İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

UNITY 3D OYUN ORTAMINDA AKILLI AJANLAR İLE KAÇMA-KOVALAMA OYUN TASARIMI

YÜKSEK LİSANS TEZİ

İhsan ŞAHİN

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Yüksek Lisans Programı

Tez Danışmanı: Doç. Dr. Tufan KUMBASAR

TEMMUZ 2020

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

UNITY 3D OYUN ORTAMINDA AKILLI AJANLAR İLE KAÇMA-KOVALAMA OYUN TASARIMI

YÜKSEK LİSANS TEZİ

İhsan ŞAHİN
(518161035)

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Yüksek Lisans Programı

Tez Danışmanı: Doç. Dr. Tufan KUMBASAR

TEMMUZ 2020

İTÜ, Fen Bilimler Enstitüsü'nün 518161035 numaralı Yüksek Lisans Öğrencisi İhsan ŞAHİN, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "UNITY 3D OYUN ORTAMINDA AKILLI AJANLAR İLE KAÇMA-KOVALAMA OYUN TASARIMI" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Doç. Dr. Tufan KUMBASAR** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Dr. Öğr. Üy. İlker ÜSTOĞLU** İstanbul Teknik Üniversitesi	.....
	Prof. Dr. Şeref Naci ENGİN Yıldız Teknik Üniversitesi	.....

Teslim Tarihi : 15 Haziran 2020
Savunma Tarihi : 7 Temmuz 2020

Aileme,

ÖNSÖZ

Bu yüksek lisans tezi, İstanbul Teknik Üniversitesi Kontrol ve Otomasyon Mühendisliği Bölümü'nde bulunan Yapay Zekâ ve Akıllı Sistemler Laboratuvarı'nda bir yıllık bir çalışmanın sonucudur. Bu süre zarfında gösterdikleri yardımseverlik ve ilgi için Yapay Zekâ ve Akıllı Sistemler Laboratuvarı üyelerine teşekkür ediyorum. Ayrıca, aileme ve arkadaşlarıma yardım ve güvenleri için teşekkür etmek isterim.

En önemlisi, tez danışmanım sayın Doç. Dr. Tufan Kumbasar'a sonsuz yardımları, teşviki ve rehberliği için teşekkürü bir borç bilirim. Kendisinin destekleri olmasaydı bu tez tamamlanamazdı.

Haziran 2020 İhsan ŞAHİN

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **vii**
İÇİNDEKİLER ..... **ix**
KISALTMALAR ..... **xi**
SEMBOLLER ..... **xiii**
ÇİZELGE LİSTESİ ..... **xv**
ŞEKİL LİSTESİ ..... **xvii**
ÖZET ..... **xix**
SUMMARY ..... **xxiii**
1. GİRİŞ **1**
2. UNİTY 3D OYUN MOTORUNA GENEL BAKIŞ **5**
2.1 Unity 3D Oyun Objeleri 6
2.2 Unity 3D Hiyerarşi Penceresi 7
2.3 Unity 3D Araç Kutuları 7
2.3.1 Kontrol araç kutusu 8
2.3.1.1 Navigasyon ağ ajan elemanı 8
2.3.1.2 Dönüşüm elemanı 9
2.3.1.3 Sert gövde elemanı 10
2.3.1.4 Çarpışma elemanı 11
2.3.1.5 Kod elemanı 12
2.3.2 Navigasyon araç kutusu 12
2.3.2.1 Navigasyon Ağları 13
2.3.2.2 A* Algoritması 16
3. KAÇMA-KOVALAMA OYUN TASARIMI **19**
3.1 Kaçma-Kovalama Oyun Modeli ve Kontrol Problemi 19
3.2 Unity 3D Oyun Motorunda Kaçma-Kovalama Oyun Tasarımı 21
3.2.1 Unity 3D'de yol nokta yapısı 23
3.2.2 Unity 3D görüş hattı yapısı 26
3.2.3 Oyunda bulunan ajanlar 28
3.2.3.1 Birbirleri ile haberleşen ajanlar: Kırmızı, mavi ve turuncu ajan 29
3.2.3.2 Bulanık mantık tabanlı ajan: Pembe ajan 35
4. UNİTY 3D OYUN ORTAMINDA YAPILAN ÇALIŞMALAR **45**
4.1 Oyun Alanının Oluşturulması ve Amacı 45
4.2 Kırmızı, Mavi, Turuncu ve Pembe Ajanların Kovalama Davranışları 46
5. SONUÇ VE TARTIŞMA **51**
KAYNAKLAR **53**

KISALTMALAR

MS **:** Milattan Sonra
ÇK **:** Çok Küçük
K **:** Küçük
N **:** Normal
B **:** Büyük
ÇB **:** Çok Büyük
ÇY **:** Çok Yakın
Y **:** Yakın
O **:** Orta
U **:** Uzak
ÇU **:** Çok Uzak

SEMBOLLER

$\boldsymbol{f(x)}$	**:** Durum fonksiyonu
$\boldsymbol{g(x)}$	**:** Başlangıç noktasından mevcut noktaya gitmenin maliyeti
$\boldsymbol{h(x)}$	**:** Mevcut noktadan hedef noktaya gitmenin tahmini maliyeti
$\boldsymbol{d}$	**:** Öklid mesafesi
$\boldsymbol{V}$	**:** Çizgisel hız
$\boldsymbol{\theta}$	**:** Yön açısı
$\boldsymbol{m}$	**:** Kütle
$\boldsymbol{W}$	**:** Açısal hız
$(\boldsymbol{e_x}, \boldsymbol{e_y})$	**:** İki nokta arasındaki koordinat farkı
$\boldsymbol{d_g}$	**:** Görüş hattı mesafesi
$\boldsymbol{\theta_g}$	**:** Görüş hattı açısı
$\boldsymbol{x, y}$	**:** Kesin değerli girişler
$\boldsymbol{A_i, B_i}$	**:** Dilsel değişkenler
$\boldsymbol{z_i}$	**:** Bulanık sistem çıkışı
$\boldsymbol{w_i}$	**:** Üyelik fonksiyonlarının ateşleme değerleri
$\boldsymbol{p}$	**:** Kural sayısı
$\boldsymbol{\mu_i}$	**:** Giriş üyelik fonksiyonlarının aitlik derecesi
$\boldsymbol{b}$	**:** Toplanan kâğıt sayısı

