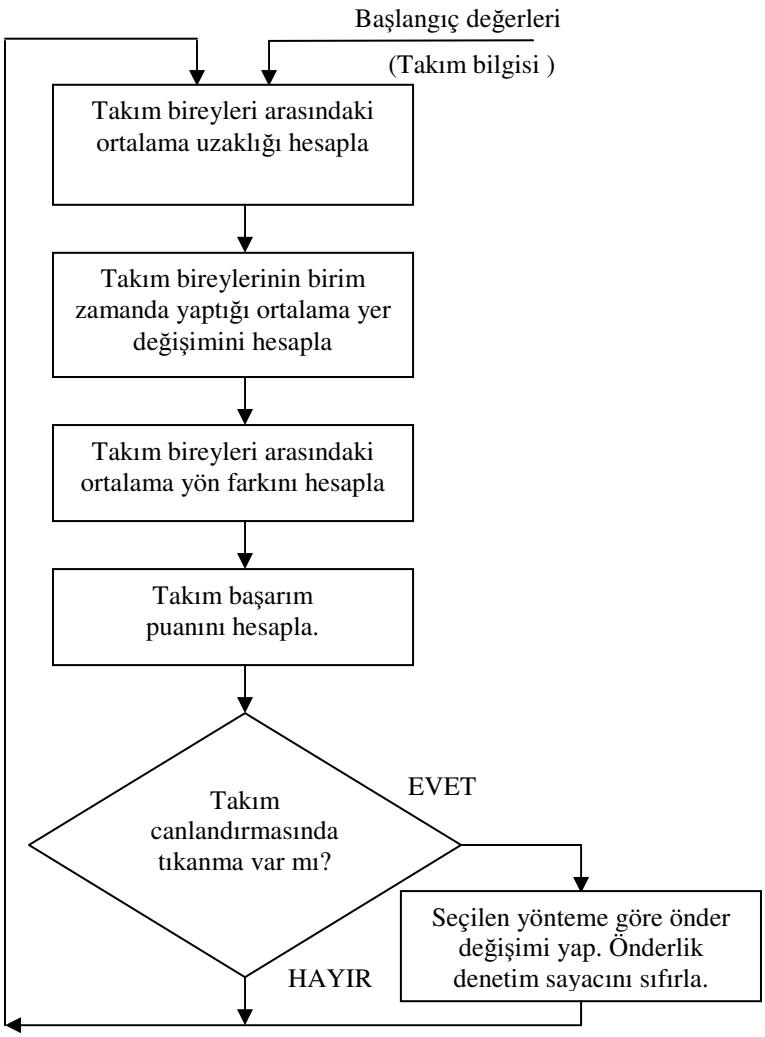
AssignLeaders( ) fonksiyonu, takım demetleme işleminin ardından belirlenmiş olan takımlara önder atamak için yürütülen fonksiyondur.

JoinByMajority(int limit) fonksiyonu ise Bölüm 7.3.2'de ele alınmış olan çokluğa bağlı birleşme kuralına ilişkin fonksiyondur.

Bunların dışında etmenlerin bir takıma dahil olmalarında, takımdan ayrılmalarında, önder olmalarında ve önderliği bırakmalarında yürütülmesi gereken işlemlere ilişkin bazı fonksiyonlar da etmen sınıfı olan *Vehicle* sınıfı içinde tanımlanıp gerçeklenmiştir.

5.4.3 Canlandırma güncelleme çevrimi

Takım modelinin oluşturulup yazılıma eklenmesinin bir sonucu olarak her canlandırma karesinde yürütülen canlandırma güncelleme fonksiyonu, takımlar üzerinde tıkanma denetimi yapıp olası bir tıkanmada ilgili takımda bir önder değişimini gerçekleyecek şekilde genişletilmiştir. Buna göre oluşan yeni canlandırma güncelleme çevrimi içinde tek bir takıma ilişkin hesaplama ve denetimler Şekil 5.11'de gösterilmiştir.



Şekil 5.11 Tek bir takıma ilişkin canlandırma güncelleme çevrimi

6. UYGULAMALAR ve SONUÇLARI

Yüksek başarımlı bir takım canlandırması, herhangi bir tıkanıklık olmadan etmenlerin birbirlerine yakın ve yüksek bir ortalama hızda hareket edebilmeleri şeklinde tanımlanmıştır. Bunun bir ölçütü olarak Bölüm 5.3'de ele aldığımız takım başarım puanı ile bir takımın hareketi sayısal bir değere karşı düşürülebilir. Öte yandan takım başarım puanının düşük olması, takımın mutlaka bir tıkanmaya girdiği anlamına gelmez. Örnek olarak, etmenlerin birbirlerinden uzak ve farklı yönlere doğru hareket ediyor olmaları, düşük bir takım başarım puanının ortaya çıkmasına neden olacaktır. Bu yüzden takımdaki bir tıkanıklığı sezmek için kullanılabilecek daha etkin bir parametre, takımın yaptığı ortalama yer değiştirme miktarıdır.

Bölüm 4'te incelediğimiz tıkanıklık durumlarının ve önder değişim algoritmalarının takımın genel hareketi üzerindeki etkilerini gözleyebilmek için engelli ve engelsiz ortamlarda bazı canlandırmalar gerçekleştirilmiştir. Her iki ortam için gerçeklenen yöntemler aşağıda listelenmiş olup, bu farklı durumlar için canlandırma süresince değişen takım başarım puanı ve takımın ortalama yer değiştirme değerleri bir dosyaya kaydedilmiştir.

- Değişmeyen önderlik (körü körüne önderlik)
- Tıkanıklığı sezme ve rastgele önder değişimi
- Tıkanıklığı sezme ve en uzaktaki bireyin önder olması
- Önderin önünden kaçınma algoritmasının yürütülmesi

Canlandırmaların ortak parametreleri için seçilmiş olan değerler ise aşağıdadır :

Toplam etmen sayısı : 30

Canlandırma ortamı : 600x600 benek boyutlarında, sınırları duvar ile çevrelenmiş.

Canlandırma sayısı : 4 farklı yöntemin her biri için 10'ar canlandırma.

Canlandırma süresi : 2500 çerçeve (yaklaşık 30 saniye).

Etmenlerin maksimum hızı : 100 benek/sn.

Etmenlerin başlangıç konumları : Rastgele.

İlk önderin seçimi : Rastgele.

Tıkanıklık eşik değeri : 10 benek.

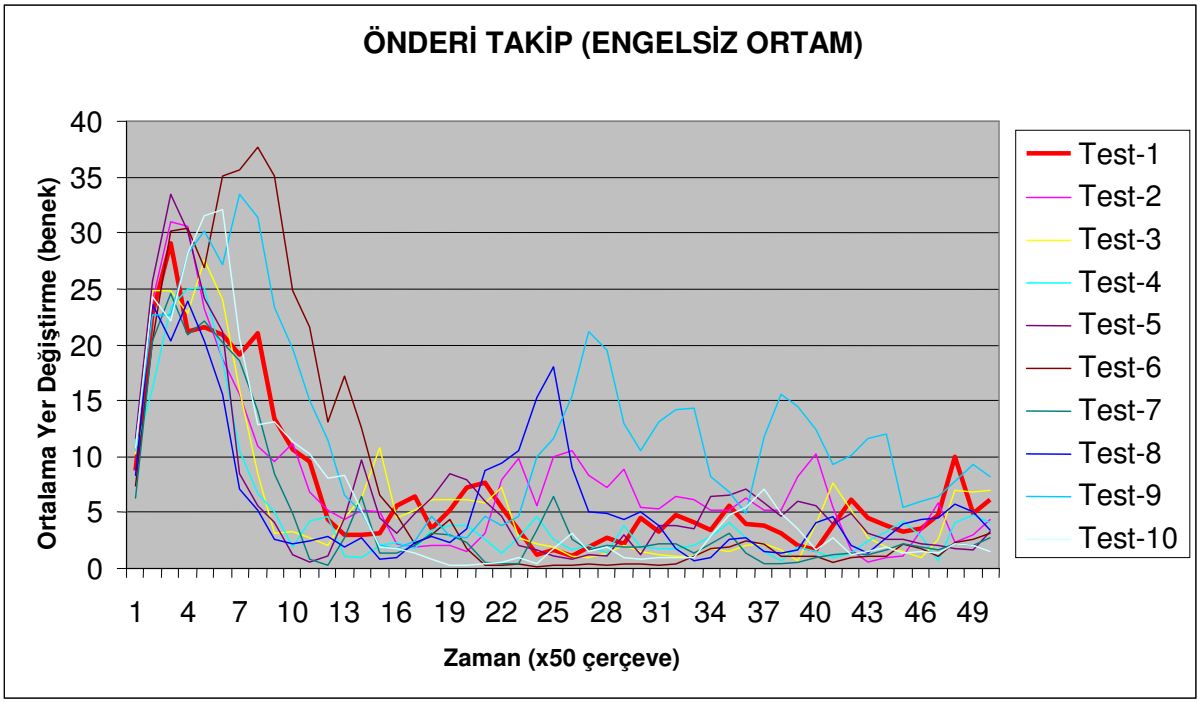
Öndere tıkanıklığı çözmesi için tanınan süre: 100 çerçeve.

Takım başarım puanı sabitleri : k_1=100, k_2=10, k_3=10

Yapılan bu canlandırmaların gerçekleştirilme amaçları; incelediğimiz yöntemlerin gerek engelsiz gerek engelli ortamlar için çalışıp çalışmadığının sınanması, bu yöntemlerin canlandırma süresince takımın ortalama yer değiştirme değeri ve başarım puanı üzerindeki değişimlerinin incelenmesi ve bu yöntemlerin takımın ortalama yer değiştirme ve takım başarım puanı açısından gerçekten anlamlı bir farkının olup olmadığının belirlenmesidir. Bunun için yapılmış olan bazı istatistiksel testler ve sonuçları, bu bölümün kapsamı içindedir.

6.1. Engelsiz Ortam Canlandırmaları

Engelsiz ortam için ilk olarak etmenlerin önderi körü körüne takip etme davranışının gerçeklendiği ve herhangi bir tıkanıklık kontrolünün yapılmadığı bir canlandırma gerçekleştirilmiş ve bu canlandırma, önder ve etmenlerin farklı başlangıç durumları için 10 kere tekrarlanmıştır. Oluşan ortalama yer değiştirme eğrileri Şekil 6.1'de görülmektedir.



Şekil 6.1 Takım ortalama yer değişimleri - engelsiz ortam, önderi takip

Ortamdaki etmenlerin öndere yönelmeleri sonucunda önderin hareketsiz kalması, aynı zamanda takımın da hareketsiz kalmasına neden olmuştur. Herhangi bir tıkanıklık sınaması ve buna çözüm getirecek bir önder değişimi yapılmadığı için takımın ortalama yer değiştirme değeri birim zamanda 10 benek değerinin altında kalmıştır.

Yapılan 10 farklı test arasından bu yöntemi temsil eden bir eğrinin belirlenebilmesi için iki farklı yöntem kullanılabilir. Bunlardan ilki, oluşan değerlerin ortalamasının alınarak bir

ortalama eğrisinin çıkartılması olacaktır. Fakat bu eğri gerçekleştirilmiş olan canlandırmalardan herhangi birine ait bir eğri olmayacağı gibi, takımın girdiği tıkanmalar da bu eğride görünmeyebilir. İkinci bir yöntem olarak, hesaplanan bu ortalama eğrisine en yakın olan test eğrisi, temsilci eğri olarak seçilebilir. Bunun için her *t* anında, ortalama eğri ile test eğrileri arasındaki hataların kareleri hesaplanır ve bu hata kareleri toplamının en küçük olduğu test eğrisi temsilci eğri olarak belirlenir. Diğer bir deyişle (6.1) eşitliğini minimum yapan i. test eğrisi, seçilecek olan temsilci eğridir.

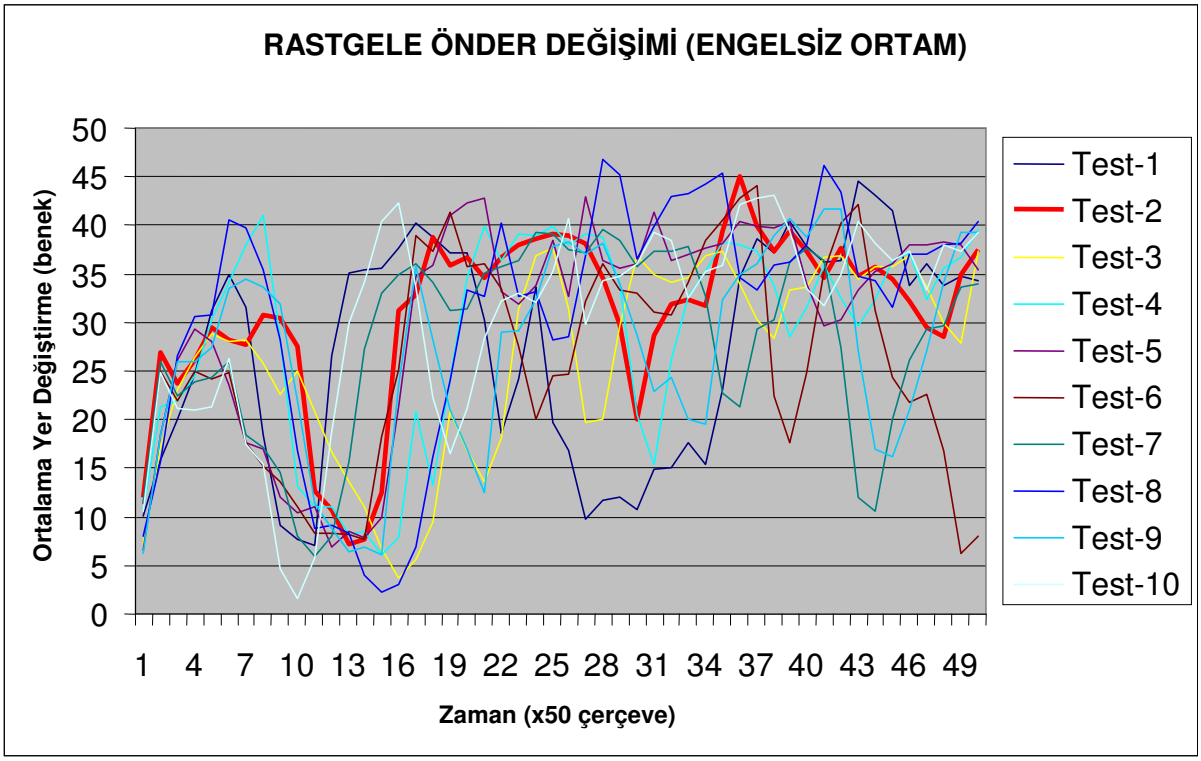
$$\Delta_i = \sum_{t=1}^{50} (d_t - x_t)^2 \qquad \text{i} = 1,2,\ldots 10 \tag{6.1}$$

d_t = hesaplanan ortalama eğrinin t anında aldığı değer.

x_t = i. test eğrisinin t anında aldığı değer.

Bu hesaptan hareketle Şekil 6.1'de koyu renk ile gösterilen eğri, engelsiz ortamda önderi körü körüne takip yöntemini temsil eden eğri olarak belirlenmiştir. Daha ilerde sunulacak olan diğer yöntemler için temsilci eğri seçimleri de benzer şekilde yapılmıştır.

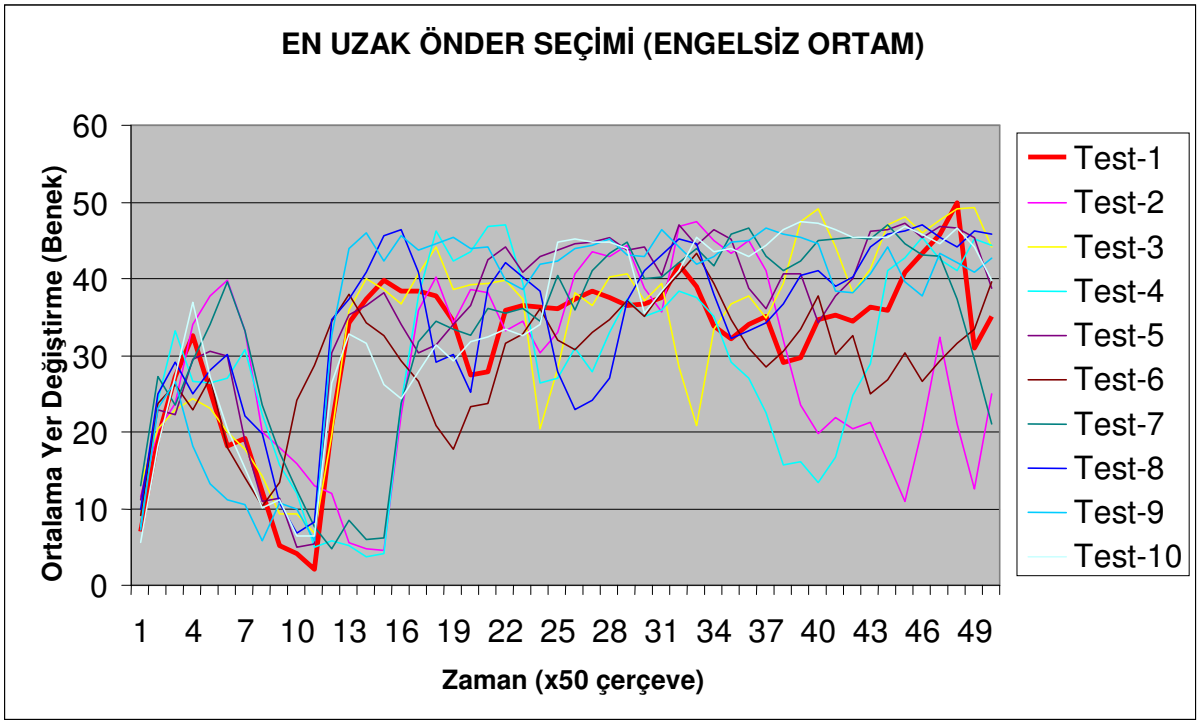
Şekil 6.2'de verilen test sonuçları, engelsiz ortamda bulunan bir takımın tıkanıklık testi sonucunda rastgele önder değişimini gerçekleştirmesine ilişkin eğrilerdir. Bu canlandırma farklı başlangıç konumları için 10 kere tekrarlanmış, yöntemin temsilci eğrisi belirlenmiştir.



Şekil 6.2 Takım ortalama yer değişimleri - engelsiz ortam, rastgele önder değişimi

Ortaya çıkan bu ortalama yer değiştirme eğrileri sistemdeki rastsallığın bir sonucu olarak çok farklı gibi görünse de, ilk tıkanmanın oluştuğu ve takım ortalama yer değiştirme değerinin 10'un altına düştüğü bölgelerdeki eğri eğilimleri oldukça benzerdir. Önder seçimi rastgele olarak yapıldığı için, tıkanıklığın sezilmesinden sonra seçilen yeni önder yine bu takımın içinden bir etmen olarak belirlenirse tıkanıklık çözülmeyecek ve sistem yeni bir önder değişimini daha tetikleyecektir. Dolayısıyla bu tıkanmanın çözülme süresi, seçilecek olan önderin takımın dış bölgesindeki bir etmen olmasına kadar geçen süre ile aynı olacaktır.

Öndere en uzakta olan etmenin yeni önder olarak seçildiği canlandırmalara ilişkin elde edilen eğriler ise Şekil 6.3'te görülmektedir. Takım tıkanmaya girdiğinde seçilecek olan yeni önder, eski öndere en uzak mesafede olan etmen olacağından bu etmen takımın dış bölgesinde yer alacaktır. Ortamda engellerin bulunmaması, hareket yönü değiştirilen yeni önderin önünün açık olması anlamına gelir; bu durumun tek istisnası tıkanmanın bir duvara yakın gerçekleşmesi ve seçilen yeni önderin duvarın hemen yanında olması durumudur. Böyle bir durumda sistem ikinci bir önder değişimini tetikleyecek ve duvara en uzak bulunan birey yeni önder olarak tıkanıklığı çözecektir.



Şekil 6.3 Takım ortalama yer değişimleri - engelsiz ortam, en uzak önder seçimi

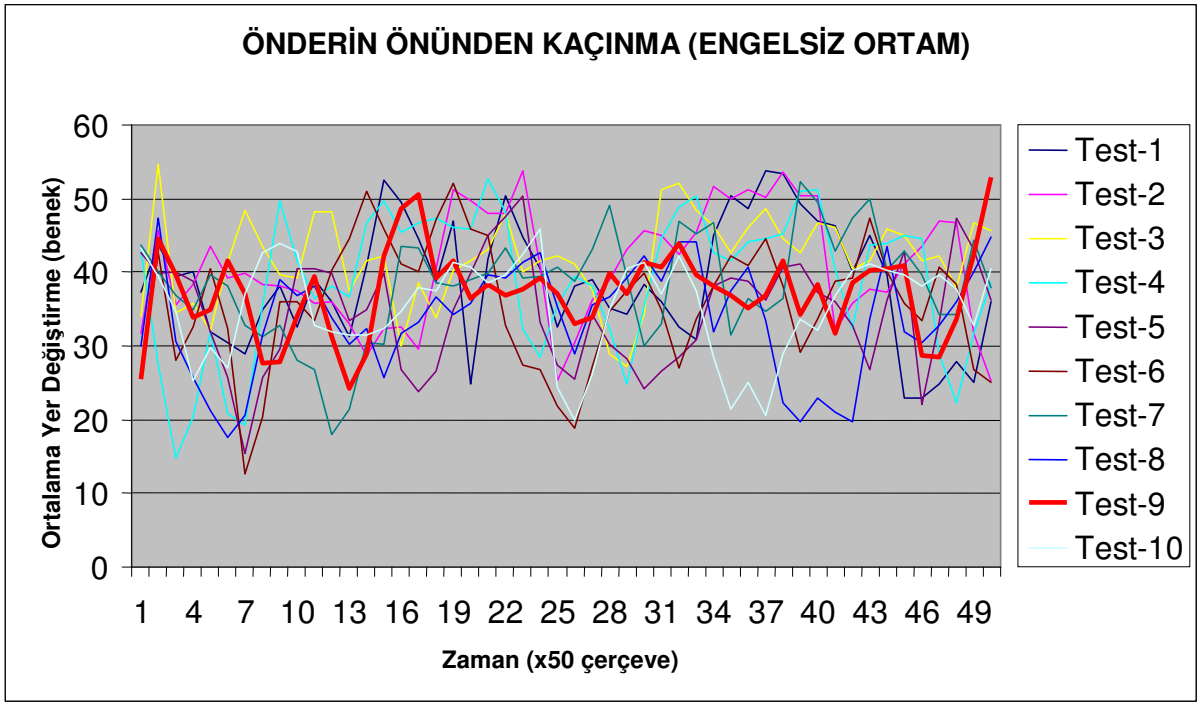
Önder değişiminin rastgele yapıldığı ve en uzak bireyin önder olarak seçildiği testlerde sistem tarafından yapılan önder değişim sayıları Çizelge 6.1'de verilmiştir. Rastgele önder değişiminde yapılan 10 testin 4'ünde tıkanıklık ilk önder değişiminde çözülebilmiştir; buna

karşın en uzak bireyin önder olarak seçiminde bu sayı 8'dir. Çizelgeden de görülebileceği gibi ortalamalar hesaplandığında tıkanıklığın çözülmesi, rastgele önder değişiminde 1.7 önder değişimi, en uzak bireyin yeni önder olarak seçildiği durumlarda ise 1.2 önder değişimi gerektirmiştir.

Çizelge 6.1 Önder değişim sayıları (engelsiz ortam)

	Test1	**Test2**	**Test3**	**Test4**	**Test5**	**Test6**	**Test7**	**Test8**	**Test9**	**Test10**	**Ort.**
Rastgele Önder	2	1	1	2	2	3	2	2	1	1	1.7
En Uzak Önder	1	1	1	2	1	1	2	1	1	1	1.2

Engelsiz ortam için gerçekleştirilen son canlandırmalar, takım bireylerinin önderin önünden kaçınma davranışını yürütmesi sonucunda önderin önünün hiç kapanmadığı ve takımın herhangi bir tıkanmaya girmediği duruma ilişkin testlerdir. Şekil 6.4'te bu testlerin sonucunda ortaya çıkan ortalama takım yer değiştirme eğrileri görülebilir. Takıma ilişkin ortalama yer değiştirme değerleri belirli bir aralıkta değişmekte olup hiçbir zaman kritik seviyenin altına düşmemektedir.



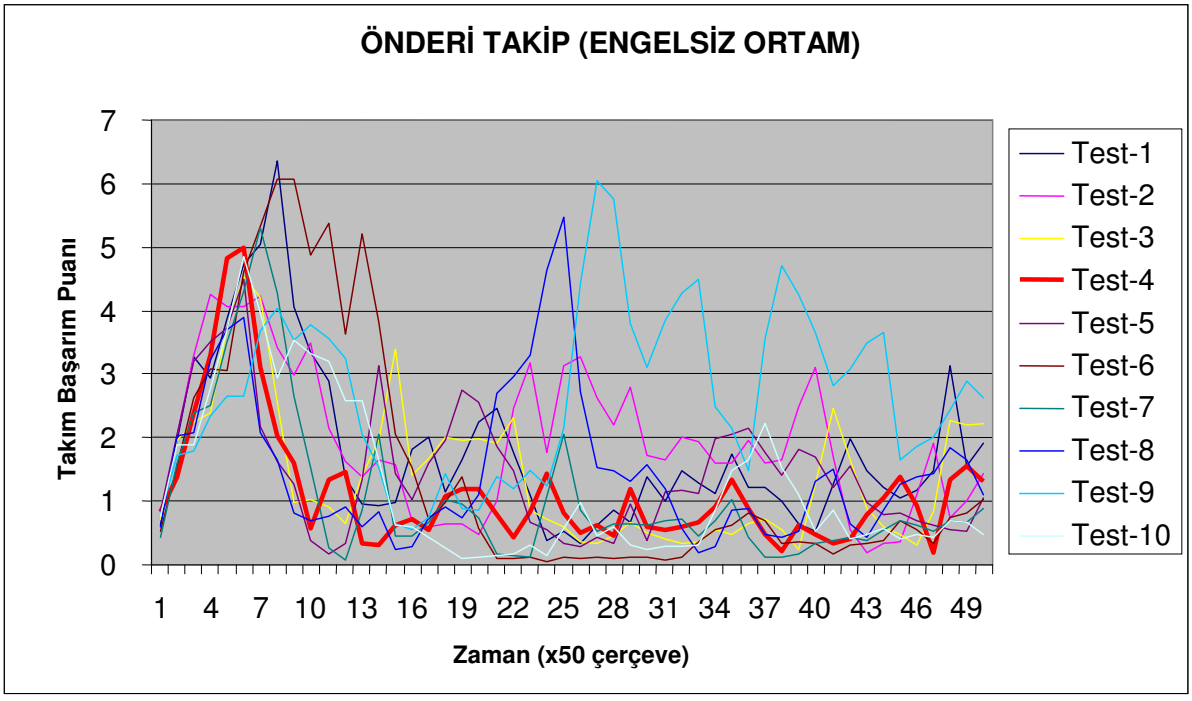
Şekil 6.4 Takım ortalama yer değişimleri - engelsiz ortam, önderin önünden kaçınma

Önderin önünden kaçınma algoritması, tıkanıklık çözümleri arasında diğer yöntemlere göre daha fazla işlem gücü isteyen çözüm olarak görünmektedir. Bu yöntemin sistem başarımı açısından getirdiği ek maliyet Çizelge 6.2'de verilmiştir.

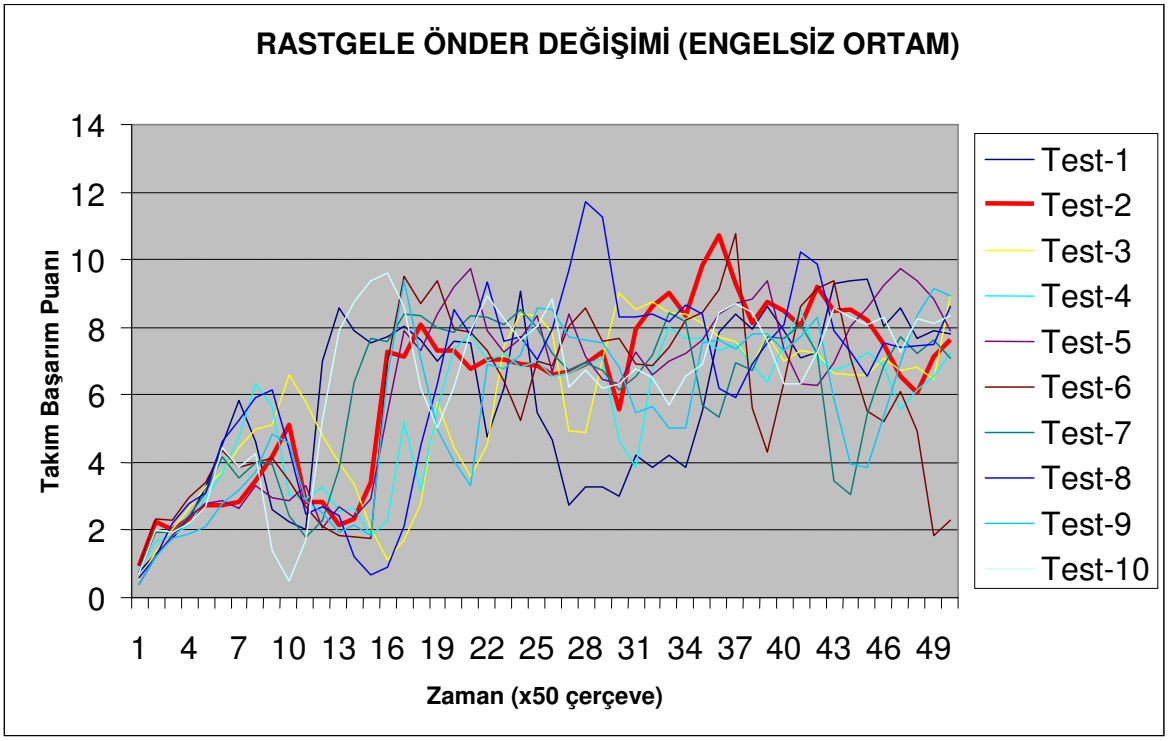
Çizelge 6.2 Önderin önünden kaçınma algoritmasının sistem başarımına etkisi

Etmen Sayısı	**Saniyede oynatılan çerçeve sayısı**	**Önderin önünden kaçınma algoritması ile saniyede oynatılan çerçeve sayısı**	**Çerçeve sayısındaki değişim**
30	62	60	- %3.2
60	31	30	- %3.2

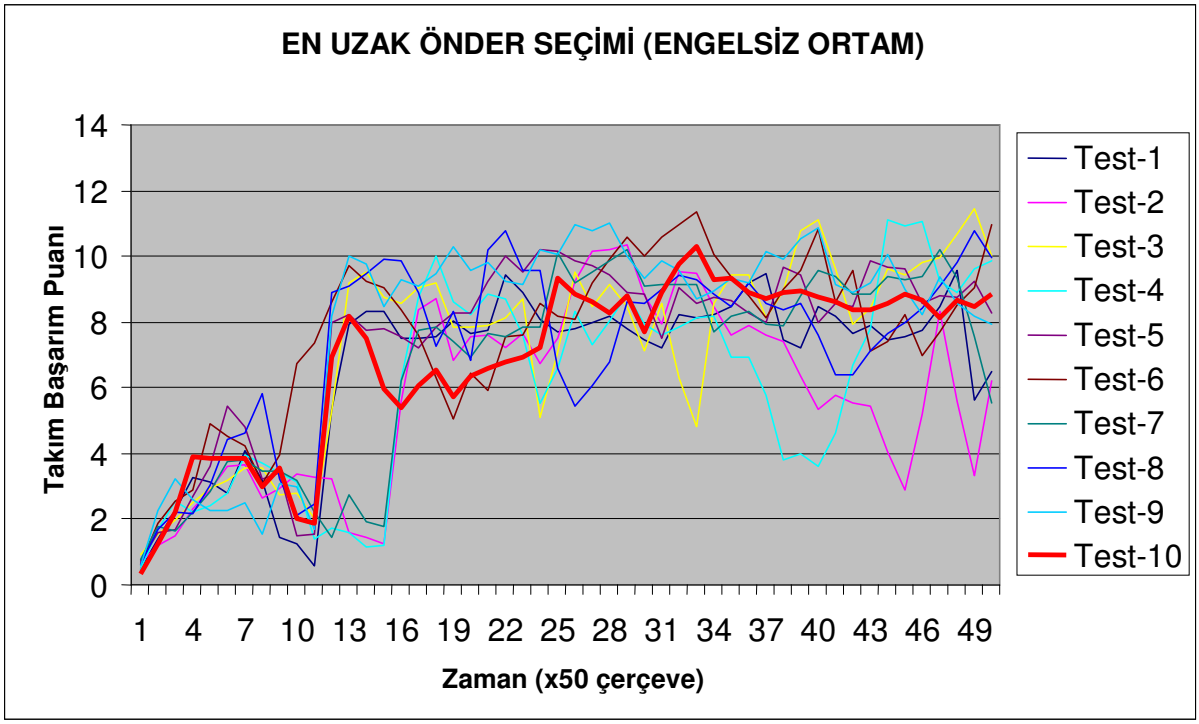
Engelsiz ortamda 4 farklı yöntem için gerçekleştirilen bu canlandırmalarda takımın ortalama yer değiştirme değeri ile birlikte takım başarım puanları da ölçülmüştür. Elde edilen takım başarım puanı eğrileri farklı yöntemler için Şekil 6.5, Şekil 6.6, Şekil 6.7 ve Şekil 6.8'de verilmiştir.



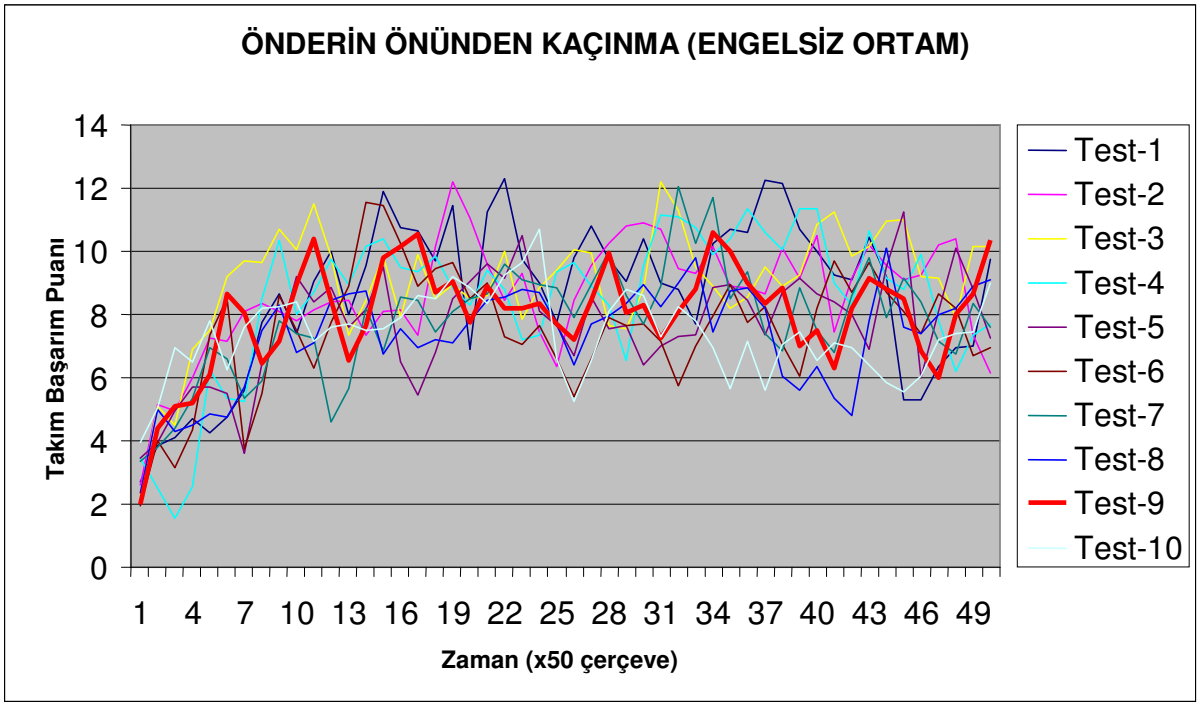
Şekil 6.5 Takım başarım puanı - engelsiz ortam, önderi takip



Şekil 6.6 Takım başarım puanı - engelsiz ortam, rastgele önder değişimi



Şekil 6.7 Takım başarım puanı - engelsiz ortam, en uzak önder seçimi

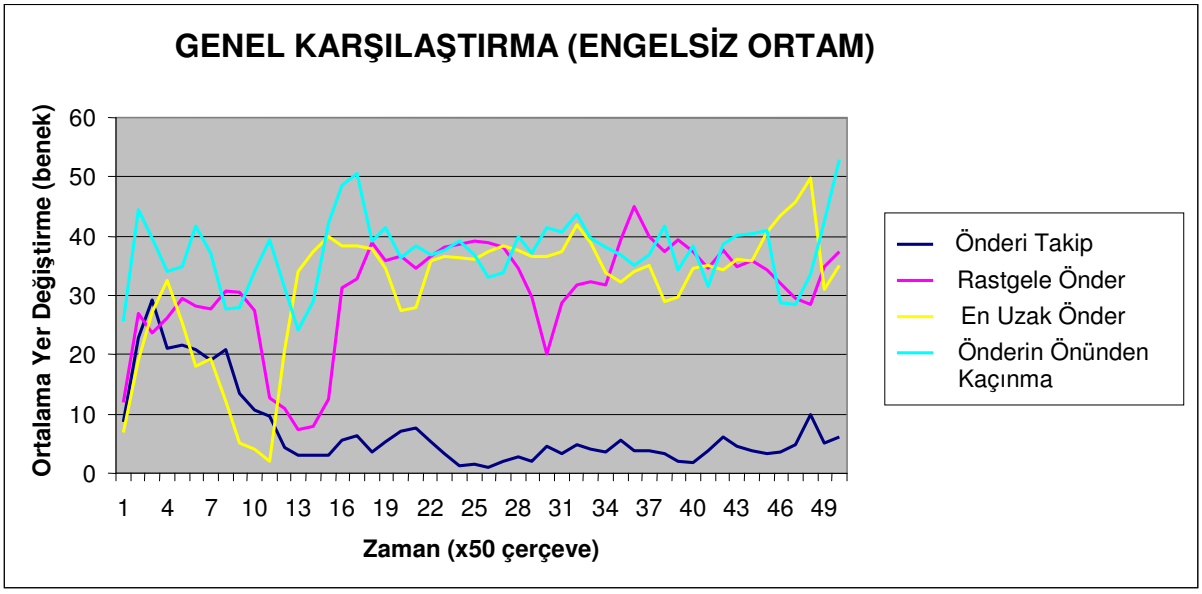


Şekil 6.8 Takım başarım puanı - engelsiz ortam, önderin önünden kaçınma

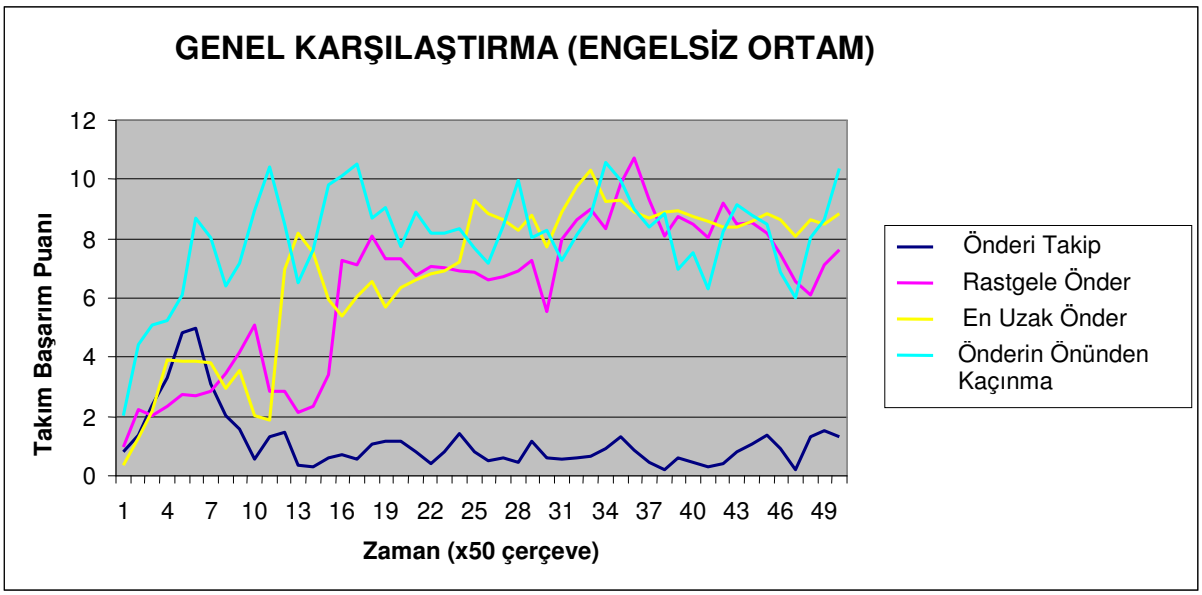
6.1.1. Yöntemlerin Karşılaştırılması (Engelsiz Ortam)

Engelsiz ortamda 4 farklı yöntem için gerçekleştirilen testlere ilişkin elde edilmiş olan temsilci eğrileri ortalama yer değiştirme değerleri için Şekil 6.9'da, takım başarım puanları için ise Şekil 6.10'da ortak grafikler üzerinde gösterilmiştir. Bu yöntemlerin takım ortalama yer değiştirme ve takım başarım puanı üzerindeki etkilerinin karşılaştırılması ve anlamlı farklar içerip içermediğinin belirlenmesi amacıyla hesaplanan değerler üzerinde varyans analizi yapılmıştır.