ÇİZELGE LİSTESİ

Sayfa

Çizelge 3.1 : V_p hızı için tanımlanan 3x3 kurallar. **40**
Çizelge 3.2 : d_b mesafesi için tanımlanan 3x3 kurallar. **42**

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1 : Unity 3D oyun motorunun arayüzü……5
Şekil 2.2 : Unity 3D'de bulunan oyun objeleri……6
Şekil 2.3 : Hiyerarşi penceresi……7
Şekil 2.4 : Navigasyon ağ ajan elemanı……8
Şekil 2.5 : Navigasyon ağ ajanı……9
Şekil 2.6 : Dönüşüm elemanı……10
Şekil 2.7 : Sert gövde elemanı……10
Şekil 2.8 : Çarpışma elemanı……11
Şekil 2.9 : Kod elemanı……12
Şekil 2.10 : Navigasyon araç kutusu ……13
Şekil 2.11 : Navigasyon araç kutusunda "Areas" bölümü……13
Şekil 2.12 : Unity 3D oyun motorunda (a)Navigasyon ağsız oyun alanı (b) Navigasyon ağlı oyun alanı……14
Şekil 2.13 : Navigasyon ağ örneği……14
Şekil 2.14 : Ajanın hedefe giderken takip ettiği en kısayol……15
Şekil 2.15 : Navigasyon ağları üzerinde (a)"Static" tanımlı engeller (b)"Navigasyon static" tanımlı engeller……15
Şekil 2.16 : A* algoritması örneği……16
Şekil 2.17 : A* algoritması örneğinde son durum……17
Şekil 2.18 : Unity 3D'de A* algoritmasının kullanımı……17
Şekil 3.1 : Kaçma-kovalama oyun modeli……19
Şekil 3.2 : Kaçma-kovalama oyun modelinin genel kontrol blok diyagramı……20
Şekil 3.3 : Unity 3D'de kaçma-kovalama oyun alanı……22
Şekil 3.4 : Unity 3D'de yol nokta yapısı……23
Şekil 3.5 : Kaçma-kovalama oyun modeli……24
Şekil 3.6 : Dinamik yol nokta yapısı……25
Şekil 3.7 : Görüş hattı yapısı……26
Şekil 3.8 : Kaçma-kovalama oyunundaki tüm ajanlar……28
Şekil 3.9 : Kırmızı ajan strateji yapısı……29
Şekil 3.10 : Kırmızı ajan stratejisinin akış diyagramı……30
Şekil 3.11 : Mavi ajan strateji yapısı……31
Şekil 3.12 : Mavi ajan stratejisinin akış diyagramı……32
Şekil 3.13 : Turuncu ajan stratejisi yapısı……33
Şekil 3.14 : Turuncu ajan stratejisinin akış diyagramı……34
Şekil 3.15 : Pembe ajan strateji yapısı……35
Şekil 3.16 : Bulanık sistemin iç yapısı……36
Şekil 3.17 : Pembe ajanın çizgisel hız kontrol blok diyagramı……39
Şekil 3.18 : Giriş değerlerinin üyelik fonksiyonları……39
Şekil 3.19 : Kaçan ajanın görüş hattında bulunduğu durumlar……40

Şekil 3.20 : V_p hızı için elde edilen yüzey……...…….**41**
Şekil 3.21 : d_b mesafesi için elde edilen yüzey……..…….**43**
Şekil 4.1 : Unity 3D kullanıcı arayüzü…….**45**
Şekil 4.2 : Kaçma-kovalama oyun alanı……..**47**
Şekil 4.3 : Kaçan ajan ile kovalayan ajanlar arasındaki mesafelerin (a) x eksenindeki, (b)y eksenindeki değişimleri…….……...….**47**
Şekil 4.4 : d_k, d_m, d_t ve d_p mesafeleri…….……..**48**
Şekil 4.5 : Kovalayan ajanların çizgisel hızları……...**49**
Şekil 4.6 : Toplanan kağıtların V_k'ya etkisi……...…..**50**
Şekil 4.7 : Toplanan kağıtların (a) W_m'ye, (b) θ_g'ye etkisi…….……**50**

UNITY 3D OYUN ORTAMINDA AKILLI AJANLAR İLE KAÇMA-KOVALAMA OYUN TASARIMI

ÖZET

Kaçma-kovalama oyunları, engelden kaçma, kaçan ajan takibi ve yol takibi gibi diğer birçok problemi kapsayan bir problem olarak görüldüğü için çeşitli çalışmalarda tercih edilmektedir. Kaçma-kovalama oyunlarında temel amaç bir veya birden fazla sayıdaki kovalayan ajanın bir veya birden fazla sayıdaki kaçan ajanı en kısa sürede yakalamasıdır. Araştırmacılar, kaçma-kovalama oyunlarını genellikle karmaşık ve dinamik ortamlarda kaçan ajanı yakalamak için ortaya çıkan yol bulma probleminde kullanmaktadır. Kaçma-kovalama oyunundaki bu tür problemi çözmek için çeşitli yöntemler önermişlerdir; bu yöntemler arasında dinamik programlama, oyun teorisi, bulanık mantık ve yapay sinir ağları sayılabilir.