Varyans analizi, istatistikte bazı değişkenlerin veya farklı yöntemlerin bir başka değişken üzerindeki etkisini incelemek amacıyla yapılan işlemlere verilen genel isimdir. En yaygın kullanılan varyans analiz teknikleri arasında T-testi, F-testi, ANOVA (*Analysis of Variance*) sayılabilir. Bu tekniklerin hepsi parametrik istatistiksel yöntemler olup veri kümesinin normal dağılım göstermesi şartını gözetir. Veri kümesinin normal dağılım göstermediği durumlarda ise parametrik olmayan diğer testler kullanılabilmektedir. Mann-Whitney U testi, Kruskal-Wallis testi, Kolmogorov-Smirnov testi ve Anderson-Darling testi, parametrik olmayan varyans analiz testleri içinde yaygın olarak kullanılmaktadır.



Şekil 6.9 Genel Karşılaştırma : Takım ortalama yer değişimleri - engelsiz ortam



Şekil 6.10 Genel Karşılaştırma : Takım başarım puanı - engelsiz ortam

Gerçekleştirdiğimiz canlandırmalar sonucunda farklı yöntemlere ilişkin elde edilen değerlerin normal dağılım gösterip göstermediğini belirlemek için normallik sınaması yapmak gerekmiştir. Önderi takip yöntemi sonucunda oluşan tıkanmalara çözüm olarak önerilen 3 farklı yöntem arasında bir varyans analizi yapılacağından, normallik sınaması için bu 3 yöntemin (rastgele önder seçimi, en uzaktaki bireyin önder olarak seçimi, takım bireylerine önderin önünden kaçınma davranışının atanması) verileri kullanılmıştır. Normallik sınaması, SPSS yazılımı içinde yer alan Kolmogorov-Smirnov tek örneklem sınama yöntemi ile yapılmıştır. Sınamanın sonuçları, program çıktısı olarak Çizelge 6.3'de verilmiştir.

Çizelge 6.3 Normallik sınaması SPSS program çıktısı (engelsiz ortam)

One-Sample Kolmogorov-Smirnov Test

		OYD	TBP
N		150	150
Normal Parameters(a,b)	Mean	33,2104	7,0591
	Std. Deviation	9,16214	2,37596
Most Extreme Differences	Absolute	,160	,144
	Positive	,099	,090
	Negative	-,160	-,144
Kolmogorov-Smirnov Z		1,965	1,760
Asymp. Sig. (2-tailed)		**,001**	**,004**

a Test distribution is Normal.
b Calculated from data.

Çizelge 6.3'de görülen OYD sütunu, 3 farklı yönteme ilişkin toplam 150 veriden oluşan takım ortalama yer değiştirme değerlerine ilişkin; TBP sütunu ise takım başarım puanı değerlerine ilişkin normallik sınaması sonuçlarıdır. Son satırda görülen değerler genel olarak kabul gören **0.05** (%5) anlamlılık düzeyinin altında olduklarından, dağılımların normal olmadığı sonucuna varılmıştır. Diğer bir deyişle, verilerin normal dağılım gösteren bir kümeden olduklarını öne süren H_0 hipotezi red edilir.

Engelsiz ortam canlandırmaları sonucu elde ettiğimiz veriler normal dağılıma uymadıkları için, önerdiğimiz yöntemlere ilişkin varyans analizi, ANOVA'nın bir bakıma parametrik olmayan karşılığı olarak tanımlanabilen Kruskal-Wallis H testi ile yapılmıştır. Bu test ile normal dağılım göstermeyen en az 3 grubun ortalamaları arasındaki farklılığın anlamlılığı sınanabilmektedir. Test için yine SPSS programından yararlanılmıştır. Buna göre;

H_0 hipotezi : Rastgele önder seçimi, en uzak bireyin önder olarak seçimi ve takım bireylerine önderin önünden kaçınma davranışı atanması yöntemlerinin arasındaki fark anlamsızdır.

H_1 hipotezi : H_0 hipotezinde adı geçen yöntemler arasındaki fark anlamlıdır.

şeklinde tanımlayacağımız hipotezler için Kruskal-Wallis H testi sonucunda bulacağımız anlamlılık değerleri, uygulamalarda kabul gören p=0.05 değerinin altında ise H_0 hipotezi red edilir, H_1 hipotezi kabul edilir. Benzer şekilde anlamlılık değerinin 0.05'in üzerinde çıkması durumunda ise H_0 hipotezi kabul edilecektir.

Takım ortalama yer değiştirme ve takım başarım puanı verilerinin SPSS yazılımına girilip Kruskal-Wallis H testinin çalıştırılması sonucunda elde edilen sonuçlar Çizelge 6.4'de görülmektedir.

Çizelge 6.4 Kruskal-Wallis H testi SPSS program çıktısı (engelsiz ortam)

Ranks

	OYDSET	N	Mean Rank
OYD	1	50	61,93
	2	50	68,80
	3	50	95,77
	Total	150	
TBP	1	50	59,82
	2	50	75,79
	3	50	90,89
	Total	150	

Test Statistics[a,b]

	OYD	TBP
Chi-Square	16,951	12,790
df	2	2
Asymp. Sig.	,000	,002

a. Kruskal Wallis Test

b. Grouping Variable: OYDSET

Çizelge 6.4'ten de görülebileceği gibi, takım ortalama yer değiştirme (OYD) ve takım başarım puanı (TBP) değerleri açısından hesaplanan anlamlılık değerleri 0.05'in altında kalmıştır. Bu da H_0 hipotezinin red edilmesi anlamına gelir. Yani bu 3 yöntem arasındaki farkın anlamlı olduğu yorumu yapılabilir. Fakat bu farkın hangi veri setinden kaynaklandığı Kruskal-Wallis H testi ile belirlenemez. Yöntemleri ikili ikili karşılaştırmak ve fark yaratan veri setini bulabilmek için kullanılabilen diğer bir test, Mann-Whitney testidir. Bu test ile yapılan karşılaştırmalarda fark yaratan yöntemin, önderin önünden kaçınma yöntemi olduğu görülmüştür. Çünkü önderin önünden kaçınma yönteminde takımda tıkanıklık oluşmadığı için elde edilen test değerleri, diğer iki yönteme göre yüksek çıkmaktadır. Önder değişimini gerektiren iki yöntemi Mann-Whitney testi ile karşılaştırdığımızda ise Çizelge 6.5'deki sonuç elde edilmiştir.

Çizelge 6.5 Önder değişim yöntemleri için Mann-Whitney testi (engelsiz ortam)

Test Statistics[a]

	OYD	GBP
Mann-Whitney U	1144,000	997,000
Wilcoxon W	2419,000	2272,000
Z	-,731	-1,744
Asymp. Sig. (2-tailed)	,465	,081

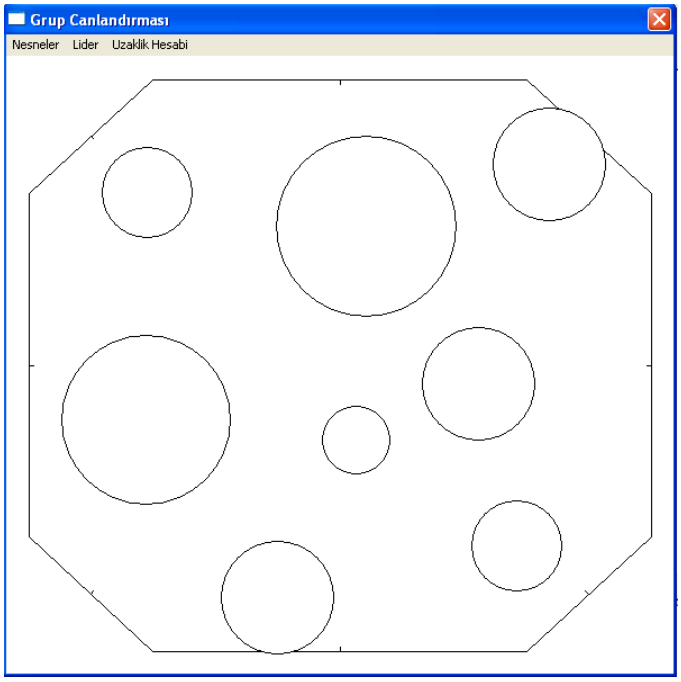
a. Grouping Variable: OYDSET

Çizelge 6.5'ten görüldüğü gibi anlamlılık değerleri 0.05'in üzerinde çıkmıştır. Bu da, engelsiz ortamlar için rastgele önder değişimi ve en uzaktaki bireyin önder olarak seçilmesi yöntemlerinin gerek takım ortalama yer değiştirme, gerekse takım başarım puanları açısından anlamlı bir farklılık taşımadığı anlamına gelir. Bu iki yöntem arasında yapılacak tercih, uygulamadan uygulamaya göre değişebilir. Rastgele önder seçimi, özellikle takımdaki birey sayısının yüksek olduğu durumlarda sistem başarımını düşürmeyen bir çözüm olabilir; fakat

rastsallıktan dolayı tıkanıklığın ilk değişimde çözülme garantisi yoktur. Öte yandan en uzak bireyin önder olarak seçimi, takım bireyleri arasında uzaklık hesaplamalarından dolayı sistem başarımında bir düşüşe neden olacaktır; fakat tıkanıklığın ilk değişimde çözülme olasılığı özellikle engelsiz ortamlarda oldukça yüksektir. Ortam engelsiz olduğundan her iki yöntemde tıkanıklık çözüldükten sonra ikinci bir tıkanma oluşma olasılığı oldukça düşüktür.

6.2. Engelli Ortam Canlandırmaları

Bölüm 6.1'de gerçekleştirilen canlandırmaların benzerleri, Şekil 6.11'de görülen ve dairesel engellerin yer aldığı bir canlandırma ortamı üzerinde tekrarlanmıştır. Engellerin kapladığı alanın toplam alana oranı yaklaşık olarak %35'tir.

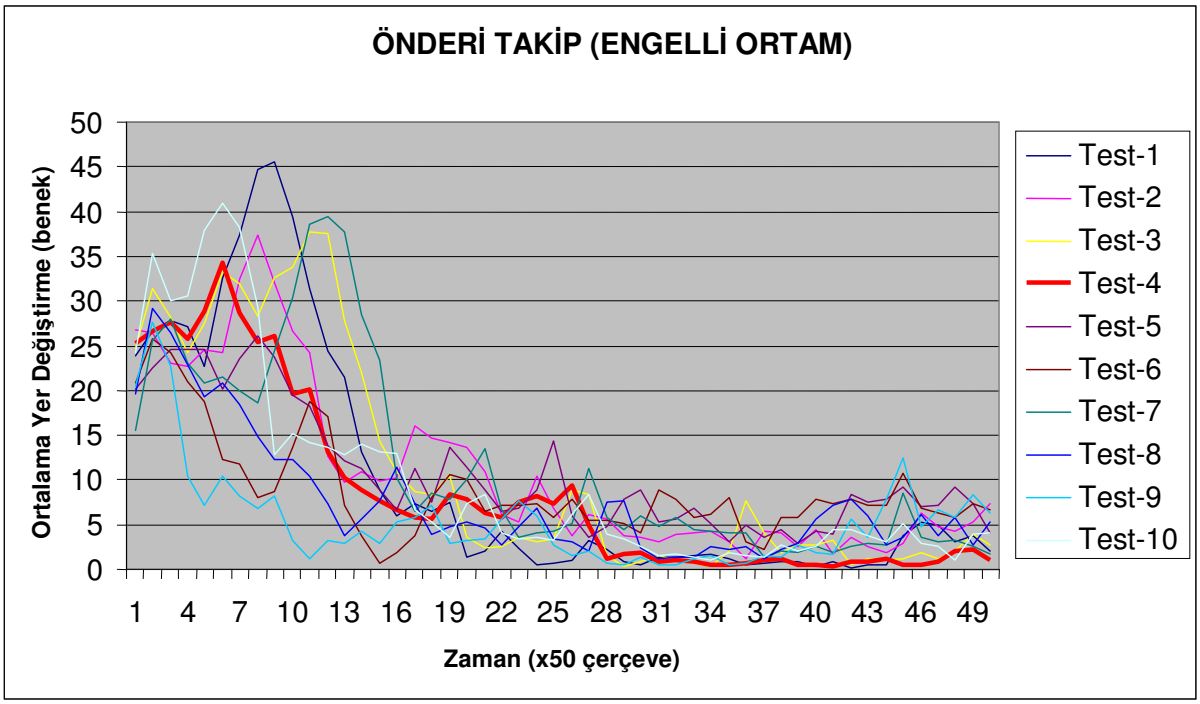


Şekil 6.11 Canlandırmaların gerçekleştirildiği engelli ortam

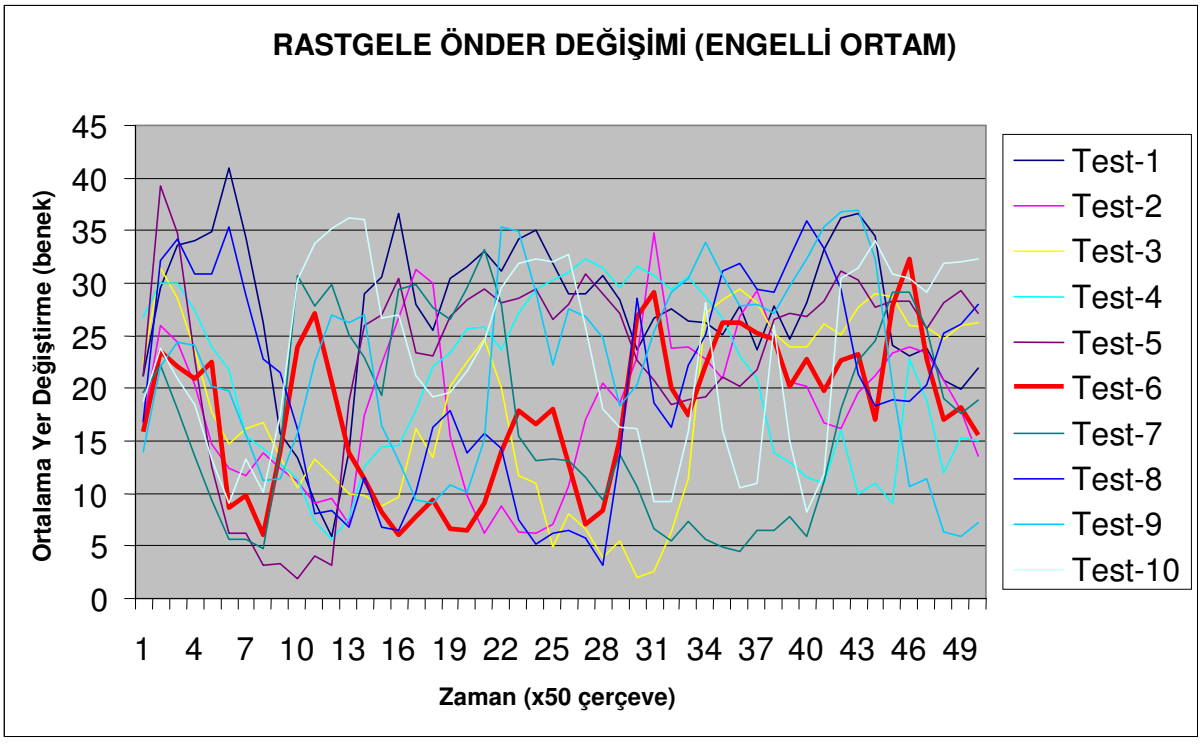
Engelli ortamda yapılan canlandırmaların büyük bölümünde gözlenen ortak bir durum, önder etrafındaki ilk gruplaşmanın engelsiz ortama göre daha geç oluşmasıdır. Bu durum, Şekil 6.12'de görülebilmektedir. Öndere yönelen etmenlerin ilk gruplaşması ve takımın bir tıkanmaya girmesi Şekil 6.1'de görülen engelsiz ortam canlandırmalarına göre daha geç olmuştur. Grafiklerde koyu renk ile belirlenen eğri, ortalama eğri ile olan uzaklıkların kareleri toplamının minimum olduğu canlandırmaya aittir ve temsilci eğri olarak belirlenmiştir.

Tıkanıklık kontrolünün yapıldığı ve böyle bir durumda rastgele önder seçiminin yürütüldüğü canlandırmalara ilişkin eğriler Şekil 6.13'de; öndere en uzak bireyin yeni önder olarak

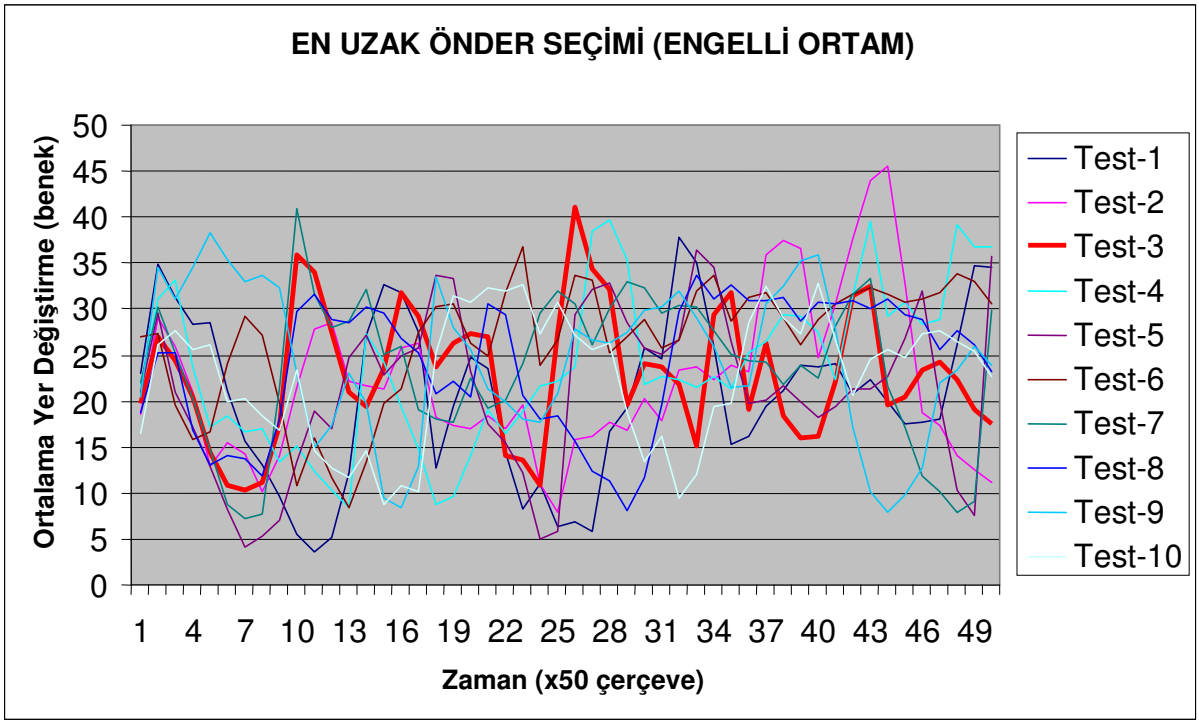
seçildiği canlandırmalar Şekil 6.14'de ve takım bireylerinin önderin önünden kaçınma algoritmasını yürüttüğü canlandırmalara ilişkin eğriler ise Şekil 6.15'de verilmiştir.



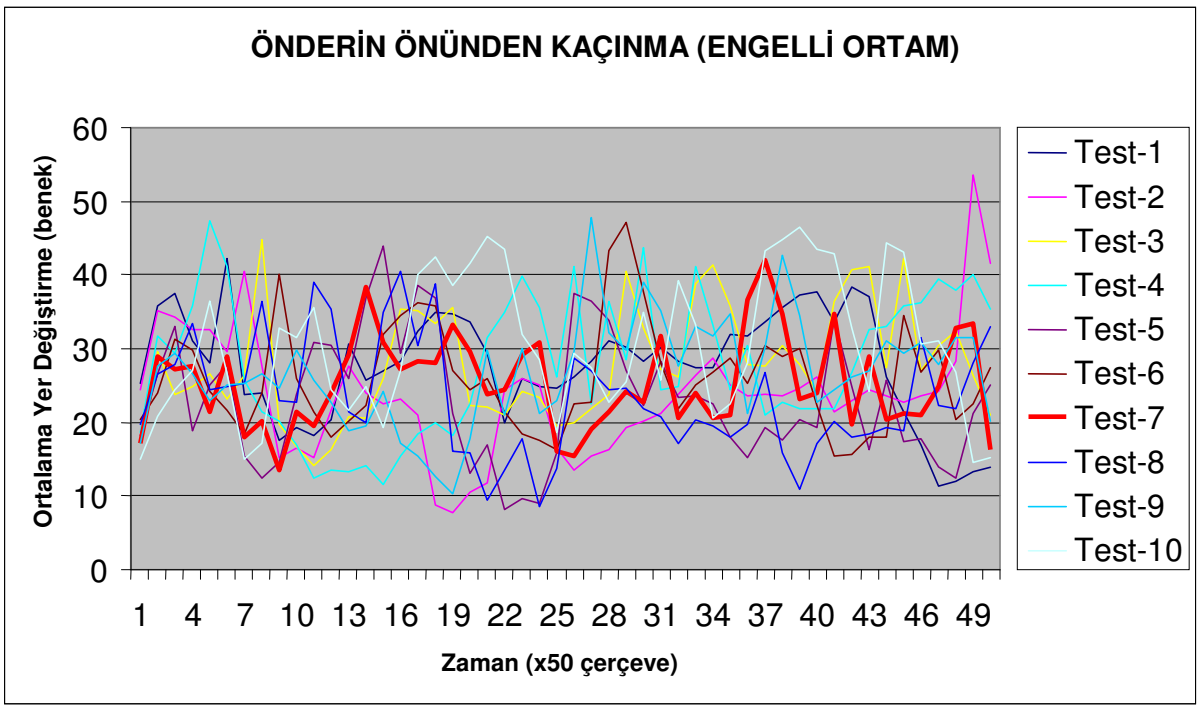
Şekil 6.12 Takım ortalama yer değişimleri - engelli ortam, önderi takip



Şekil 6.13 Takım ortalama yer değişimleri - engelli ortam, rastgele önder değişimi



Şekil 6.14 Takım ortalama yer değişimleri - engelli ortam, en uzak önder seçimi



Şekil 6.15 Takım ortalama yer değişimleri - engelli ortam, önderin önünden kaçınma

Engelli ortamda gerçekleştirilen ve takım bireylerine önderin önünden kaçınma davranışının atandığı duruma ilişkin canlandırmalarda da sistemin bazen önder değişimine gittiği gözlenmiştir. Önderin önünden kaçınmak isteyen bireylerin kaçma yönleri bir engel veya duvar ile kapandığında bu bireyler hareketsiz kalmış ve takımın ortalama yer değiştirme

değeri kritik seviyenin altına düştüğünden sistem önder değişimini tetiklemiştir. Engelli ortamdaki canlandırmalarda ortaya çıkan önder değişim sayıları Çizelge 6.6'da verilmiştir.

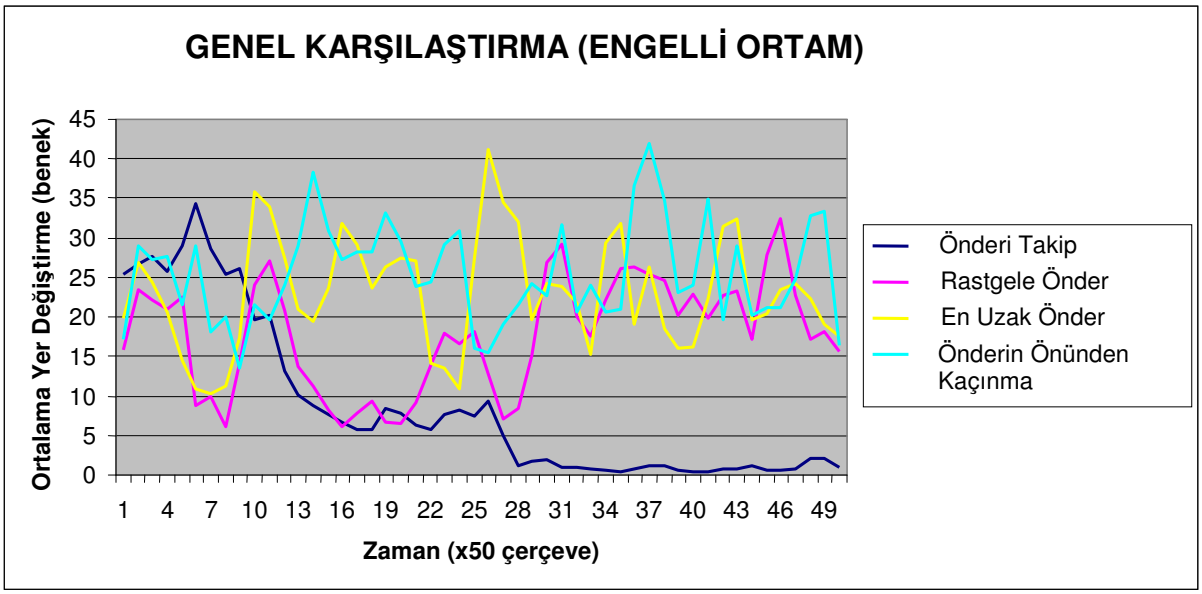
Çizelge 6.6 Önder değişim sayıları (engelli ortam)

	Test1	**Test2**	**Test3**	**Test4**	**Test5**	**Test6**	**Test7**	**Test8**	**Test9**	**Test10**	**Ort.**
Rastgele Önder	1	3	5	4	2	3	5	4	3	4	3.4
En Uzak Önder	2	2	2	1	3	1	2	2	2	2	1.9
Önderin Önünden Kaçınma	0	1	0	0	1	0	0	1	0	0	0.3

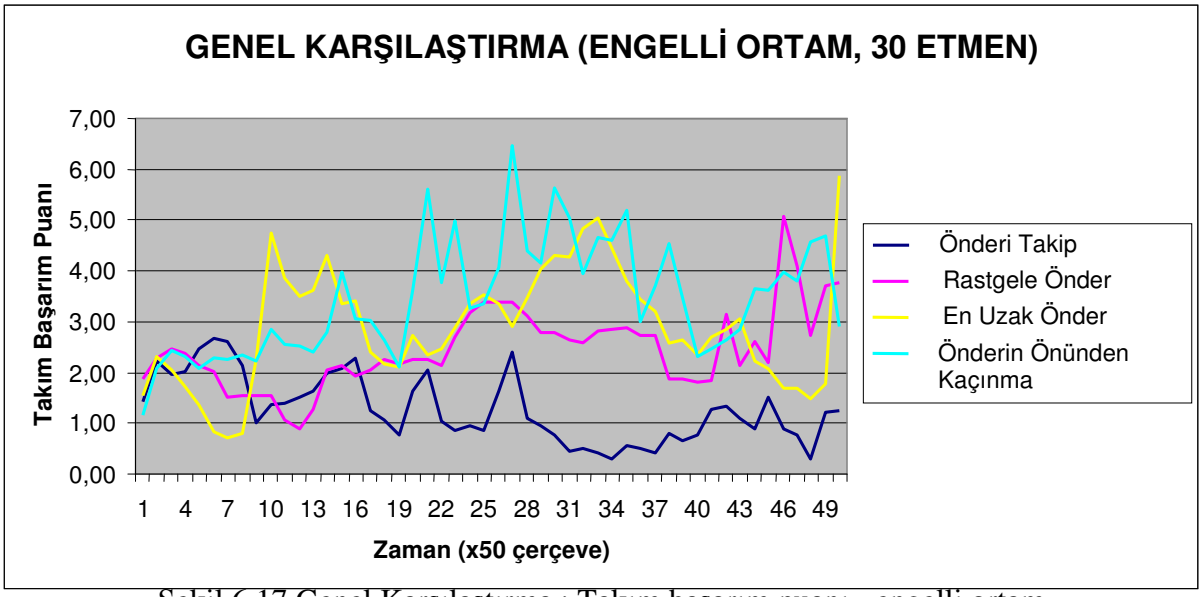
Engelli ortamda gerçekleştirilen canlandırmalarda takım başarım puanları da ölçülmüştür ve 4 farklı yönteme ait testler arasından seçilen temsilci eğriler benzer yöntemlerle belirlenmiştir. Bu eğriler Şekil 6.17'de görülmektedir.

6.2.1. Yöntemlerin Karşılaştırılması (Engelli Ortam)

Bir önceki bölümde elde edilen temsilci eğrilerin aynı grafik üzerinde gösterimi, ortalama takım yer değişimi değerleri için Şekil 6.16'da, takım başarım puanı değerleri için Şekil 6.17'de verilmiştir. Gerçekleştirilen engelli ortam canlandırmalarına ait temsilci eğriler arasında varyans analizi yapabilmek için, dağılımların normal olup olmadığı araştırılmıştır. Bunun için engelsiz ortam değerlendirmelerinin yapıldığı Bölüm 6.1.1'de de kullandığımız SPSS programı ile Kolmogorov-Smirnov tek örneklem sınama yöntemi seçilmiştir.



Şekil 6.16 Genel Karşılaştırma : Takım ortalama yer değişimleri - engelli ortam



Şekil 6.17 Genel Karşılaştırma : Takım başarım puanı - engelli ortam

Engelli ortam canlandırmalarına ilişkin normallik sınaması sonuçları Çizelge 6.7'de görülmektedir. OYD sütunu takım ortalama yer değişim değerleri, TBP sütunu ise takım başarım puanı değerleri için olan sonuçları içermektedir. Her iki veri kümesi için bulunan anlamlılık düzeyi kabul edilebilir değerlerde olduğu için (>0.05) bu verilerin normal dağılıma uygun olduğu sonucu ortaya çıkmıştır.

Çizelge 6.7 Normallik sınaması SPSS program çıktısı (engelli ortam)

One-Sample Kolmogorov-Smirnov Test

		OYD	TBP
N		150	150
Normal Parameters(a,b)	Mean	22,0216	2,9245
	Std. Deviation	7,58242	1,10920
Most Extreme Differences	Absolute	,043	,100
	Positive	,041	,100
	Negative	-,043	-,050
Kolmogorov-Smirnov Z		,527	1,220
Asymp. Sig. (2-tailed)		**,944**	**,102**

a Test distribution is Normal.
b Calculated from data.

Normal dağılım gösteren ikiden fazla grubun varyans analizi için yaygın olarak kullanılan test tek yönlü ANOVA testidir. Buradaki tek yön ifadesi, karşılaştırılacak grupları (yani tıkanıklığa karşı önerilen 3 farklı çözüm yöntemini) birbirinden ayıran tek özellik (takım ortalama yer değiştirme değerleri) olduğu anlamına gelir. Bu test için ele aldığımız hipotezler:

H_0 : Yöntemler arasında bir fark yoktur.

H_1 : En az iki yöntem arasında anlamlı bir fark vardır.

şeklindedir. SPSS programı yardımı ile gerçekleştirilen tek yönlü ANOVA test sonuçları Çizelge 6.8'de verilmiştir.

Çizelge 6.8 Karşılaştırılan yöntemler için ANOVA sonuçları (engelli ortam)

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
OYD	Between Groups	1594,566	2	797,283	16,810	,000
	Within Groups	6971,907	147	47,428		
	Total	8566,473	149			
TBP	Between Groups	24,672	2	12,336	11,430	,000
	Within Groups	158,646	147	1,079		
	Total	183,318	149			

Hem takım ortalama yer değiştirme hem de takım başarım puanı değerleri açısından bulunan anlamlılık seviyesi kabul edilen seviyenin altında kaldığından H_0 hipotezi red edilir. Bu da, karşılaştırılan yöntemlerin en azından ikisi arasında anlamlı bir fark olduğu sonucunu ortaya çıkarır. ANOVA testi hangi grupların farklılık gösterdiği konusunda bir bilgi vermediğinden, bunu öğrenebilmek amacıyla uygun bir ANOVA sonrası test seçebilmek için öncelikle varyansların homojenlik testi (Levene Testi) uygulanmıştır.

Çizelge 6.9 Varyans homojenlik testi

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
OYD	,393	2	147	,676
TBP	5,189	2	147	,007

Çizelge 6.9'dan da görülebildiği gibi anlamlılık seviyeleri kabul edilebilir değerin (0.05) üzerinde kalmıştır. Bu da varyansların homojen dağıldığı sonucunu ortaya çıkarır. SPSS programında homojen dağılımlı varyanslar için kullanılabilecek pek çok ANOVA sonrası test bulunmaktadır. Bunların arasından yaygın olarak kullanılan Tukey testi seçilmiş ve önerilen yöntemler arasındaki ikili karşılaştırmalar yapılmıştır. Sonuçlar Çizelge 6.10'da verilmiştir.

Çizelge 6.10 Yöntemler arası ikili karşılaştırmalar

Multiple Comparisons

Tukey HSD

Dependent Variable	(I) OYDSET	(J) OYDSET	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
OYD	1	2	-5,37800 *	1,37736	,000	-8,6392	-2,1168
		3	-7,80220 *	1,37736	,000	-11,0634	-4,5410
	2	1	5,37800 *	1,37736	,000	2,1168	8,6392
		3	-2,42420	1,37736	,187	-5,6854	,8370
	3	1	7,80220 *	1,37736	,000	4,5410	11,0634
		2	2,42420	1,37736	,187	-,8370	5,6854
TBP	1	2	-,43820	,20777	,091	-,9301	,0537
		3	-,99120 *	,20777	,000	-1,4831	-,4993
	2	1	,43820	,20777	,091	-,0537	,9301
		3	-,55300 *	,20777	,023	-1,0449	-,0611
	3	1	,99120 *	,20777	,000	,4993	1,4831
		2	,55300 *	,20777	,023	,0611	1,0449

*. The mean difference is significant at the .05 level.

Çizelge 6.10'da 1 numaralı grup rastgele önder değişim yöntemini, 2 numaralı grup en uzaktaki bireyin önder olarak seçilme yöntemini ve 3 numaralı grup ise bireylere önderin önünden kaçınma algoritmasının atandığı yöntemi temsil etmektedir. 2. sütunda * ile işaretlenmiş satırlar, o sütunda karşılaştırılan yöntemler arasında anlamlı bir farklılığın bulunduğunu belirtir. Buna göre Çizelge 6.10'da elde edilmiş sonuçların bir özeti Çizelge 6.11'de verilmiştir.

Çizelge 6.11 Engelli ortamda tıkanıklık çözüm yöntemlerine ilişkin varyans analizi sonuçları

	Takım Ortalama Yer Değişimi			Takım Başarım Puanı		
Yöntem	Rastgele Önder	En Uzak Önder	Önder Önünden K.	Rastgele Önder	En Uzak Önder	Önder Önünden K.
Rastgele Önder		FARK VAR	FARK VAR		FARK YOK	FARK VAR
En Uzak Önder	FARK VAR		FARK YOK	FARK YOK		FARK VAR
Önderin Önünden K.	FARK VAR	FARK YOK		FARK VAR	FARK VAR	

Çizelge 6.11'den de görülebileceği gibi engelli ortamlarda takım ortalama yer değişimi için rastgele önder çözümü, diğer iki yönteme göre fark üretmiştir. Bunun en önemli nedeni Çizelge 6.6'dan görülebildiği gibi ortaya çıkan fazla sayıdaki önder değişimidir. Takımdaki tıkanıklığı çözemeyen her önder değişimi, takım ortalama yer değişim değerinin düşük kalmasına neden olmaktadır.

Engelli ortamda gerçekleşen canlandırmalara takım başarım puanı açısından bakıldığında ise önder değişimi gerektiren iki yöntem arasında belirgin bir fark olmadığı; buna karşın önderin

önünden kaçınma algoritmasının yürütüldüğü durumlarda takım başarımının daha yüksek çıktığı gözlenmiştir.

6.3. Önemli Parametrelerin Duyarlılık Analizleri

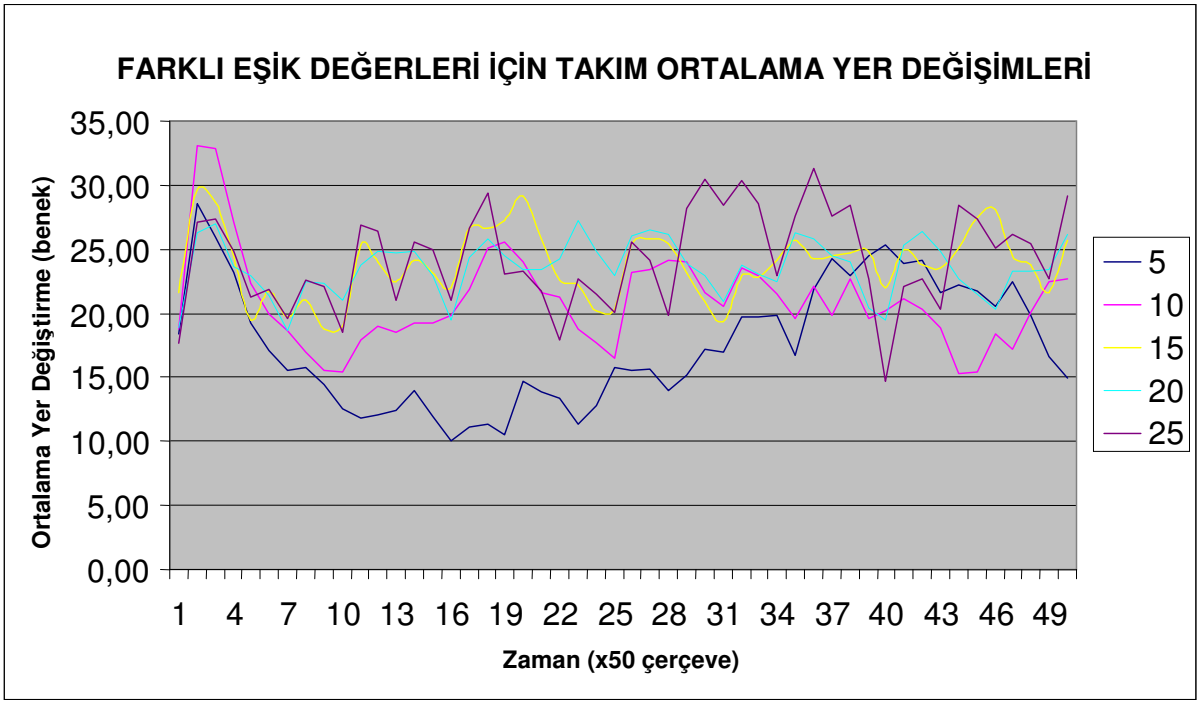
Bu bölümde, takım canlandırmalarında oluşabilecek tıkanma durumları ile doğrudan ilişkili iki önemli parametrenin gerek takım başarım puanı, gerekse takımın ortalama yer değiştirme değerleri üzerinde yapılan duyarlılık analizlerinin sonuçları verilmiştir. Bu parametreler, Bölüm 4.1'te bahsedilmiş olan ve bir takımın tıkanmaya girme sınırını belirleyen eşik değeri ile bir önder değişimi sonrası seçilen yeni önderin tıkanıklığı açması için bu öndere tanınan süreye ilişkin parametrelerdir. Ayrıca her iki parametrenin, tıkanıklığın çözülebildiği toplam önder değişim sayısı üzerinde de doğrudan etkisi bulunmaktadır.