Bu tez çalışmasında, Unity 3D oyun motorunda kaçma-kovalama oyunu tasarlanmıştır. Bu oyun motorunun tercih edilme sebebi ise kolay kullanımı ve fizik motoruna sahip olmasıdır. Kaçma-kovalama oyunu için oyun motorunda kontrol araç kutusu ve navigasyon araç kutusu kullanıldı. Kontrol araç kutusunda navigasyon ağ ajan elemanı, dönüşüm elemanı, sert gövde elemanı, çarpışma elemanı ve kod elemanı oyun tasarımı için tercih edildi. Navigasyon ağ ajan elemanı, ajanlara oyun alanı içerisinde otonom hareket etmesi, engellerden kaçınması ve hedef takip etmesi için çizgisel ve açısal hız sağlamaktadır. Sert gövde elemanı fizik motorunu kullanmaktadır. Bu eleman ile oyun alanına yer çekimi ve ajanlara kütle eklenmektedir. Ayrıca sert gövde elemanı ajanlara kuvvet ve tork uygulayarak hareket ettirmektedir. Çarpışma elemanı da fizik motorunu kullanmaktadır. Bu eleman ile ajanlar ve oyun alanı fiziksel çarpışma özelliği kazanmaktadır. Kontrol araç kutusunda kullandığımız son eleman olan kod elemanı ile ajanların hareketleri kontrol edilip, yapay zekâ sistemi oluşturulmakta ve ajanlar arasında haberleşme sağlanmaktadır. Navigasyon araç kutusu ile oyun alanında navigasyon ağları oluşturuldu. Bu ağlar ile ajanlar oyun alanında otonom hareket edebilmekte, engellerden kaçabilmektedir. Navigasyon ağları yapısında A* algoritması bulunmaktadır. Bu algoritma sayesinde ajanlar hedef takibi için en kısa yolu hesaplayabilmektedir.

Bu tez kapsamında, Unity 3D oyun motorunda kullanılan araç kutularının özellikleri ve oyun objeleri ile kaçma-kovalama oyun tasarımı yapıldı. Öncelikle oyun alanı için zemin, duvar ve engeller tasarlandı. Oyunun amacı doğrultusunda kaçan ajan tarafından toplanması gereken kâğıtlar oyun alanına eklendi. Daha sonrasında oyuna bir adet kaçan ajan ve dört adet kovalayan ajan yerleştirildi. Kırmızı, mavi, turuncu ve pembe ajanlar kovalayan ajanlar, sarı ajan ise kaçan ajandır. Kaçan ajan insan kullanıcı tarafından kontrol edilirken, kovalayan ajanlar geliştirilen stratejiye göre kaçan ajanı yakalamaya çalışır. Kovalayan ajanların oyunda devriye hareketi için dinamik yol nokta yapısı oluşturuldu. Dinamik yol nokta yapısı ile kovalayan ajanlar oyun alanın tüm bölgelerine gitme olasılığı kazanmıştır. Ayrıca kovalayan ajanlara oyun içerisinde

görüş menzili oluşturmak için görüş hattı yapısı eklendi. Bu sayede kovalayan ajanların görüş hattı içerisinde kaçan ajan bulunduğunda yakalamaya çalışır. Kaçan ajan görüş hattında olmadığı durumda kovalayan ajanlar oyun içerisinde devriye olarak hareket edecektir. Tasarlanan dinamik yol nokta yapısı ve görüş hattı yapısı tüm kovalayan ajanlarda bulunmaktadır.

Kaçma-kovalama oyun modelinde kovalayan ajanların çizgisel hızı kaçan ajandan büyük ve kovalayan ajanların yön açısı kaçan ajanın yön açısından küçük olacak şekilde oluşturuldu. Ayrıca tasarlanan modelde yakalanma tanımı kaçan ajan ile kovalayan ajanların yüzey teması olarak belirlendi. Bu modelde kaçan ajan ile kovalayan ajan arasındaki Öklid mesafesini hata sinyali olarak düşünülebilir. Bu hata sinyalini minimize etmek ve kaçan ajanı yakalamak için kovalayan ajanlara çeşitli kontrolör tasarımı yapıldı. Bu kontrolörler ile kovalayan ajanların çizgisel hızları kontrol edildi. Kovalama sırasında kovalayan ajanlara yön açıları ise oyun motorunda gerekli hesaplama yapılarak sağlandı. Kırmızı ajanda Öklid mesafesini azaltması için PD kontrolör tasarımı yapıldı. Böylece kaçan ajan kırmızı ajanın görüş hattına girdiğinde, kırmızı ajan kaçan ajanı aralarındaki mesafe değişimine göre çizgisel hızını kontrol edecektir. Ayrıca oyunda kâğıt sayısı azaldıkça kırmızı ajanın çizgisel hızı artmaktadır. Kırmızı ajan görüş hattında kaçan ajan olmadığında oyun alanında devriye olarak gezinmektedir. Mavi ve turuncu ajan Öklid mesafesini azaltmak için P kontrolör tasarımı yapıldı. Böylece kaçan ajanı mavi ve turuncu ajan sabit hızla kovalamaya çalışacaktır. Ayrıca bu iki ajana, kırmızı ajan ile haberleşme özelliği sağlanmıştır. Bu sayede mavi ve turuncu ajan oyun içerisinde kırmızı ajanın konumu hakkında bilgi sahibi olmaktadır. Kaçan ajan ile kırmızı ajan arasındaki Öklid mesafesi azaldığında mavi ajan sol taraftan, turuncu sağ taraftan yakalamaya çalışacaktır. Oyunda kâğıt sayısı azaldıkça mavi ajanın açısal hızı ve turuncu ajanın görüş hattı açısında artış meydana gelmektedir. Mavi ve turuncu ajan da görüş hattında kaçan ajan olmadığı durumlarda oyun alanında devreyi gezmektedir.