6.3.1. Tıkanma eşik değeri

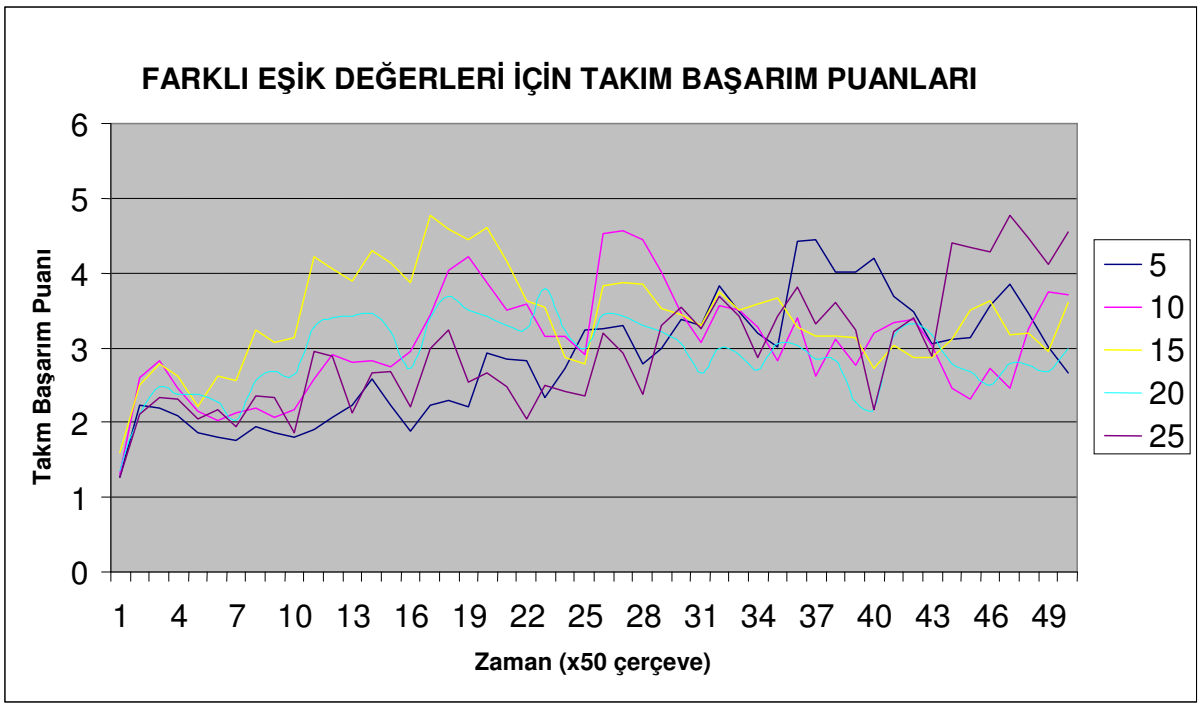
Tıkanma eşik değeri, bir takımın tıkanmaya girme sınırı olarak belirlenen minimum takım ortalama yer değişim değeridir. Diğer bir deyişle, bir takımın önder değişimi yapmadan izin verilebilen en düşük ortalama yol alma değeridir. Bu değer, takımdaki bireylerin yapabileceği en yüksek hız ile doğrudan ilişkilidir. Çünkü çok yavaş hareket eden bir takımın (örneğin bir fil sürüsü) tıkanma eşik değeri, kuşkusuz yüksek hızlarda hareket eden bir takımın tıkanma eşik değerine göre daha düşük olacaktır.

Tıkanma eşik değerinin takım ortalama yer değişimi ve takım başarım puanı üzerindeki etkisinin belirlenebilmesi için Bölüm 6'da detayları verilmiş olan engelli canlandırma ortamında bir dizi test canlandırması gerçekleştirilmiştir. Uygulamada 10 olarak seçilmiş olan tıkanma eşik değerinin 5, 10, 15, 20 ve 25 değerleri için 10'ar adet test gerçekleştirilmiş ve her bir test grubunu belirleyen temsilci eğriler oluşturulmuştur. Bu testler sonucunda elde edilen takım ortalama yer değişim eğrileri Şekil 6.18'de; takım başarım puanı eğrileri ise Şekil 6.19'da görülmektedir.

Her iki eğri grubundaki veriler, Bölüm 6.1 ve Bölüm 6.2'de gerçekleştirilen istatistiksel testlere tabi tutulmuştur. Öncelikle dağılımların normalliği sınanmış ve bu verilerin normal dağılım oluşturduğu sonucuna varılmıştır. Ardından gerçekleştirilen ANOVA testleriyle eğrilerin anlamlı farklılıklar içerdiği (diğer bir deyişle farklı eşik değerlerinin takım ortalama yer değişimi ve takım başarım puanı üzerinde anlamlı farklarının olduğu) görülmüştür.

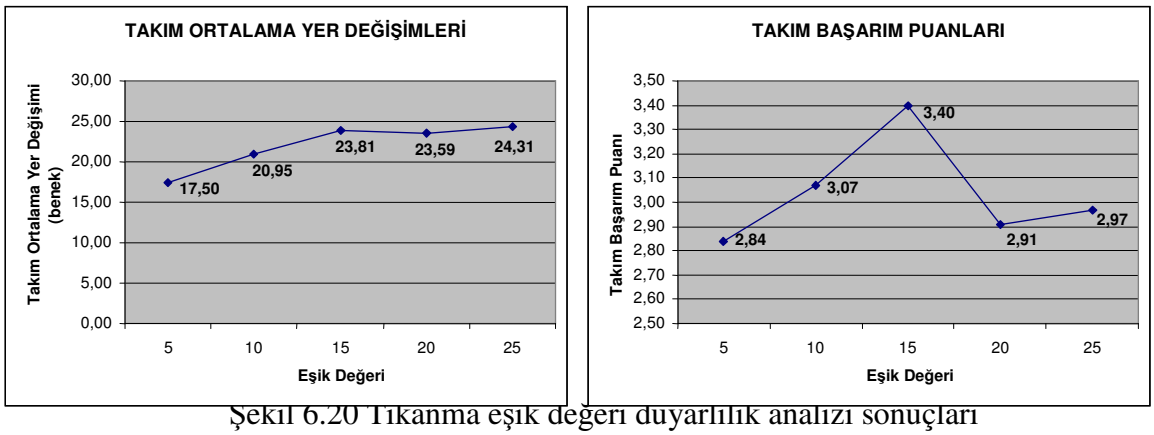


Şekil 6.18 Tıkanma eşik değeri duyarlılık analizi (takım ortalama yer değişimleri)



Şekil 6.19 Tıkanma eşik değeri duyarlılık analizi (takım başarım puanı)

Testlerde seçilen her bir eşik değeri için gerçekleştirilen 10 adet canlandırma sonucunda elde edilen değerlerin ortalamalarına bakıldığında, bu eşik değerlerinin takım ortalama yer değişimi ve takım başarım puanını ne şekilde etkiledikleri görülebilir. Bu grafikler Şekil 6.20'de verilmiştir.

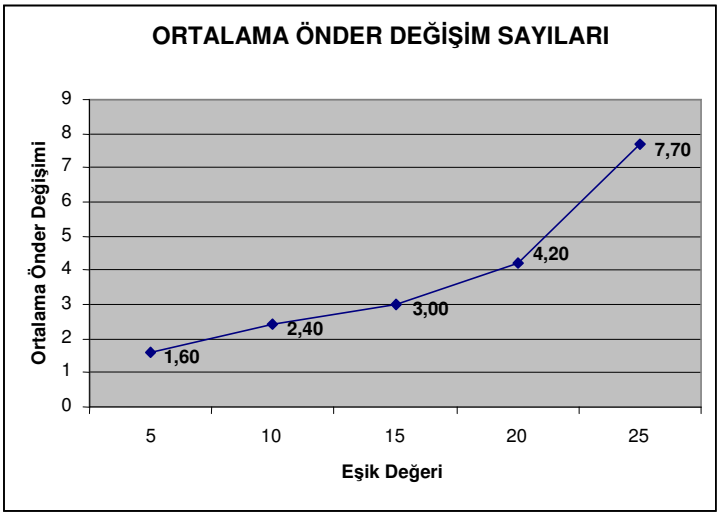


Şekil 6.20 Tıkanma eşik değeri duyarlılık analizi sonuçları

Düşük seçilen bir eşik değeri, takımın düşük hızlarda hareketine izin verdiğinden takım ortalama yer değişimi değeri de düşük çıkmaktadır. Yüksek seçilen bir eşik değeri ise takımın ortalama yer değişiminin çok düşük değerlere inmesini engeller. Öte yandan Şekil 6.20'deki ilk grafikte görüldüğü gibi belirli bir eşik değerinden sonra takımın ortalama yer değişimi fazla etkilenmemektedir. Bunun nedeni, ortamdaki engeller ve takım hareketinin bir sonucu olarak zaten takım ortalama yer değişiminin belirli bir değerin üstüne çıkamaması olarak gösterilebilir.

Tıkanma eşik değeri ile takım başarım puanı arasındaki ilişki Şekil 6.20'deki ikinci grafikte görülebilmektedir. Eşik değerinin yüksek olması, her zaman için yüksek bir takım başarım puanı çıkması anlamına gelmez. Bunun nedenleri; yüksek hızlarda tetiklenen fazla sayıdaki önder değişimi sonucunda takımın ani yön değişimleri, kısa zaman dilimlerinde ve fazla sayıda oluşan bu önder değişimleri sürecinde takımın hızlı bir şekilde toparlanamaması ve bunun bir sonucu olarak takımda oluşan parçalanmalardır.

Tıkanma eşik değerinin yüksek seçilmesi, takımın yüksek hızlarda da önder değişimine gitmesini gerektirir; bu da düşük seçilen bir eşik değerindekinden daha fazla sayıda önder değişimi olacağı anlamına gelir. Farklı eşik değerlerine göre oluşan ortalama önder değişimleri Şekil 6.21'deki grafikte görülebilmektedir. Tıkanma eşik değerinin çok yüksek seçildiği durumlarda sistem sürekli olarak önder değişimini tetikler; bunun sonucu olarak takım bireyleri her önder değişiminde yeni öndere yönlenirken ortalama yer değiştirme değerleri anlık olarak düşük seviyelerde kalır. Eğer öndere tıkanıklığı çözmesi için tanınan süre yeteri kadar uzun değilse, bu sürenin ardından sistem tekrar yeni bir önder değişimine gider.

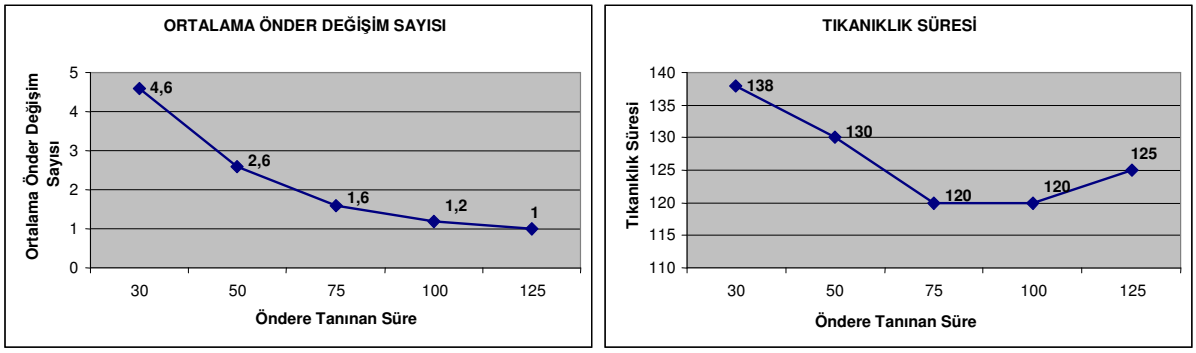


Şekil 6.21 Tıkanma eşik değeri için önder değişim sayıları

6.3.2. Yeni öndere tanınan süre

Bir önder değişiminin ardından yeni öndere tıkanıklığı çözebilmesi için tanınan süre, tıkanma anında yapılacak önder değişimi sayısı ile doğrudan ilişkilidir. Bu sürenin çok kısa seçilmesi, çok fazla sayıda önder değişiminin tetiklenmesine; sürenin uzun seçilmesi ise yeni önder denetimindeki takımın tekrar tıkanıklığa girdiği durumlarda bunun sezilememesine neden olacaktır.

Bir önceki bölümde de görüldüğü gibi sistemde fazla sayıda yapılan önder değişimi, takım başarım puanını olumsuz etkilemektedir. Öte yandan tıkanıklığı çözebilmesi için öndere tanınan süre çok uzun seçilir ve bu süre içinde tıkanıklık çözülemezse, takım başarım puanı yine düşük çıkacaktır. Bu sebeple, yeni öndere tanınan sürenin duyarlılık analizi yapılırken ortalama önder değişim sayıları ve tıkanıklıkta geçen maksimum süre incelenmiştir. Öndere tanınan sürenin 30, 50, 75, 100 ve 125 çerçeve olması durumları için engelli ortamda 10'ar adet canlandırma gerçeklenmiş ve her bir farklı değer için canlandırmalardaki ilk tıkanıklık durumunda ortaya çıkan önder değişim sayılarının ortalamaları alınmıştır. Engelli ortam parametreleri olarak Bölüm 6'da verilen parametreler seçilmiştir. Testlerde tıkanıklık eşik değeri 10 benek olarak alınmış ve önder değişim algoritması olarak en uzak bireyin yeni önder olarak belirlendiği algoritma yürütülmüştür. İlk tıkanıklık çözülene kadar yapılan ortalama önder değişim sayıları ve tıkanıklıkta geçen maksimum süreler Şekil 6.22'de görülmektedir.



Şekil 6.22 Öndere tanınan süreye ilişkin duyarlılık analizi sonuçları

Şekil 6.22'deki ilk grafikten de görüldüğü gibi, tıkanıklık durumunda yeni öndere tanınan sürenin artması, ortalama önder değişim sayısında azalmaya neden olur. Öte yandan az sayıda önder değişimi her zaman için tıkanıklığın daha kısa sürede çözülmesi anlamına gelmemektedir. Şekil 6.22'deki ikinci grafik, ilk grafikteki değerlerin öndere tanınan süre ile çarpılması sonucunda ortaya çıkan grafik olup, tıkanıklık süresince geçen süreler için üst limitleri göstermektedir. Örnek olarak, öndere tanınan sürenin 30 çerçeve seçildiği 10 test canlandırması için hesaplanan ortalama önder değişim sayısı 4.6 olarak bulunmuştur. Bu, en kötü ihtimalle 4.6 adet önder değişimi süresince tıkanıklığın sürdüğü anlamına gelir. İki önder değişimi arasında 30 çerçeve süre geçtiği için 4.6 adet önder değişimi 4.6x30 = 138 çerçeve sürede gerçekleşmektedir. Şekil 6.21'deki ikinci grafik incelenirse, bu sistemde öndere tanınan süre için en optimal değerin 75 veya 100 çerçeve olduğu görülebilir.

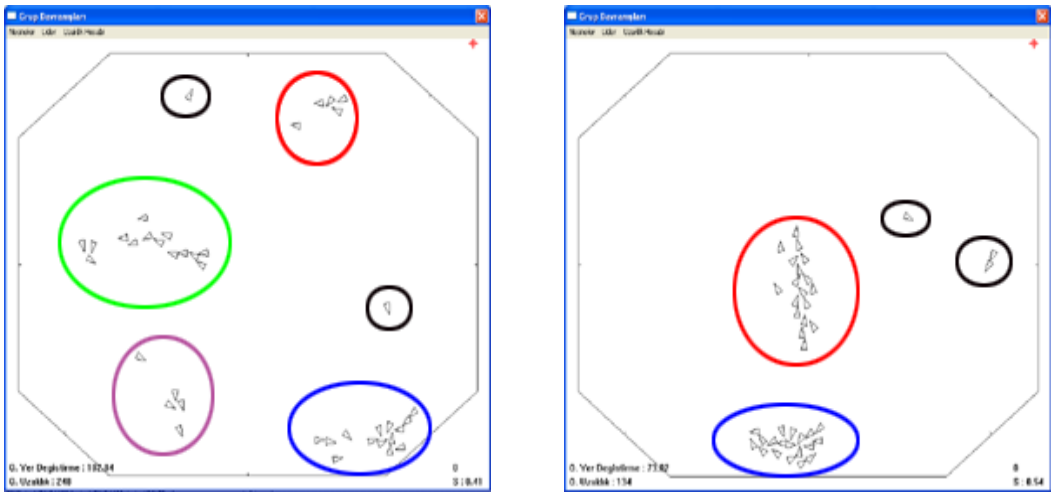
7. TAKIM CANLANDIRMA SİSTEMİNİN BİR UYGULAMASI OLARAK ÇOK TAKIMLI SİSTEMLER

Bu çalışmada sunulan takım canlandırma sisteminin uygulamalarında gözlemlenen bir sonuç, bireylerin başlangıç konumları rastgele seçildiği için bu bireylerin tek bir önder peşinde tek bir takım oluşturmalarının belirli bir süre almasıdır. Çünkü hareketlerine önderden uzakta başlayıp rastgele gezinme (*wandering*) davranışı sergileyen bireylerin ana takıma dahil olmaları, ancak takımdaki bireylerin algı mesafelerine girmelerinin ardından mümkün olabilmektedir. Ayrıca canlandırma ortamı genişledikçe ve ortamdaki engellerin sayısı arttıkça bu süre daha da uzamaktadır. Oysa takım önderinden uzakta bulunan bireylerin küçük takımlar oluşturmaları, hatta bu takımlar içindeki bazı bireyleri kendilerine önder olarak seçmeleri gerçek yaşamda da sıkça rastlanan bir olay olduğundan önerilen sistemde de bu konu üzerinde çalışılması düşünülmüştür.

Gerçek yaşamda takım davranışı sergileyen canlıların zaman zaman farklı takımlar oluşturdukları, bu takımların birleşip daha büyük takımlara dönüştükleri ve zaman zaman da tekrar bölündükleri gözlenebilmektedir. Bu süreç dinamik olup belirli bir andaki takım ve bu takımlardaki birey sayıları sürü davranışının gelişimine göre değişebilmektedir.

Çalışmamızda bireylerin takımlara ayrılması işlemi bölüm 7.1'de ele alınan "Takım Demetleme Algoritması" uyarınca gerçeklenir. Bu aşamada, takım bireylerini rastgele seçerek yeni takımlar oluşturmak gerçekçi bir uygulama olamaz; çünkü bu durumda alt takımların iç içe geçmesi canlandırmada istenmeyen sonuçların ortaya çıkmasına neden olur. Oysa ki takımın bölünmesi gerçek yaşamda olduğu gibi dış etkenlerin zorlaması ile gerçekleşir. Bu noktadaki temel problemler, bölünme anının belirlenmesi ve dış etkenlerin dağıttığı bireylerin alt takımlara dahil olmasının sağlanmasıdır.

Örnek olarak 40 etmen içeren bir canlandırmanın iki farklı anına ilişkin ekran görüntüleri Şekil 7.1'de gösterilmektedir. Şekil 7.1.a'da 6 farklı takım oluşumu gözlenirken, ilerleyen evrelerde bu takımlardan bazıları karşılaşmış, takımlar arasında birleşmeler yaşanmış ve Şekil 7.1.b'de de görülebileceği gibi toplam takım sayısı 4'e inmiştir.



(a) (b)

Şekil 7.1 Örnek bir canlandırmanın farklı anlarında oluşan takımlar

Bir canlandırma ortamında bireylerin başlangıç konumları rastgele seçildiği için, canlandırmanın ilk adımında oluşacak takım sayısını bireylerin varsa türleri ve birbirlerine olan uzaklıkları belirleyecektir. Eğer bireyler özdeş türde ise, birbirlerini algı bölgeleri içinde gören bireyler aynı takıma dahil olur ve canlandırma ilerledikçe karşılaşan bu takımlar birleşme eğilimi gösterir.

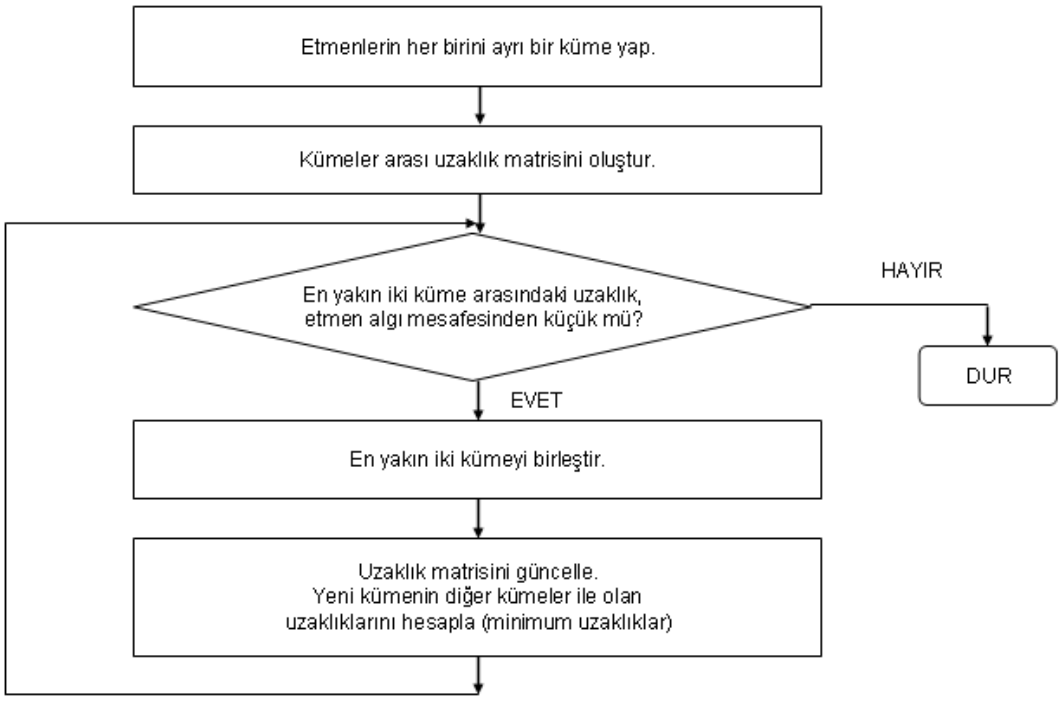
Öte yandan canlandırma ortamında farklı türden bireyler yer alıyorsa gruplaşma ancak aynı türe ait bireyler arasında olacaktır. Birbirleri ile karşılaşan farklı türdeki bireyler birleşme eğilimi göstermez; diğer bir deyişle farklı türdeki bireyler tarafından oluşturulmuş takımlar türe bağımlı olarak hareket eder. Bağımsız hareket eden takımlar konusu Bölüm 7.2'de ele alınmıştır.