Pembe ajana strateji geliştirmek için bulanık mantık kullanılmıştır. Bulanık mantık tercih edilme sebebi bulanık mantık tabanlı sistemlerin doğrusal olmayan etkenler ve bilinmeyen sistem dinamikleri karışışında başarılı bir şekilde cevap vermesidir. Bu bağlamda, pembe ajana yakalama stratejileri tanımlamak için "Takagi-Sugeno-Kang" tipi iki adet bulanık sistem tasarlanmıştır. Bu sistemler pembe ajanın görüş hattında kullanılmıştır. Bulanık sistemler iki giriş, bir çıkışa sahiptir. Sistemin girişleri ise kaçan ajan ile pembe ajan arasındaki mesafe ve bu mesafenin birim zamandaki değişimidir. Birinci bulanık sistem fizik motorunun pembe ajana uygulaması gereken kuvvet değerini hesaplamaktadır. Bu kuvvet ile pembe ajanın kaçan ajanı yakalaması gereken çizgisel hız oluşur. Böylelikle pembe ajan, kaçan ajanla arasındaki değişim gösteren mesafeye göre çizgisel hızını ayarlamaktadır. Tasarlanan ikinci bulanık sistem ise pembe ajanın kaçan ajanı yakalaması için gitmesi gereken konumu hesaplamaktadır. Böylece iki adet bulanık sistem ile akıllı bir kovalama stratejisi geliştirildi.

Kovalayan ajanlar için tasarlanan kontrolör yapılarını karşılaştırmak için Unity 3D oyun ortamında çalışmalar yapılmıştır. Tüm kovalayan ajanlar oyun alanında devriye gezerken sabit hızlarla hareket etmektedir. Kovalayan ajanların görüş hattına kaçan ajan girdiğinde ise yakalamak için yapılarında bulunan kontrolör devreye girecektir. Kırmızı ajan yapısında bulundurduğu PD kontrolör ile kaçan ajanı görüş hattı içerisinde yakalamaya çalışacaktır. Değişim gösteren mesafeye göre kırmızı ajan çizgisel hızını ayarlamaktadır. Ancak kırmızı ajan kaçan ajanla yüzey teması sağlayamadığı için kaçan ajanı yakalayamamıştır. Mavi ve turuncu ajan ise P kontrolör

ile kaçan ajanı değişim gösteren mesafede sabit hızla kovalamıştır. Bu iki ajanda kırmızı ajan gibi kaçan ajanla yüzey teması sağlayamadığı için kaçan ajanı yakalayamamıştır. Pembe ajan için tasarladığımız bulanık sistemler ile kaçan ajanı yakalaması için gerekli olan çizgisel hız ve gitmesi gereken konum bilgisi hesaplanmıştır. Pembe ajan tasarladığımız iki adet bulanık sistem sayesinde kaçan ajan ile yüzey teması sağlayıp yakalamış oldu. Böylece tasarladığımız bulanık sistemler ile tatmin edici kovalama performansı sağlanmış ve akıllı kovalama stratejisinin diğerlerine göre üstün olduğu görülmüştür. Ayrıca kaçan ajan tarafından toplanan kâğıtların kırmızı ajanın çizgisel hızına, mavi ajanın açısal hızına ve turuncu ajanın görüş hattı açısına olan etkisi gösterilmiştir.

DESIGN OF A PURSUIT-EVASION GAME WITH INTELLIGENT AGENTS IN THE UNITY 3D GAME-ENVIRONMENT

SUMMARY

Pursuit-evasion games are favored in a variety of studies as they are seen as a problem covering many other problems such as obstacle evading, evader agent tracking, and road tracks. In pursuit-evasion games, the main goal is for one or more pursuer agents to capture one or more evader agents as soon as possible. Researchers often work pursuit-evasion games in the resulting path-finding problem to capture the evader agent in complex and dynamic environments. They have proposed several methods to solve this type of problem in pursuit-evasion gameplay; these methods include dynamic programming, game theory, fuzzy logic, and artificial neural networks.

In this thesis work, a pursuit-evasion game has been designed in the Unity 3D game engine. The reason for choosing this game engine is that it is easy to use and has a physics engine. The control toolbox and navigation toolbox have been used in the game engine for the pursuit-evasion game. Navmesh agent component, transform component, rigid body component, collider component, and script component in the control toolbox have preferred for game design. Navmesh agent component allows to linear and angular speed for pursuer agents to move autonomously within the game environment, avoid obstacles, and follow targets. Rigid body component uses the physics engine. In this way, this component adds gravity to the playing field and mass to the agents. Also, the rigid body component moves the agents by applying force and torque. The collider component also uses the physics engine. With this component, agents and playground gain physical collision features. The script component, which is the last element we use in the control toolbox, controls the movements of the agents, creates an artificial intelligence system, and provides communication between the agents. Navigation meshes are created in the game environment with the navigation toolbox. With these meshes, agents can move autonomously and avoid obstacles in the playground. There is an A * algorithm in the navigation networks mesh. Thanks to this algorithm, agents calculate the shortest path for target tracking.