Çoklu takımların söz konusu olduğu bir ortamda takımların bölünmeleri söz konusu olabilir. Bu durumda bu bölünmelerin sezilerek yeni takımlar oluşturulmasının sağlanması gerekecektir. Ana takımdan ayrılan bireylerin sayısı bir veya daha fazla olabilir. Bireyleri aynı türe ait olan farklı takımlar karşılaştığında, bu takımlar birleşme eğilimi içine girer. İki takımın birleşme işlemi, takım önderlerinin karşılıklı algı bölgelerinin içine girmesi ile başlar. Birleşme sonucu oluşan takımın önderinin hangi önder olacağı ise kurallarla belirlenir. Örneğin çokluğa dayalı birleşme kuralı, nüfusu az olan takımın daha kalabalık olan takıma katılması şeklinde tanımlanmıştır. Aynı türden takımların etkileşimi konusu Bölüm 7.3'de incelenmiştir.

7.1 Takım Demetleme Algoritması

Ortamda rastgele konumlarda yerleştirilen bireylerden birden fazla sayıda takım oluşturmak için; önce bireyleri kümelere ayırmak ve sonra da her bir küme içinden bir bireyi o kümenin önderi olarak atamak gerekir. Benzetim ortamlarında birden fazla kümenin oluşturulabilmesi, demetleme (*clustering*) yöntemleri kullanılarak sağlanır.

Literatürdeki temel demetleme algoritmaları, küme sayısının önceden bilinip bilinmemesine göre iki gruba ayrılmıştır. Yani temel demetleme algoritmaları küme sayısını veya kümeleme yaparken durma ölçütünü parametre olarak alır; önerdiğimiz sistemde ise bu parametrelerin varlığı veya yokluğu uygulamaya göre değişebilmektedir. Çalışmamızda takımların birleşmesi ve parçalanması dinamik olduğundan, küme sayısı önceden bilinmemektedir. Dolayısıyla küme sayısının parametre olarak alındığı demetleme algoritmaları (örneğin en yaygın olarak kullanılan k-ortalamalar algoritması (MacQueen, 1967)) önerdiğimiz sistem için kullanışsızdır. Küme sayısının önceden bilinmesini gerektirmeyen algoritmalardan biri olan hiyerarşik demetleme (*hierarchical clustering*) ile sistemimizin gerektirdiği takım demetleme problemi çözülebilir. Her ne kadar bu algoritmanın durma noktası istenen küme sayısına ulaşılma anı olsa da; bizim sistemimiz için durma noktası, var olan kümeler arasındaki en yakın mesafenin etmen algı mesafesinden büyük olduğu durum olarak kabul edilebilir (Şekil 7.2).

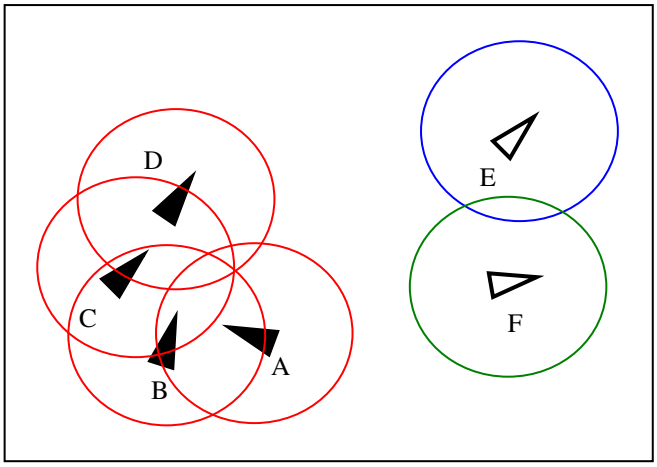


Şekil 7.2 Hiyerarşik demetleme algoritması kullanılarak gerçeklenen takım demetleme işlemi

Şekil 7.2'de akış diyagramı verilmiş olan hiyerarşik demetleme algoritması her çerçeve güncelleme çevriminde çalıştırılmalıdır. Bu algoritma, bir önceki demetlemenin hiçbir verisini kullanmaz (geri besleme yoktur). Algoritmanın karmaşıklığı O(n2) dir. Her adımda kümeler arası mesafeler hesaplanmalı, en kısa mesafe bulunmalı ve uzaklık matrisi güncellenmelidir.

Hiyerarşik demetleme algoritmasının içerdiği işlem yükü sebebiyle sistemimiz için alternatif olarak uygun yapıda ve daha basit bir demetleme algoritması geliştirilmiştir. Geliştirdiğimiz takım demetleme algoritması her etmenin komşuluğundaki kalabalık olan kümeye dahil edilme mantığına göre çalışır. Bu algoritma, aynı zamanda sistemde önceden hesaplanmış verileri (her bir etmen için komşu etmen bilgilerini) kullandığı için işlem karmaşıklığı daha azdır.

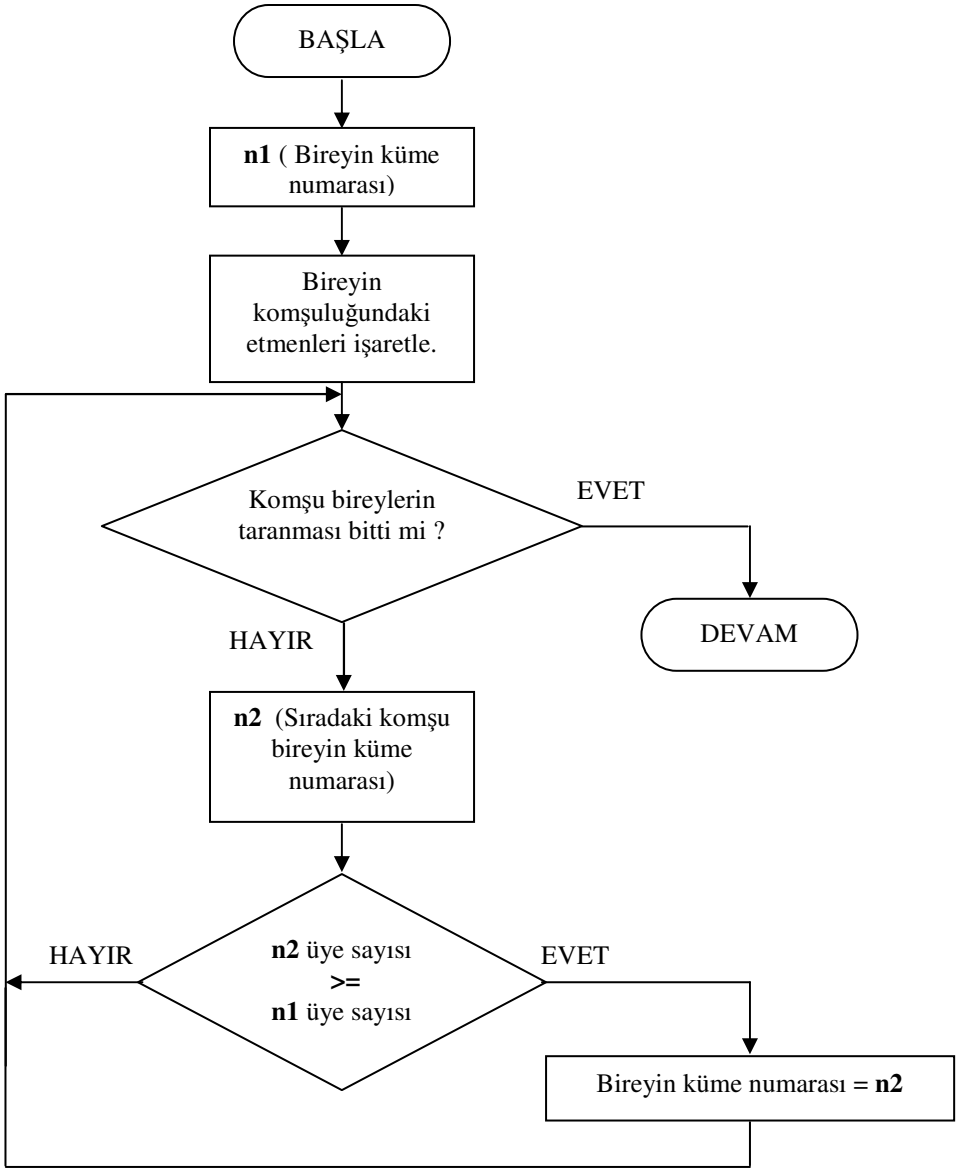
Önerilen demetleme algoritmasının temelinde bireylerin algı bölgesi özelliğinden yararlanılmaktadır. Buna göre *n* adet bireyin aynı küme içinde yer alabilmesi için, herhangi bir bireyin algı bölgesi içinde aynı kümeden en az bir bireyin bulunması gerekmektedir. Bu durumdaki iki birey komşu olarak tanımlanır. Kümenin tüm bireylerini düşündüğümüzde, bu kümenin birbirine en uzakta bulunan iki farklı bireyini bağlayan bir komşuluk zinciri varsa bu zinciri oluşturan her birey aynı kümenin üyeleridir.



Şekil 7.3 Bireylerin komşuluk zinciri

Şekil 7.3'de verilen örnekte sol bölgedeki koyu renkli bireylerin hepsi komşu olarak birbirlerine bağlıdır. Fakat sağ bölgedeki iki etmenin soldaki küme ile herhangi bir ilişkisi yoktur; dolayısıyla bu iki birey sol taraftaki büyük küme içerisine dahil edilmemektedir. Ayrıca yine E ve F bireyleri komşu olmadıklarından ikisi birlikte tek küme içinde de yer alamazlar. Şekil 7.2'de A bireyine küme numarası olarak 1 verdiğimizi düşünelim. A bireyinin komşuluğundaki tüm bireyler (bu durumda sadece B) 1 numaralı kümeye dahil

edilir. Bu komşuluk zincirindeki bireyler için işleme devam edilirse; A,B,C ve D bireylerinin 1 numaralı küme içinde yer aldıkları kolayca görülebilmektedir. Diğer yandan E ve F bireyleri farklı iki küme numarasına (2 ve 3) sahip olacaklardır.



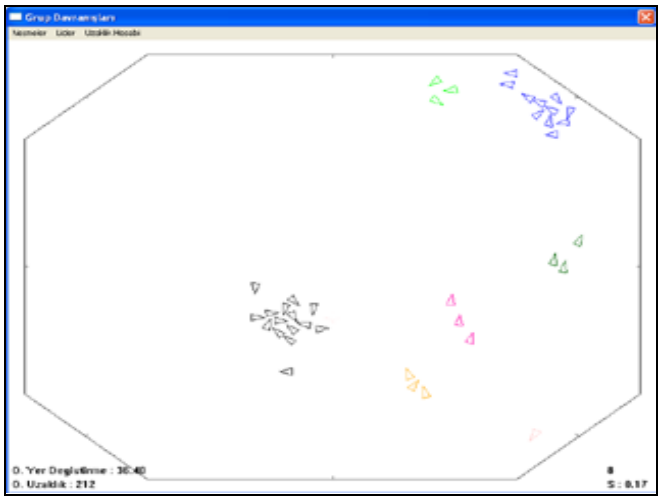
Şekil 7.4 Bir bireye ilişkin takım demetleme algoritması akış diyagramı

Takım demetleme algoritmasının tek bir bireye uyarlanan akış diyagramı Şekil 7.4'te verilmiştir. Bu işlemler dizisi canlandırma ortamında bulunan tüm etmenlere ayrı ayrı uygulanmaktadır.

Canlandırmanın başlangıcında oluşturulan tüm etmenlere farklı birer küme numarası atanmaktadır. Takım demetleme algoritmasında, denetlenen bireyin diğer komşu bireylerin içinde oldukları kümelere dahil olup olmamaları sınanmakta ve komşu bireylerin bağlı olduğu

kümelerin üye sayıları eğer sınanan bireyin kümesindeki üye sayısından büyük veya eşit ise birey bu yeni kümeye katılmaktadır.

Farklı küme numarasına sahip her birey için farklı çizim renkleri kullanılmıştır; böylelikle çalışma anında kümelerin oluşumu ve birleşmeleri Şekil 7.5'de verilen örnek bir uygulamada da görülebileceği gibi daha rahatça izlenebilmektedir.



Şekil 7.5 Çoklu takımlara ilişkin örnek bir uygulama

7.2 Bağımsız Hareket Eden Takımlar

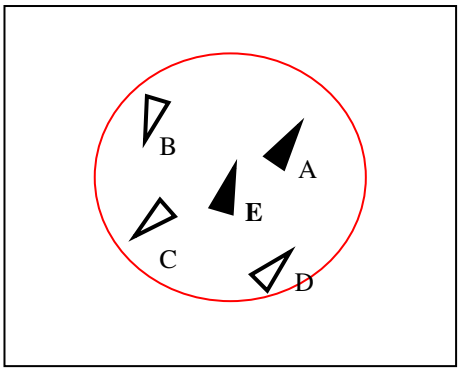
Bağımsız hareket eden takımları oluşturmak için bireyler yaratılırken her bir takımda kaç üye olması isteniyorsa bireylere ona göre doğrudan bir takım numarası atanabilir. Takım üye sayılarının rastgele seçilmesi ile uygulamada kolaylık sağlanabilir ancak yerleri de rastgele atanan takım üyelerinin birleşerek bir sürü hareketi gerçekleştirmeleri ya uzun bir süre alır ya da ortamdaki birey ve engellerin sayısı fazla ise özdeş bireylerden tek sürü veya tek takım oluşturmak mümkün olmaz.

Halbuki başlangıçta oluşan kümelerden takımlar oluşturularak gerçekçi görülen bir canlandırma ortamı daha kolay kurulabilir. Canlandırmanın başlangıcında takım demetleme algoritması çalıştırılır ve her bir bireyin bir kümeye dahil olması alması sağlanır. Daha sonra belirlenmiş bu kümelere birer önder atanarak takım haline dönüşmeleri sağlanır. Her bir takımın ayrı bir tür olarak takımlar arası etkileşimi olmaması isteniyorsa; diğer bir deyişle takımların birbirleriyle birleşmemeleri isteniyorsa, takım demetleme algoritması canlandırmanın bundan sonraki adımlarında çalıştırılmaz. Sistem sonraki çerçevelerde yeniden takım demetleme yapmayacağı için, her takım üye sayısını tüm canlandırma boyunca koruyabilecektir. Bu noktada gerçekçi görünen bir canlandırmanın sağlanabilmesi için sürü

yönlendirme davranışlarında bazı değişikliklerin yapılması gerekmiştir. Sistemde farklı türden *N* adet takım bulunduğundan her takımdaki bireyin, sadece kendi takımına ait bireyler ile etkileşimde bulunmaları gerekmektedir. Sürü hareketini sağlayan üç temel davranış kuralı (bağlılık, ayırma ve hizalama) üzerinde yapılacak ek bir denetim ile bu gerçeklenebilir. Adı geçen üç yönlendirme davranışında, bireyin komşuluğunda bulunan diğer bireyleri belirlemeye yarayan koşula ek olarak aynı zamanda bu iki bireyin küme numaralarının da aynı olması gerektiğinin denetimi eklenmelidir. Böylece birbirinden bağımsız hareket eden takımlarda yer alan bireyler için üç temel sürü davranışının devreye girme şartları şunlardır:

- Komşu birey, merkez bireyin algı mesafesi içinde olmalı (*komşuluk şartı*).
- Komşu birey ile merkez bireyin küme numaraları aynı olmalı (*takım birliği şartı*).

Örnek olarak Şekil 7.6'da açık ve koyu renkli bireylere sahip iki farklı türde takım gösterilmektedir. Merkezde bulunan koyu renkli bireyin (E) komşuluğunda toplam dört adet birey (A,B,C,D) bulunmaktadır. Bu bireylerin hepsi merkezdeki bireyin algı bölgeleri içinde olmalarına karşın sadece bir tanesi (*A*) aynı kümeden olduğu için merkezdeki birey sadece A bireyi ile etkileşimde bulunur. Bu durumda B,C ve D bireyleri merkezde bulunan etmenin sürü hareketini belirleme konusunda herhangi bir etkiye sahip olmayacaklardır.



Şekil 7.6 Farklı türlere ait bireylerin komşulukları

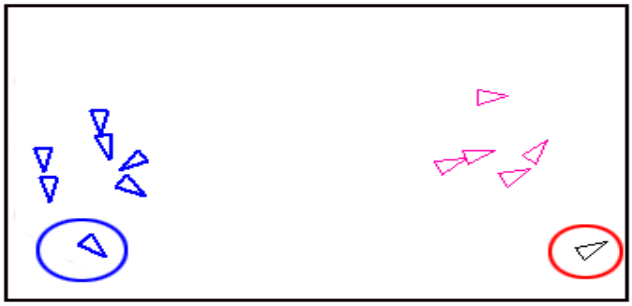
Canlandırma başlangıcında bireyler rastgele konumlarda olacakları için, ilk çerçevede oluşacak takım sayısı oldukça fazla olacaktır. Takımlar arası etkileşime izin verildiği takdirde, belirli bir süre sonunda bu takımlar birleşeceği için az sayıda takıma doğru bütünleşme hareketi göze çarpacaktır. Eğer canlandırma takım sayısının belirli bir *N* sayısında olması isteniyorsa, canlandırmanın her çerçevesinde takım denetimi yapılırken takım sayısı hesaplanıp bu sayı *N* sayısı ile karşılaştırılır ve eşitlikte takım demetleme algoritması artık yürütülmez.

7.3 Aynı Türden Takımların Etkileşimi

Uygulamada bir takımın bireylerinin uzaklaşarak ayrı düşmesi veya daha başlangıçtaki uzaklıklar nedeni ile farklı takımlarda yer almış aynı türden bireyleri kapsayan canlandırmalarda gerçekçi gözüken bir görünüm sergilenmesi için takımların bölünmelerinin sezilmesi, ayrı düşmüş takımlar yakınlaştığında da bunların birleştirilmesi gerekmektedir. Bu konular aşağıdaki bölümlerde ele alınmaktadır.

7.3.1 Takımların bölünmesi

Bir takıma ait olan bireyler çeşitli nedenlerden dolayı (hız farkları, ortamdaki engeller, vb.) bazen bağlı bulundukları takımlardan kopabilirler. Böyle bir durumda oluşacak yeni takımların bireylerinin ve küme numaralarının belirlenmesi gerekecektir.



Şekil 7.7 Takımdan ayrılma

Şekil 7.7'de iki farklı küme için örnek bir senaryo görülmektedir. Sol taraftaki takıma ait olan ve daire içine alınmış olan birey, henüz takımdan ayrılmamıştır (kendisine en yakın takım bireyinin algı mesafesi içindedir). Fakat sağ tarafta daire içine alınmış olan birey, bağlı olduğu takımdan uzaklaşmıştır. Bu durumda bu bireyin farklı bir küme numarası alması gerekmektedir.

Takım bölünmesinin denetimi, canlandırmanın her çerçevesinde her takım için ayrı ayrı yapılmaktadır. Takımda bir bölünme olduğu taktirde, ana takımdan ayrılan bireylere farklı birer küme numarası verilmesi gerekmektedir. Bununla beraber ana takımın küme numarası da değişmemelidir. Hangi bireylerin ana takımdan ayrıldıklarını bulabilmek için izlenen yöntem, yeniden demetleme yöntemidir.

Bunun için geliştirilmiş olan algoritma şu şekildedir:

1) Takımdaki her bir bireye farklı birer denetim numarası verilir.

2) Takımdaki her bir bireyin denetim numarası, komşuluğundaki bireylerin denetim numaraları ile karşılaştırılır ve bu numaraların en küçüğü o bireyin yeni denetim numarası olarak atanır.

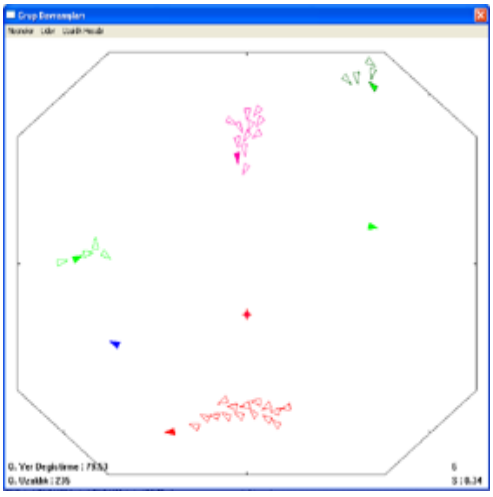
3) Bir önceki aşamada en azından bir atama yapılmışsa, takımdaki tüm bireyler için 2. madde tekrar edilir.

4) 2. maddenin tekrar edilmesi sonucunda hiçbir atama yapılmamış ise, yeniden demetleme işlemi tamamlanmış demektir. Oluşan yeni takımların eleman sayıları hesaplanır. En fazla elemana sahip olan takım, ana takım olarak korunduğu için bu takımın elemanları eski küme numaralarını korur. Diğer bireyler için, yeni bir küme numarası atanır.

Böylelikle ana takımdan ayrılan bireyler yeni küme numaralarına sahip olacaklardır.

7.3.2 Takımların birleşmesi

Takım demetleme algoritmasının çalıştırılması sonucunda oluşan takımlara önderler atanarak, her takımın kendi önderini izlemesi sağlanır.



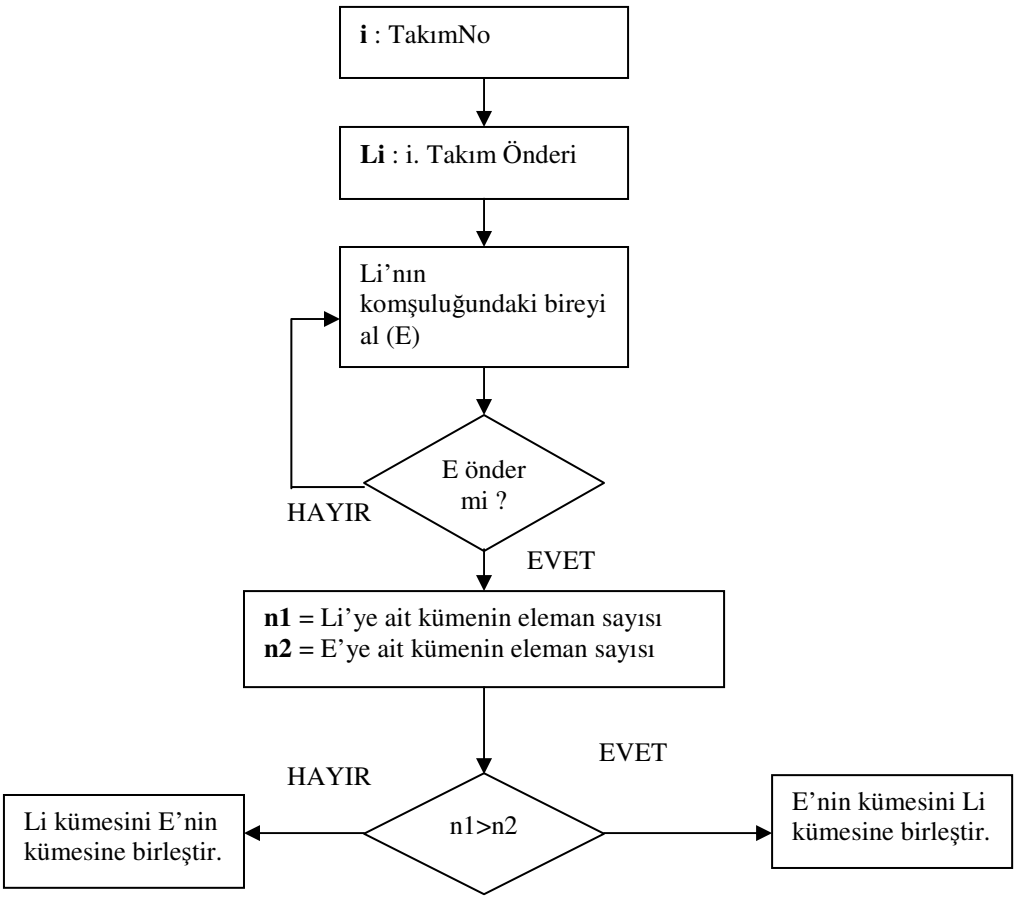
Şekil 7.8 Önder denetimli takımlar

Şekil 7.8'de verilen senaryoda takım önderleri, o kümenin ilk bireyleri olarak seçilmiştir; fakat bu seçim Bölüm 3.2'de açıklanan diğer yöntemler ile de yapılabilir. Örnek senaryoda takım önderlerinin sergileyeceği davranış modeli rastgele gezinme olarak belirlenmiştir. Takımın diğer bireylerine ise önderlerini takip etme ve önderlerinin önünden kaçınma davranışı atanmıştır. Böylelikle her takım kendi önderini belirlemiş ve birbirlerinden etkilenmeden kendi önderini takip edebilmektedir.

Birbirleriyle karşılaşan iki takım arasındaki etkileşimde, iki farklı durum söz konusu olabilir: Birey denetimli etkileşim ve önder denetimli etkileşim.

Birey denetimli etkileşim modelinde A takımına ait bir birey, algı bölgesi içinde B takımına ait bir birey ile karşılaşırsa B takımına dahil olup olmama kararını tek başına verebilir. Takım bireylerinin önderlerinden bağımsız olarak verdikleri bu karar sonucu dahil oldukları takımın önderi, aynı zamanda bu bireyin de yeni önderi olur. Öte yandan önder denetimli etkileşim modelinde küme etkileşimleri tamamen önderlerin denetiminde olur. Takım önderleri birbirlerinin algı bölgelerine girmediği sürece, takım bireyleri karşılaşmış bile olsalar takımlar arasında bir etkileşim olmaz.

Çokluğa dayalı birleşim kuralında takımların birleşimi takım önderlerinin denetiminde sağlanır (Şekil 7.9). Bu durumda iki takımın birleşebilmesi için öncelikle takım önderlerinin karşılaşmaları gerekir. Üye sayısı az olan takım, kalabalık olan takım ile birleşir; birleşen takımın önderi önderlik özelliğini kaybeder ve bu takımdaki tüm bireyler yeni takımın önderini izlemeye başlar. Çok büyük takımların (örneğin üye sayısı belirli bir değerin üzerinde olan takımların) başka takımlar ile birleşmemesi, ek kurallar ile sağlanabilmektedir.



Şekil 7.9 Çokluğa dayalı birleşim kuralı akış diyagramı

8. SONUÇLAR

Bu tez çalışmasında önerdiğimiz önder modeli, takım modeli, takım bireyi modeli ve bu modellerin yazılım tarafında gerçeklenmelerinin bir sonucu olarak sunduğumuz önder denetimli takım canlandırma sistemi ile daha önce sürü canlandırması konusunda yapılan çalışmalarının özellikle önderlik ile ilgili alanlarında görülen boşluğun giderilmesi hedeflenmiştir. Bireylerin sadece yakın çevrelerinde bulunan diğer bireyler ile etkileşimi sonucu ortaya çıkan ve öndersiz hareket eden bireyler topluluğu olarak tanımlanan sürü kavramına alternatif olarak, bir önder denetiminde hareket eden bireyler topluluğunu ifade eden takım kavramı önerilmiş ve sürü canlandırmalarında oluşabilecek tıkanıklık problemleri konusuna tıkanıklığın sezilmesinin ardından önder değişimine gidilerek bir çözüm getirilmesi önerilmiştir.

Körü körüne önderlik şeklinde tanımlayabileceğimiz, takımda yer alan bireylerin önder olarak belirledikleri bir bireyi izlemesi durumlarında, takım hareketi tıkanmaya sürüklenebilmektedir. Buna neden olarak ortamdaki engeller ve tüm etmenlerin öndere doğru yönelmeleri sonucu önderin çevresinin kapanabilme olasılığı verilebilir.

Takımın bir tıkanmaya girip girmediği, takımın hareket yetisinin bir ölçüsü olan ve takım başarım puanı adını verdiğimiz bir parametre ile sınanmaktadır. Bu parametreyi oluşturan kriterlerden takım ortalama yer değiştirme değerinin belirli bir eşik değerin altına düşmesi durumunda takımın bir tıkanmaya girdiği anlaşılmakta ve bu tıkanmanın çözümü için önder değişim algoritmaları devreye sokulmaktadır.

Önder değişim algoritmalarından rastgele önder değişimi, gerçeklenmesi en basit olan ve sistem başarımına fazla yük getirmeyen; fakat uygulamada her zaman istenen sonuçları vermeyen bir yöntemdir. Birey sayısının nispeten az olduğu durumlarda, rastgele seçilen önderin önü açık olma ihtimali fazla olduğu için tercih edilebilir.

Bir tıkanma durumunda öndere en uzaktaki bireyin bulunup yeni önder olarak belirlenmesi, rastgele önder değişimine oranla daha iyi sonuç veren bir yöntemdir ve gerçekleştirilen canlandırmalarda ve yapılan istatistiksel analizlerde bu olumlu sonuçlar gözlenmiştir.

Önderin önünün her zaman açık tutulması, çoğu zaman takım hareketinin tıkanmaya girmesini önleyici yönde bir çözümdür. Fakat sistemde bulunan her etmenin bu davranışı canlandırma süresince çalıştırıyor olması, diğer çözüm yöntemlerine göre sistem başarımına bir miktar ek yük getirmektedir. Ayrıca takım bireylerinin önderin önünden kaçınırken

sergiledikleri dalgalanma ve yarılma hareketleri, özellikle takım canlandırmasının görselliğinin önemli olduğu uygulamalar için kullanışsız olabilmektedir.

Takım canlandırmalarında oluşabilecek tıkanmalara karşı önerilen çözüm yöntemleri engelli ve engelsiz ortamlar için sınanmış, elde edilen takım ortalama yer değiştirme ve takım başarım puanı verilerinin istatistiksel analizleri yapılarak bu yöntemler karşılaştırılmıştır. Bunun sonucu olarak, engelsiz ortamlar için önder değişim yöntemleri arasında anlamlı bir farkın olmadığı; engelli ortamlar için ise en uzak bireyin önder olarak seçildiği yöntemin takım başarımı açısından daha iyi sonuçlar verdiği ölçülmüştür.

Canlandırma sırasında herhangi bir takımdan uzak düşmüş olan tek bireyler için yakındaki bir takıma dahil olmalarını sağlayan takım arama davranışı önerilmiştir. Bu davranışın bir sonucu olarak tek başına hareket eden bireyin, algı bölgesinin kademeli olarak büyütülmesinin ardından olarak karşılaştığı ilk bireye doğru yönelmesi sağlanmış; böylece bir takıma dahil olma süreci hızlandırılmıştır.

Ayrıca canlandırma süresince mevcut takımların dış etkenler ile bölünmesi ve bunların tekrar birleşmesi sonucu oluşabilecek farklı takımların etkileşimi, geliştirilmiş olan takım demetleme algoritması ile desteklenmiş ve takım etkileşimlerinin gerek bireyler gerekse önderler tabanında gerçeklenmesi sağlanmıştır.

Yaptığımız bu çalışma sonucunda ortaya çıkan ve sistemi özellikle dağıtık yapılar için kullanışlı hale getirecek bazı ileri çalışma konuları olarak şunları önerebiliriz:

Önder değişiminin dağıtık bir yapıda gerçeklenmesi : Sistemimizde önder değişimi merkezi bir yapıda gerçeklenmektedir. Yaptığımız çalışma bilgisayarda canlandırma alanında tek bir bilgisayar üzerinde gerçeklendiğinden, dağıtık yapının sistemimize fazla bir katkısı bulunmamaktadır. Fakat fiziksel canlandırmalarda (örneğin : robotlar, hareketli üniteler, paralel bilgi işleme) dağıtık yapı gerekli hale gelmektedir. Buradaki temel problem bilginin dağıtılması problemi olup, literatürde bu konu üzerinde geliştirilmiş olan yöntemler ile önder değişimi dağıtık bir yapıda gerçeklenebilir.

Takım demetlemesinin dağıtık bir yapıda gerçeklenmesi : Sistemimizde her birey, dahil olacağı takımı kendisi seçmektedir. Fakat bu seçimi yaparken merkezi bir bilgi olan takım üye sayıları kullanılmaktadır. Örnek olarak bireylerin koordinat bilgilerini algı alanındaki diğer bireylerle paylaşması, merkezi bilginin dağıtılmasını sağlayacaktır.

Canlandırma ortamının fiziksel ortama benzer olarak modellenmesi : Sistemimizde her örneklemede etmenlerin konumları ve hızları, fiziksel hareket yasaları ile belirlenmekte olup kusursuz ortam modeli benimsenmiştir. Fakat gerçek hayatta dış etkilerin (örneğin: rüzgar, akıntı, sürtünme vb.) elbette ki bu değerler üzerinde etkisi vardır. Dolayısıyla sisteme eklenecek bu parametreler ile daha gerçekçi canlandırmalar elde edilebilir.

KAYNAKLAR

Aubé, F. ve Shield, R. (2004), "Modeling the Effect of Leadership on Crowd Flow Dynamics. Lecture Notes in Computer Science, 3305:601–611.

Badler N.I, Korein J.D, Radack G.M ve Brotman L.S. (1985), "Positioning and Animating Figures in a Task-oriented Environment", The Visual Computer, 1(4):212-220.

Bayazit O.B., Lien J-M. ve Amato N.M. (2002), "Better Group Behaviors Using Rule-Based Roadmaps", Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Nice, France.

Bécheiraz P. ve Thalmann D. (1998), "A Behavioral Animation System for Autonomous Actors Personified by Emotions", Proceedings of First Workshop on Embodied Conversational Characters, Lake Tahoe, CA: 57-65.

Braun A., Musse S., de Oliveira L. ve Bodmann B. (2003), "Modeling Individual Behaviors in Crowd Simulation", Proc. Computer Animation and Social Agents, New Jersey:143–148

Bruderlin A. ve Williams L. (1995), "Motion Signal Processing", Computer Graphics, Annual Conf. Series, New York:97-104.

Buckland M. (2005), Programming Game AI by Example, Wordware Publishing Inc., Plano.

Burtnyk N. ve Wein M. (1971), "Computer Generated Key-frame Animation", Journal of Society Motion Picture Television Engineers, 80(3):149-153.

Calvert T., Ovans R. ve Mah S. (1994), "Towards the Autonomous Animation of Multiple Human Figures", Proceedings of Computer Animation '94: 69–75.

Canepa D. ve Gradinariu M. (2007), "Stabilizing Flocking via Leader Election in Robot Networks", Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 4838: 52-66.

Gervasi V. ve Prencipe G. (2003), "Coordination without Communication: The Case of Flocking Problem", Discrete Applied Mathematics, 144(3) : 324-344.

Ji Q. ve Gao C. (2007), "Simulating Crowd Evacuation with a Leader-Follower Model", International Journal of Computer Sciences and Engineering Systems, 1(4): 249–252.

Kelly I.D. (1997), "The Development of Shared Experience Learning in a Group of Mobile Robots", PhD Thesis, Department of Cybernetics, University of Reading, UK .

Lasseter J. (1987), "Principles of Traditional Animation Applied to 3D Computer Animation", SIGGRAPH '87 Proceedings:35-44.

Lebar Bajec I., Zimic N. ve Mraz M. (2005) "Simulating Flocks on the Wing: The Fuzzy Approach", Journal of Theoretical Biology 233(2):199-220.

Li T., Jeng Y.J. ve Chang S. (2001), "Simulating Virtual Human Crowds with a Leader-Follower Model" Proceedings of the 2001 Computer Animation Conf., Korea.

MacQueen J.B., (1967) "Some Methods for classification and Analysis of Multivariate Observations", Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297

Magnenat-Thalmann N. ve Thalmann D. (1990), "Computer Animation: Theory and Practice", Springer, Tokyo.

Magnenat-Thalmann N. ve Thalmann D. (1996), "Computer Animation", ACM Computing Surveys, 28:161-163.

Millar R.J., Hanna J.R.P. ve Kealy S.M. (1999), "A Review of Behavioural Animation", Computers and Graphics, 23:127–143.

Musse S. R. ve Thalmann D. (2001), "Hierarchical Model for Real Time Simulation of Virtual Human Crowds", IEEE Transactions on Visualisation and Computer Graphics, 7(2):152-164.

Pelechano N. ve Badler N. (2006), "Modeling Crowd and Trained Leader Behavior During Building Evacuation", IEEE Computer Graphics and Applications, 26(6):80-86.

Reich B.D., Ko H., Becket W. ve Badler N.I. (1994), "Terrain Reasoning for Human Locomotion", Proceedings of Computer Animation '94: 76–82.

Renault O., Magnenat-Thalmann N. ve Thalmann D. (1990), "A Vision Based Approach to Behavioural Animation", Journal of Visualization and Computer Animation, 1(1):18–21.

Reynolds C.W. (1987), "Flocks, Herds, Schools: A Distributed Behavioural Model", Proceedings of the SIGGRAPH '87 Computer Graphics, 21(4):25–34.

Reynolds C.W. (1993), "An Evolved, Vision-based Behavioural Model of Co-ordinated Group Motion. From Animals to Animats", Proceedings of the 2nd International Conference on Simulation of Adaptive Behaviour: 384–92.

Reynolds C.W. (1999), "Steering Behaviors For Autonomous Characters", Proceedings of Game Developers Conference, March 15-19, San Jose, CA: 763-782.

Ridsdale G. (1990), "Connectionist Modelling of Skill Dynamics", Journal of Visualisation and Computer Animation, Part 2, 1(2):66–72.

Takeuchi T., Unuma M. ve Amakawa K. (1992), "Path Planning and its Application to Human Animation Systems" Proceedings of Computer Animation '92: 163–74.

Treuille A., Cooper S., Popovic Z. (2006), "Continuum Crowds", ACM Trans. Graph. 25(3): 1160-1168.

Tu X. ve Terzopoulos D. (1994), "Artificial Fishes: Physics, Locomotion, Perception, Behaviour", Proceedings of Computer Graphics: 43–50.

Ulicny B. ve Thalmann D. (2001), "Crowd Simulation for Interactive Virtual Environments and VR training systems,", Proceedings of Eurographics Workshop on Animation and Simulation: 163-170.

Witkin A., Baraff D., Blum M., Tonnesen D. ve Monheit G. (1997), "Physically Based Modeling: Principles and Practice", SIGGRAPH'97 Course #19 Notes.

Yan Y., Xiong N., Chong N.Y. ve Défago X. (2008), "A Decentralized and Adaptive Flocking Algorithm for Autonomous Mobile Robots", The 3rd International Conference on Grid and Pervasive Computing –Workshops, Kunming, China.