The pursuit-evasion game design has made with the toolboxes features and game objects in the Unity 3D game engine. The floor, walls, and barriers were designed primarily for the playground. Ground, walls, and obstacles have been designed with cube game objects for the game-environment. For the aim of the game, the papers to be collected by the evader agent added to the game area. Then, one evader agent and four pursuer agents are placed in the game. Red, blue, orange, and pink agents are the pursuer agents, and the yellow agent is the evader agent. While the evader agent is controlled by the human user, the pursuer agents try to catch the evader agent according to the developed strategy. A dynamic waypoint structure has been created for the patrolling movement of the pursuer agents in the game-environment. With the dynamic path point structure, pursuer agents have gained the possibility of going to

the entire section of the playing field. Also, the line of sight structure has been added to pursuer agents to create a viewing range within the game. In this way, pursuer agents try to catch the evader agent in the line of sight. When the evader agent is not in the line of sight, the pursuer agents move patrol mode within the game. The designed dynamic waypoint structure and line of sight structure are in found all pursuer agents.

In the pursuit-evasion game, the game model is designed so that the linear speed of the pursuer agents is greater than the evader agent, and the direction angle of the pursuer agents is smaller than the direction of the evader agent. Also in the designed model, the definition of capture has been determined to be surface contact of pursuer agents with the evader agent. In this model, the Euclidean distance between the evader agent and the pursuer agent can be considered an error signal. In order to minimize the signal of this error, various controller designs have made for the pursuer agents. Linear speeds of the pursuer agents are controlled with these controllers. During the chase, the direction angles of the pursuer agents are provided by making the necessary calculation in the game engine. The PD controller is designed to reduce the Euclidean distance of the red agent. Thus, when the evader agent enters the line of sight of the red agent, the red agent controls its linear velocity according to the distance change between them. Also, as the number of papers decreases in the game, the linear speed of the red agent increases. The red agent navigates as a patrol in the playground when there is no evader agent in the line of sight. P controller has been designed to reduce Euclidean distance in the blue and orange agents. Thus, blue and orange agents try to chase the evader agent at a constant speed. Also, these two agents have been provided with the communication feature with the red agent. In this way, blue and orange agents have information about the location of the red agent in the game-field. When the Euclidean distance between the evader agent and the red agent decreases, the blue agent tries to catch it from the left side and the orange tries to catch it from the right side. As the number of papers decreases in the game, the angular speed of the blue agent and the angle of view of the orange agent increase. Blue and orange agents move autonomously in the playing field when there is no evader agent in their line of sight.

Fuzzy logic has been used to develop a strategy for the pink agent. Fuzzy logic is preferred since fuzzy logic-based systems successfully respond in the confusion of nonlinear factors and unknown system dynamics. For this reason, two fuzzy systems have designed with the “Takagi-Sugeno-Kang” fuzzy model for a pink agent. These systems were used in the pink agent's line of sight. Fuzzy Systems have two inputs and one output. The inputs of the system are the distance between the evader agent and pink agent, and the change of that distance in unit time. The distance, which is the first input, has three membership functions. These are “Close”, “Middle”, and “Far” membership functions according to the distance of the pink agent with the evader agent. The membership functions of the second input are called “Positive”, “Zero” and “Negative” according to the change of distance in unit time. Nine rules have been constructed in fuzzy systems with two inputs that are defined with three antecedent membership functions. The first fuzzy system calculates the force value required to apply the physics engine to the pink agent. With this force, the linear velocity occurs, which pink agent must catch the evader agent. Thus, the pink agent adjusts the linear velocity according to the distance that changes between the evader agent. The second fuzzy system is designed to calculate the position where the pink agent must go to catch the evader agent, thus developing an intelligent chase strategy with two fuzzy systems.

In order to compare controller structures designed for pursuit agents, simulation studies have been conducted in the Unity 3D game-environment. All pursuit agents are moving at constant speeds while patrolling the playground. When the evader agent enters the line of sight of the pursuit agents, the controller in their structure will be activated to capture it. The red agent tries to capture the evader agent within the line of sight with the PD controller. According to the changing distance, the red agent adjusts its linear velocity. Since there was no surface contact with the evader agent, the red agent has not been caught the evader agent. Blue and orange agents chase the evader agent with the P controller at a constant speed at a changing distance. These two agents were unable to catch the evader agent because they could not make surface contact with it, such as the red agent. With the fuzzy systems we designed, the pink agent provided the necessary linear speed and location information to catch the evader agent. Thanks to the two fuzzy systems we have designed, the pink agent has caught the surface contact with the evader agent. Thus, we have designed two fuzzy systems that provide satisfactory pursuit performance and intelligent pursuit strategy is seen to be superior to others. Furthermore, the effect of the papers collected by the evader agent on the linear velocity of the red agent, the angular velocity of the blue agent, and the angle of the orange agent's line of sight structure are shown.