Ek 1 Gerçeklenen yönlendirme davranışlarına ilişkin program kodları

Duvar İzleme Davranışı

```
Vector2D Behaviors::FollowWall(const std::vector<Wall2D>& walls)
{
   const double offsetFromWall = 25.0;
   const double projectedDistance = 15.0;
   double currentDist    = 0.0;
   double closestDist = MaxDouble;
   int closestWall = -1;
   Vector2D closestPoint, nearestPoint, targetPoint;
   // Etmenin 0.1 sn sonraki konumu
   Vector2D projectedPoint = m_pVehicle->Pos() + (m_pVehicle->Velocity() * 0.1);
   for (unsigned int w=0; w<walls.size(); ++w)
   {
         currentDist=NearestDistanceToLine(walls[w].From(),walls[w].To(),
         projectedPoint, &nearestPoint);
         if (currentDist < closestDist)
         {
                  closestDist = currentDist;
                  closestWall = w;
                  closestPoint = nearestPoint;
         }
   }
   if (closestWall >=0)
   {
         Vector2D wn = walls[closestWall].Normal();
         Vector2D pn = Vec2DNormalize(m_pVehicle->Heading());
         targetPoint = closestPoint + (wn * offsetFromWall);
         if (wn.Dot(Vec2DNormalize(pn)) == -1) // Duvara dik gidiyorsa
         {
                  targetPoint += Vector2D(5,5);
         }
         return Seek(targetPoint);
   }
   else
         return Vector2D(0,0);
  }
```

Önder Takip ve Sıralı Takip Davranışları

```
Vector2D Behaviors::OffsetPursuit(const Vehicle*  leader,
                                  const Vector2D offset, bool bDuckling)
{
   const double radiusSq = 120.0f * 120.0f;         // Etmen algı uzaklığı
   if (!bDuckling)          // Önder Takip
   {
         Vector2D worldOffsetPos = PointToWorldSpace(offset,
                                        leader->Heading(),
                                        leader->Side(),
                                        leader->Pos());

          Vector2D toOffset = worldOffsetPos - m_pVehicle->Pos();
          if (toOffset.LengthSq() > radiusSq)
          {
                    return Vector2D(0,0);
          }
          double lookAheadTime = toOffset.Length() /
                   (m_pVehicle->MaxSpeed() + leader->Speed());

          return Arrive(worldOffsetPos + leader->Velocity() * lookAheadTime, fast);
   }
   else               // Sıralı Takip
   {
            if (m_pVehicle->IsLeader())
                     return Wander();
            else
            {
                     int i=0,j=0;
                     bool bFound = false;
                     unsigned int size = m_pVehicle->World()->Agents().size();
                     do
                     {
                              if (m_pVehicle == m_pVehicle->World()->Agents()[i])
                              {
                                       j = (i+1)%size;
                                       bFound = true;
                              }
                              else
                                       i++;
                     }
                     while (!bFound);

                     return OffsetPursuit(m_pVehicle->World()->Agents()[j], offset, false);
            }
     }
}
```

Saldırı Davranışı

```
Vector2D Behaviors::Attack()
{

        m_pVehicle->World()->TagVehiclesWithinViewRange(m_pVehicle,
        Prm.ViewDistance);
         Vector2D toAgent;
         double minLength = 1000.0;
         int k=0;
         int tagCounter=0;
         int size = m_pVehicle->World()->Agents().size();

         Vehicle* evader=m_pVehicle->World()->Agents()[0];

         for (int i=0; i<size;i++)
        {
                if (m_pVehicle->World()->Agents()[i]->IsTagged())
                {
                        tagCounter++;
                        toAgent = m_pVehicle->World()->Agents()[i]->Pos() –
                        m_pVehicle->Pos();
                        if (toAgent.Length()<minLength)
                        {
                                minLength = toAgent.Length();
                                evader = m_pVehicle->World()->Agents()[i];
                                k=i;
                        }
                }
         }
         if (minLength < 25)
        {
                m_pVehicle->World()->SetVicNo(k);
                return Wander();
        }

        if (size==0)
                return Vector2D(0,0);
        else
                return Pursuit(evader);
}
```

Takım Arama Davranışı

```
Vector2D Behaviors::LookForAGroup()
{
	int r = m_pVehicle->GetRadius();
	int size = m_pVehicle->World()->Agents().size();
	bool bAlone = true;
	int newRadius, k;

	m_pVehicle->World()->TagVehiclesWithinViewRange(m_pVehicle, r);

	for (int i=0; i<size;i++)
	{
		if (m_pVehicle->World()->Agents()[i]->IsTagged())
		{
			bAlone = false;
			k = i;
			break;
		}
	}

	if (bAlone)
	{
		newRadius = m_pVehicle->GetRadius()+(Prm.ViewDistance/10);
		newRadius = (newRadius > Prm.ViewDistance*2 ? Prm.ViewDistance*2 :
		newRadius);
	}
	else
	{
		newRadius = m_pVehicle->GetRadius()-(Prm.ViewDistance/10);
		newRadius = (newRadius < Prm.ViewDistance ? Prm.ViewDistance :
		newRadius);
	}

	m_pVehicle->SetRadius(newRadius);

	return Vector2D(0,0);
}
```

Önderin Önünden Kaçınma Davranışı

```
Vector2D Behaviors::EvadeLeader(const Vehicle* leader, bool bRect)
{
    if (bRect)    // Dikdörtgen
        {
                const double boxLength = 50.0f;
                Vector2D localPos = PointToLocalSpace(m_pVehicle->Pos(),
                        leader->Heading(), leader->Side(),leader->Pos());

                if (localPos.x > 0 && localPos.x < boxLength)
                {

                        if (abs(localPos.y)<=40)
                        {
                                if (localPos.y > 0)
                                        return (leader->Side());
                                else
                                        return (leader->Side().GetReverse());
                        }
                }
                return Vector2D(0,0);
        }
        else    // Daire Dilimi
        {
                const double radiusSq = 120.0f * 120.0f;
                Vector2D fromLeader = m_pVehicle->Pos()-leader->Pos();

                if (fromLeader.lengthSq() < radiusSq)
                {
                        Vector2D fromLeaderNorm = Vec2DNormalize(fromLeader);
                        double pr = fromLeaderNorm.Dot(leader->Heading());

                        if (pr>=0.866) // cos 30
                        {
                                Vector2D localPos = PointToLocalSpace (
                                m_pVehicle->Pos(),      leader->Heading(),
                                leader->Side(), leader->Pos());

                                if (localPos.y > 0)
                                        return (leader->Side());
                                else
                                        return (leader->Side().GetReverse());
                        }
                }
                return Vector2D(0,0);
        }
}
```

Ek 2 *Group* sınıfına ilişkin program kodu

```
Group::Group(int id)
{
	nCount = 0;
	nLeader = 0;
	m_nDist = 0;
	m_dHdg = 0.0f;
	m_dDisp = 0.0f;
	m_dScore = 0.0f;
	m_nLeaderCounter = 0;
	nID = id;
	bLeaderAssigned = true;
	nGroupCount++;
}

Group::~Group()
{
	nGroupCount--;
}

int Group::GetID()
{
	return nID;
}

void Group::AddMember(Vehicle* pVeh)
{
	nMembers.push_back(pVeh);
	nCount++;
}

void Group::RemoveMember(Vehicle* pVeh)
{
	for (int i=0; i<nCount; i++)
		if (nMembers[i] == pVeh)
		{
			nMembers.erase(nMembers.begin()+(i));
			nCount--;
			break;
		}
}

int Group::GetCount()
{
	return nCount;
}

void Group::SetLeader(int n)
{
	if (n>=0)
	{
		nLeader = n;
		bLeaderAssigned = true;
	}
}
```

```
int Group::GetLeader()
{
	if (bLeaderAssigned)
		return nLeader;
	else
		return -1;
}

int Group::GetLeaderCounter()
{
	return m_nLeaderCounter;
}

void Group::IncreaseLeaderCounter()
{
	m_nLeaderCounter++;
}

void Group::ResetLeaderCounter()
{
	m_nLeaderCounter = 0;
}

int Group::GetAverageDistance()
{
	return m_nDist;
}

double Group::GetAverageHeading()
{
	return m_dHdg;
}

double Group::GetAverageDisplacement()
{
	return m_dDisp;
}

int Group::SelectLeader(int selectionMode)
{
	int n;
	unsigned int i;
	Vector2D centerPoint = Vector2D(0,0);
	Vector2D currentVehPos;
	double tempDist;

	switch (selectionMode)
	{
	case 1:  // İlk birey
			n = 0;
			break;

	case 2:  // Merkeze en yakın birey
			for (i=0; i<nCount; i++)
				centerPoint += nMembers[i]->Pos();
			centerPoint /= nCount;
```

```
                        tempDist = (nMembers[0]->Pos() - centerPoint).Length();
                        n=0;

                        for (i=1; i<nCount; i++)
                        {
                                currentVehPos = nMembers[i]->Pos();
                                if ((currentVehPos - centerPoint).Length() < tempDist)
                                {
                                        tempDist = (currentVehPos - centerPoint).Length();
                                        n = i;
                                }
                        }
                        break;

        case 3: // Merkezin en uzağındaki birey
                        for (i=0; i<nCount; i++)
                                centerPoint += nMembers[i]->Pos();

                        centerPoint /= nCount;

                        tempDist = (nMembers[0]->Pos() - centerPoint).Length();
                        n=0;

                        for (i=1; i<nCount; i++)
                        {
                                currentVehPos = nMembers[i]->Pos();
                                if ((currentVehPos - centerPoint).Length() > tempDist)
                                {
                                        tempDist = (currentVehPos - centerPoint).Length();
                                        n = i;
                                }
                        }
                        break;

        case 4: // Rastgele bir birey
                        n= (rand() % nCount);
                        break;
        default:n=-1;
        };

        SetLeader(n);
        return nLeader;
}

int Group::ChangeLeader(bool bRandom)
{
        int newLeader;
        if (bRandom)
                newLeader = SelectLeader(4);
        else
        {
                // En uzaktaki önder
                int dist = 0;
                int ld = 0;
                double td;
```

```
                for (int i=0; i<nCount; i++)
                {
                        td = (nMembers[i]->Pos()- nMembers[nLeader]->Pos()).Length();
                        if (dist < td)
                        {
                                dist = td;
                                ld = i;
                        }
                }
                newLeader = ld;
        }

        nMembers[newLeader]->GiveLeadership();
        nMembers[newLeader]->SetVelocity(-1*nMembers[newLeader]->Velocity());
        SetLeader(newLeader);
        for (int i=0; i<nCount; ++i)
        {
                if (i!=nLeader)
                {
                        nMembers[i]->TakeLeadership();
                        nMembers[i]->AccessBehavior()->WanderOff();
                        nMembers[i]->AccessBehavior()->FlockingOn();
                        nMembers[i]->AccessBehavior()->
                        OffsetPursuitOn(nMembers[newLeader],Vector2D(2,0),false);
                        nMembers[i]->AccessBehavior()->
                        EvadeLeaderOn(nMembers[newLeader],true);
                }
        }
                return newLeader;
}

int Group::CalculateAverageDistance(bool m_bDistLeader)
{
        m_nDist=0;
        if (!m_bDistLeader)
        {
                for (unsigned int a=0; a<nCount; ++a)
                {
                        Vector2D  currentVehPos = nMembers[a]->Pos();
                        Vector2D  tempDist = Vector2D(0,0);

                        for (unsigned int b=0; b<nCount; b++)
                        {
                                if (b!=a)
                                        tempDist += (nMembers[b]->Pos() - currentVehPos);
                        }
                        tempDist /= nCount-1;
                        m_nDist += tempDist.Length();
                }
                m_nDist /= nCount;
        }
        else
        {
                Vector2D  currentVehPos = nMembers[nLeader]->Pos();
                Vector2D  tempDist = Vector2D(0,0);
```

```
                for (unsigned int b=0; b<nCount; b++)
                {
                        if (b!=nLeader)
                                tempDist += (nMembers[b]->Pos() - currentVehPos);
                }
                tempDist /= nCount-1;
                m_nDist += tempDist.Length();
        }
        return m_nDist;
}

double Group::CalculateAverageHeading()
{

        double tempAngle;
        double Angle = 0;
        double a1,a2,d;

        for (unsigned int a=0; a<nCount; ++a)
        {
                Vector2D Heading;
                tempAngle = 0;
                        a1 = (atan2f((-1*nMembers[a]->Heading().y),
                        nMembers[a]->Heading().x)*180) / 3.1415926534;
                for (unsigned int b=0; b<nCount; b++)
                {
                        if (a!=b)
                        {
                                a2 = (atan2f((-1*nMembers[b]->Heading().y),
                                nMembers[b]->Heading().x)*180)/3.1415926534;
                                d = a1 - a2;
                                if (d<0) d *= -1;
                                if (d>180) d=360-d;
                                tempAngle += d;
                        }
                }
                tempAngle /= (nCount-1);
                Angle+=tempAngle;
        }
        Angle /= nCount;
        return Angle;
}

double Group::CalculateAverageDisplacement(bool bFile, FILE* fp)
{
        m_dDisp = 0;

        for (int i=0; i<nCount; i++)
                m_dDisp += nMembers[i]->GetTraveledDistance();
        for (i=0; i<nCount; i++)
                nMembers[i]->ResetTraveledDistance();
        m_dDisp /= nCount;
        if (bFile && fp!=NULL)
        {
                char data[5];
                sprintf(data,"%.2f",m_dDisp);
```

```
                fwrite(data,sizeof(data),1,fp);
                fwrite("\n",1,1,fp);
        }
        return m_dDisp;
}

double Group::CalculateScore(double k1, double k2, double k3)
{
        m_dScore = (k1*m_dDisp/((k2*m_nDist)+(k3*m_dHdg)));
        return m_dScore;
}

void Group::CodeGroup(GWorld* world)
{
        int code=1;
        for (int k=0; k<nCount; k++)
        {
                nMembers[k]->SetControlCode(code);
                code++;
        }

        bool  bFlag, bSet;
        do
        {
                bFlag = false;
                for (int b=0; b<nCount; b++)
                {
                        bSet=false;
                        int gr1 = nMembers[b]->GetControlCode();
                        int ek = gr1;
                        world->TagVehiclesWithinViewRange(nMembers[b],
                        nMembers[b]->GetRadius());

                        for (int c=0; c<nCount; c++)
                        {
                                int gr2 = nMembers[c]->GetControlCode();
                                if (b!=c && nMembers[c]->IsTagged() && gr2<ek && gr2>0)
                                {
                                        ek = gr2;
                                        bFlag = true;
                                        bSet = true;
                                }
                        }
                        if (bSet)
                                nMembers[b]->SetControlCode(ek);
                }
        }while(bFlag);

}

bool Group::CheckLeadership()
{
        int k=0;
        bool bFound = false;

        while (k<nCount && !bFound)
```

```
	{
		if (nMembers[k]->IsLeader())
		{
			bLeaderAssigned=true;
			bFound = true;
		}
		else
			k++;
	}

	if (bFound)
		return true;
	else
		return false;
}
```

Ek 3 *GWorld* sınıfı içinde gerçeklenmiş olan fonksiyonlara ilişkin program kodu

```
int GWorld::Grouping()
{
	SetGroups();

	for (unsigned int gr=0; gr<Group::GetGroupCount(); gr++)
	{
		if (m_Groups[gr]->GetCount()>1)
		{
			m_Groups[gr]->CodeGroup(this);		// Bireyleri kodlama

			//Yeniden demetleme
			int code = 1;
			int ef = 0;
			int efcode;
			do
			{
				int count = 0;
				for (unsigned int w=0; w<m_Groups[gr]->GetCount(); w++)
					if (m_Groups[gr]->
					Members()[w]->GetControlCode()==code)
						count++;
				if (count>ef)
				{
					ef = count;
					efcode = code;
				}
				code++;

			}while (code<=m_Groups[gr]->GetCount());

			int size = m_Groups[gr]->GetCount();

			bool bDone = false;
			int h=0;
			while (!bDone)
			{
				Vehicle *pVeh = m_Groups[gr]->Members()[h];

				if (pVeh->GetControlCode()!=efcode)
				{
					Group* pGroup = pVeh->GetGroup();
					pGroup->RemoveMember(pVeh);
					CheckGroup(pGroup);


					int newgr = GetFreeGroupNumber();
					pVeh->SetGroupNumber(newgr);
					pVeh->GiveLeadership();
					pGroup = new Group(newgr);
					pGroup->AddMember(pVeh);
					pVeh->JoinGroup(pGroup);
					m_Groups.push_back(pGroup);
					size--;
				}
```

```
                                        else
                                                h++;
                                        if (h==size)
                                                bDone=true;
                                } // end of while

                                for (h=0;h<Prm.NumAgents;h++)
                                        m_Vehicles[h]->SetControlCode(0);

                                SetGroups();

                        } // end of if
                } // end of for
                AssignLeaders();

                return 0;
}

void GWorld::SetGroups()
{
   int curgr,n1,n2,s1,s2,i,j;
   int n = Prm.NumAgents;
        for (i=0;i<n; i++)
        {
                curgr = m_Vehicles[i]->GetGroup()->GetID();
                TagVehiclesWithinViewRange(m_Vehicles[i], m_Vehicles[i]->GetRadius());
                for (j=0;j<n;j++)
                {
                        if (m_Vehicles[j]->IsTagged())
                        {
                                n1 = m_Vehicles[i]->GetGroup()->GetID();
                                n2 = m_Vehicles[j]->GetGroup()->GetID();

                                if (n2>=0 && n1!=n2)
                                {
                                        s1 = m_Vehicles[i]->GetGroup()->GetCount();
                                        s2 = m_Vehicles[j]->GetGroup()->GetCount();
                                        if (s2>=s1)
                                                m_Vehicles[i]->ChangeGroup(n2);
                                }
                        }
                }
        }
}

int GWorld::CountMembersOfGroup(int n)
{
        int s=0;
        for (unsigned int k=0; k<Group::GetGroupCount(); k++)
                if (m_Groups[k]->GetID()==n)
                {
                        s = m_Groups[k]->GetCount();
                        break;
                }
                return s;
}
```

```
void GWorld::CheckGroup(Group *pGroup)
{
	if (pGroup->GetCount() == 0)
	{
		for (unsigned int a=0; a<Group::GetGroupCount(); a++)
			if (m_Groups[a]==pGroup)
			{
				m_Groups.erase(m_Groups.begin()+(a));
				delete pGroup;
				break;
			}
	}
}

int GWorld::GetFreeGroupNumber()
{
	int gn = 0;
	while (CountMembersOfGroup(gn))
		gn++;

	return gn;
}

void GWorld::AssignLeaders()
{
	for (unsigned int i=0; i<Group::GetGroupCount(); i++)
	{
		if (m_Groups[i]->CheckLeadership() == false)
		{
			int ln = m_Groups[i]->SelectLeader(4);

			Vehicle *pLeader = m_Vehicles[m_Groups[i]->
			Members()[ln]->GetID()];
			pLeader->GiveLeadership();

			for (unsigned j=0; j<m_Groups[i]->GetCount(); j++)
			{
				if (j!=ln)
				{
					m_Groups[i]->Members()[j]->TakeLeadership();
					m_Groups[i]->Members()[j]->
					AccessBehavior()->FlockingOn();
					m_Groups[i]->Members()[j]->
					AccessBehavior()->WanderOn();
					m_Groups[i]->Members()[j]->
					AccessBehavior()->
					OffsetPursuitOn(pLeader,Vector2D(0,10),false);
					m_Groups[i]->Members()[j]->
					AccessBehavior()->EvadeLeaderOn(pLeader,true);
				}
			}
		}
	}
}
```

```
void GWorld::JoinByMajority(int limit)
{
	int n1,n2,s1,s2;
	  int n = Prm.NumAgents;
	for (int i=0;i<n; i++)
	{
		if (m_Vehicles[i]->IsLeader())
		{
			TagVehiclesWithinViewRange
			(m_Vehicles[i], m_Vehicles[i]->GetRadius());

			for (int j=0;j<n;j++)
			{
				if (i!=j && m_Vehicles[j]->IsLeader() &&
				m_Vehicles[j]->IsTagged())
				{

				    Group *gr1 = m_Vehicles[i]->GetGroup();
				    Group *gr2 = m_Vehicles[j]->GetGroup();
				    n1 = gr1->GetID();
				    n2 = gr2->GetID();
				    s1 = gr1->GetCount();
				    s2 = gr2->GetCount();
				    int c = (s1 >= s2 ? n1 : n2);
				    if (s1>s2 && s2<limit)
				    {
					m_Vehicles[j]->TakeLeadership();

					for (int k=0; k<s2; k++)
						gr2->Members()[k]->ChangeGroup(n1);
				    }

				    if (s2>s1 && s1<limit)
				    {
					m_Vehicles[i]->TakeLeadership();

					for (int k=0; k<s1; k++)
						gr1->Members()[k]->ChangeGroup(n2);
				    }		// end-if
				}	// end-if
			}	//end-for
		}	//end-if
	}	//end-for
}	//end-function
```

ÖZGEÇMİŞ

Doğum tarihi	18.07.1973	
Doğum yeri	İstanbul	
Lise	1986-1990	Haydarpaşa Anadolu Teknik Lisesi
Lisans	1990-1996	İstanbul Teknik Üniversitesi Elektrik-Elektronik Fakültesi Kontrol ve Bilgisayar Mühendisliği Bölümü
Yüksek Lisans	1996-1999	İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Kontrol ve Bilgisayar Mühendisliği Anabilim Dalı Kontrol ve Bilgisayar Mühendisliği Programı
Doktora	2002-2008	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı

Çalıştığı kurumlar

1996-2000	Tesan Tekstil Makinaları San. ve Tic. Ltd. Şti.
2001-Devam ediyor	İstanbul Kültür Üniversitesi Fen-Edebiyat Fakültesi Matematik - Bilgisayar Bölümü Öğretim Görevlisi