1. GİRİŞ

Oyun kavramı, M.Ö 2000'li yıllarda Mısır'da satrancın oynandığına dair piramitlerde bulunan bulgularla hayatımıza girmiştir[1]. Satranç ile ilgili ilk yazılı belgeler ise M.S 6. yüzyılda Hindistan'da oynandığı yöndedir[1]. Strateji tabanlı olan bu oyunda amaç oyunda bulunan taşların özelliklerini kullanarak rakibin şahını ele geçirip oyunu kazanmaktır. Satrançla başlanan oyun kültürü 1940'lı yıllara kadar fiziksel olarak oynanmaya devam etti. 1946 yılında ilk dijital bilgisayarın icadı ile birlikte oyunlar bilgisayar ortamında geliştirilmeye başlandı. Bilgisayar ortamında geliştirilen ilk oyun 1960 yılında William Higinbotham tarafından tasarlanan basit bir tenis oyunuydu. 1962 yılında ise Cambridge-Massachusetts'teki Hingham Enstitüsü'nde araştırma görevlisi olan Steve Russell tarafından iki kişilik oyun seçeneği olan ve torpido fırlatma özellikleri içeren uzay gemisi oyunu geliştirildi. Alan Alcorn isimli mühendis ise "Pong" isimli oyunu geliştirdi[2]. Bu oyunda karşılıklı duran iki oyuncu kendi bölgesinde bulunan dikdörtgen şeklindeki beyaz kutucuğu hareket ettirerek oyun alanında hareket eden noktayı kendi sahalarına girmesini engellemekti. 1970'li yıllara kadar oyunlar iki boyutlu olarak tasarlanmaya devam etti. İlk üç boyutlu oyun 1974 yılında tasarlanan "Maze War" isimli oyundu. 1980 yılında "Namco" firması tarafından "Pac-man" oyunu yapılmıştır[3]. Amaç ise oyunda bulunan dört farklı renkteki hayaletlerden kaçarak labirentte bulunan sarı diskleri toplamaktı. "Pacman" oyunundaki gelişmiş yapay zekâ sistemi ile oyunlarda yapay zekâ önemli bir konu olmaya başladı[4-6]. 1990'lı yıllara kadar geliştirilen oyunlarda fizik motoru kullanılmıyordu, sadece gerçek fiziği taklit eden bir dizi kurala dayanıyordu. 90'lı yıllar ile birlikte oyunlar oyun motoru ve fizik motoru ile birlikte geliştirilmeye başladı[7]. Oyun motorları ile geliştirilen oyunlarda kullanıcı tarafından kontrol edilmeyen karakterlerde "Non-Player-Character" (NPC) ortaya çıkan problemler için çeşitli yöntemler ve algoritmalar kullanılmıştır. Bilgisayar oyunlarında bu karakterlerin yol bulma probleminde A* algoritması[8,9] ve Dijkstra algoritması[10] kullanılmıştır. Ayrıca bu karakterlerin oyun ortamında değişen durumlar karşısında cevap vermeleri ve aksiyona girmeleri için sonlu durum makinesi[11], davranış

ağacı[12] ve “Goal Oriented-Action Planning” (GOAP)[13-15] gibi yöntemlerle yapay zekâ sistemi oluşturulmuştur.

Günümüzde geliştirilen bilgisayar oyunları genel olarak kaçma-kovalama senaryoları ile oluşturulmaktadır. Kaçma-kovalama oyunları genel olarak kaçan ve kovalayan olmak üzere iki ajandan oluşmaktadır. Yapılan çalışma doğrultusunda kaçan ve kovalayan ajan sayısı artırılabilir. Örneğin, bir kaçan ajan ve bir kovalayan ajandan oluşan senaryo oluşturulduğu gibi, bir kaçan ajan ve birden fazla sayıda kovalayan ajandan oluşan diğer bir senaryo düşünülebilir. Kaçma-kovalama oyunlarındaki temel amaç bir veya birden fazla sayıdaki kovalayan ajanların en kısa sürede bir veya birden fazla sayıdaki kaçan ajanları yakalaması, kaçan ajanların ise mümkün olduğunca kovalayan ajanlara yakalanmaması amaçlanmaktadır[16].

Bu oyun türünün gözetim ve izleme, arama-kurtarma çalışmalarında, düşman tehdidi tespit etme ve yakalama, çevresel tehlikeleri bulma ve etkisizleştirme gibi konularda faydaları bulunmaktadır[17]. Kaçma-kovalama oyununun tipik bir örneği katil şoför oyunudur[18]. Şoför yayaların üzerinden geçmeyi hedefler ancak asıl amaç şoförün yayaların kaçmadığı duruma karar vermesidir. Diğer bir örnek ise kapalı bir alan içerisinde robotların kayıp bir kişiyi aradığını düşünelim, robotlar burada kovalayan ajan, kayıp kişi ise kaçan ajan olarak tanımlanabilir.

Kaçma-kovalama oyun kavramı çeşitli uygulama alanlarında kullanılmaktadır. Örneğin bilgisayar oyunlarında, robotikte, insansız hava araçlarında, hayvan davranışlarının modellenmesi ve ağ güvenliği gibi çeşitli alanlarda tercih edilmektedir[16]. Geniş uygulama alanı nedeniyle, literatürde kaçma-kovalama oyunu ile ilgili araştırmacılar tarafından çeşitli senaryolar önerilmiştir. Bu senaryolar aşağıda listelenmiştir[16]:

- Oyunda bulunan ajanların sayısı: Oluşturulan senaryoda bir kaçan ajan ve bir kovalayan ajan ya da bir kaçan ajan ve birden fazla sayıda kovalayan ajan olması durumu.
- Oyunda bulunan ajanların hareket kabiliyeti: Kaçan ve kovalayan ajanların oyun alanı içerisindeki çizgisel hız değerlerinin ve yörünge açılarının sınırlandırılması durumu.

- Oyun alanı: Oyun alanının şekli (üçgen, dikdörtgen, kare), oyun alanının boyutu, oyun alanının engeller içerip içermemesine göre kaçan ajanın hareketinin sınırlandırılması durumu.
- Yakalanma durumu: Kaçan ajan ile kovalayan ajan arasındaki mesafenin belirlenen değerin altına düşmesi, kaçan ajanın kovalayan ajanlar tarafından çevrelenmesi gibi çeşitli koşullar altında bu durum tanımlanmıştır. Tez kapsamında yakalanma durumu ajanların yüzey teması olarak belirlenmiştir. Bu tanımlama doğrultusunda kovalayan ajan kaçan ajan ile yüzey teması sağladığında yakalanma durumu gerçekleşmiş olur.

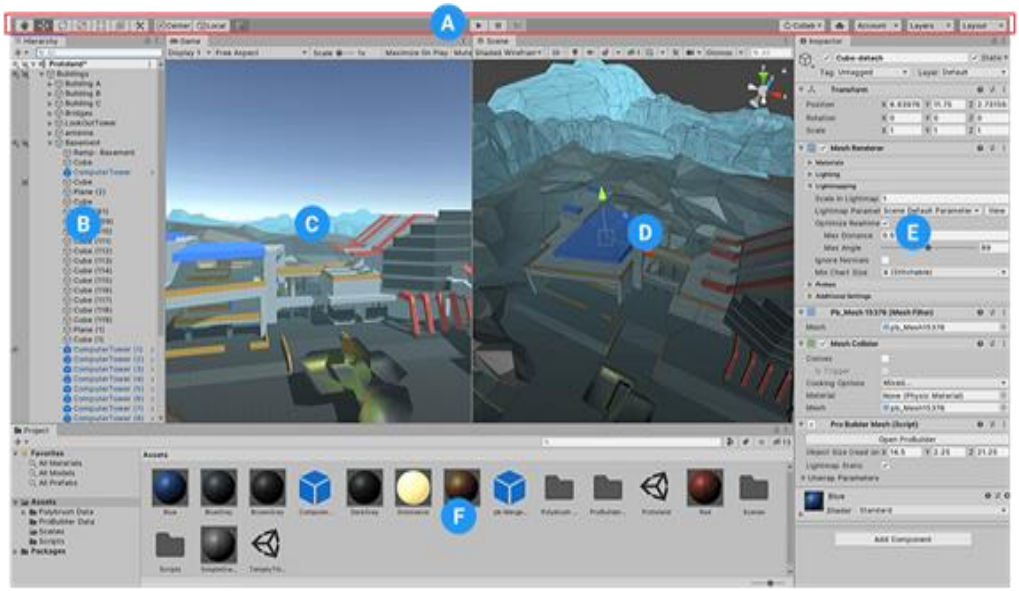
Kaçma-kovalama oyunlarında karşılaşılan problemler ise genel olarak; kaçan ve kovalayan ajanların oyun alanı içerisinde bulunan engellerden kaçınması, kovalayan ajanın hızı ve konumunu kaçan ajana göre ayarlaması, kaçan ajanın kovalayan ajana yakalanmaması ve kaçan ajanın kovalayan ajana yakalanmadığı bölgeyi belirlemesi gibi çeşitli problemler bulunmaktadır. Araştırmacılar tarafından kaçma-kovalama oyun problemini çözmek için çeşitli yöntemler önerilmiştir; bu yöntemler arasında bulanık mantık[19], dinamik programlama[20], oyun teorisi[21], ve yapay sinir ağları[22] bulunur.

Tez kapsamında Unity 3D oyun motoru üzerinde bir adet kaçan ajan (sarı ajan) ve dört adet kovalayan ajandan (kırmızı, mavi, turuncu ve pembe ajan) oluşan kaçma-kovalama oyun tasarımı gerçekleştirilmiştir. Tasarlanan kaçma-kovalama oyununda meydan gelen problemi çözmek için "Takagi-Sugeno-Kang" bulanık mantık tabanlı takip sistemi geliştirilmiştir. Oyun tasarımı için geliştirilen senaryoda kaçan ajan insan kullanıcı tarafından kontrol edilirken, dört adet kovalayan ajan ise oyun alanında tanımlı navigasyon ağları ile otonom gezinmek ve kendilerine tanımlanmış strateji ile kaçan ajanı yakalamaya çalışmaktadır. Oyun alanı Unity 3D oyun motorunda bulunan oyun objeleri ile tasarlandı. Kaçan ajanın oyun alanından çıkmasını engellemek için duvar olarak tanımladığımız oyun objeleri ile oyun alanı kapatılmıştır. Oyun alanı içerisine kaçan ajanın saklanması için çeşitli engeller yerleştirilmiştir. Oyun alanı tasarımı tamamlandıktan sonra oyun bölgesi navigasyon ağları ile taranmıştır. Oyun alanına bir adet kaçan ajan ve dört adet kovalayan ajan yerleştirildi. Kovalayan ajanların ortak özelliklerinden ilki navigasyon ağları ve yol nokta yapısı ile oyun alanında otonom olarak hareket etmesidir. İkinci sahip oldukları ortak özellik ise görüş

hattının olmasıdır. Bu görüş hattı içerisine kaçan ajan girdiğinde kovalayan ajanlar tarafından görüş hattından çıkana kadar kovalanır, kaçan ajan görüş hattından çıktığında ise kovalayan ajanlar otonom hareketine devam eder. Kovalayan ajanların görüş hattına kaçan ajan girdiğinde navigasyon ağları sayesinde kaçan ajanın konumu tespit edilir. Kaçan ajanın yakalanması için en kısa mesafeyi A* algoritması ile hesaplanır. Üçüncü ortak özellik ise, oyun motorunda bulunan kontrol araç kutusu ile çeşitli elemanlara sahip olmalarıdır. Ajanların kendilerine ait yakalama stratejileri ise oyunda bulunan kırmızı ajan "Pacman" oyununda bulunan "Blinky" gibi agresif hareket ederek kaçan ajanı yakalamaya çalışır. Mavi ve turuncu ajan, kırmızı ajan ile haberleşerek kaçan ajanı sağ ve sol taraftan yakalamaya çalışmaktadır. Kaçan ajanın sağ ve sol taraftan yakalanması için Unity 3D oyun motorunda bulunan Ebeveyn-Evlat ilişkisi kullanılmıştır. Pembe ajana strateji geliştirmek için bulanık mantık kullanıldı. Pembe ajanın görüş hattına kaçan ajan girdiğinde yakalaması için iki adet bulanık sistem tasarlanmıştır. Birinci bulanık sistem ile oyun motorunda bulunan fizik motoru kontrol edildi. Fizik motorunun pembe ajana uygulaması gereken kuvvet değerine göre pembe ajan yeni bir çizgisel hıza sahip oldu. Bu hız değeri ile pembe ajan kaçan ajanı görüş hattı içerisinde yakalamaya çalışmaktadır. İkinci bulanık sistem ise, pembe ajanın kaçan ajanı yakalayacağı konumu hesaplar. Tasarladığımız oyundaki amaç ise kaçan ajanın oyun alanında bulunan kâğıtları kovalayan ajanlara yakalanmadan toplayıp oyunu bitirebilmesidir. Kâğıt sayısı azaldıkça ajanların davranışları agresifleşmeye başlayacaktır. Oyun alanına eklediğimiz son özellik ise ölüm alanı olarak tanımladığımız oyun objeleridir. Kaçan ajan bu bölgelerle temas ettiğinde oyun sona ermektedir. Ölüm alanlarının konumu ve boyutu oyun tekrar başladığında rastgele değer alacaktır.

2. UNİTY 3D OYUN MOTORUNA GENEL BAKIŞ

Unity 3D öncelikli amacı bilgisayar, konsollar ve mobil cihazlara oyun geliştirmek için 2005 yılında "Unity Technologies" firması tarafından geliştirilen bir oyun motorudur. İlerleyen teknoloji ile birlikte bu oyun motoru günümüzde mimari, animasyon ve askeri benzetim gibi çeşitli alanlarda kullanılmaktadır. Unity 3D oyunun motorunu sahip olduğu arayüz Şekil 2.1'de gösterilmiştir. Şekilde gösterilen "A" bölümü oyun motoru içerisinde kullandığımız temel "toolbar" yapısını göstermektedir. Bu "toolbar" ile oyun objesine erişebilir, oyun alanı içerisinde hareket ettirilebilir ve boyutunu ayarlanabilir. "B" bölgesi hiyerarşi penceresini, "C" kısmı oyun tasarımı tamamlandığında nihai halini gördüğümüz oyun ekrandır. "D" bölümü sahne görünüm ekranıdır. Bu ekran ile oyun objelerine gerekli düzenlemeler yapılmaktadır. "E" kısmı Unity 3D oyun motorunun sahip olduğu kontrol araç kutusu olup, bünyesinde çeşitli sayıda elemanlar bulunmaktadır. Arayüzdeki son bölüm olan "F" bölümünde ise proje ve konsol pencereleri bulunmaktadır. Proje penceresinde dosya düzenleme, kütüphane ekleme ve oyun objelerine malzeme ekleme işlemi yapılmaktadır. Konsol penceresinde ise oyun alanı içerisinde veya yazılan kodda bir hata olduğunda kullanıcıya geri bildirim yapmaktadır.



Şekil 2.1: Unity 3D oyun motorunun arayüzü.

2.1 Unity 3D Bulunan Oyun Objeleri

Oyun motoru içerisinde farklı özelliklere sahip oyun objeleri bulunmaktadır. Oyun içerinde çevre tasarımı ve ajanlar için üç boyutlu nesneler olan küp, silindir, küre, kapsül ve yüzey gibi oyun objeleri bulunmaktadır. Bu oyun objeleri oyun alanına standart bir birim metre uzunluğunda tanımlanarak oluşturulur. Şekil 2.2'de oyun motorunda bulunan oyun objeleri gösterilmiştir. Oyun alanında görüntü alabilmek ve oyun başladığında erişim sağlamak için kamera objesi kullanılır. Oyun bölgesine ses eklemek ve ses kontrolü yapmak için "Audio" objesi bulunmaktadır. Kullanıcıyı bilgilendirmek, oyun için gerekli açıklamaları eklemek için "User-Interfaces" (UI) oyun objesi bulunmaktadır. Oyun alanı içerisine ışık eklemek ve ışık miktarını ayarlamak için ise "Light" oyun objesi kullanılır.



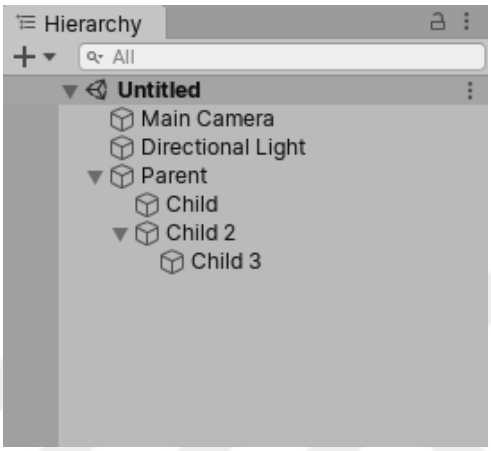
Şekil 2.2: Unity 3D oyun objeleri.

Unity 3D oyun motorunda bulunan tüm oyun objelerinin koordinat ekseni bulunmaktadır. Bu koordinat eksenleri objelerin oyun alanı içerisindeki konumunu ve yönünü temsil etmektedir. Oyun objesi üzerinde bulunan z ekseni ileri-geri yönde hareket, x ekseni sağ-sol yönde hareket ve y ekseni ise yukarı ve aşağı yönde hareket için tanımlanmıştır. Şekil 2.2'de z ekseni mavi renkli ok işareti, x ekseni kırmızı renkli ok işareti ve y ekseni ise yeşil renkli ok işareti ile gösterilmiştir.

Oyun objeleri oyun alanına eklendiğinde yapılarında sadece koordinat eksenleri tanımlıdır. Oyun objelerinden karakter yaratmak veya oyun alanı oluşturmak için çeşitli özelliklerdeki elemanların eklenmesi gerekir. Bu elemanlar kontrol araç kutusu ve navigasyon araç kutusu içerisinde bulunmaktadır. Ayrıca Unity 3D içerisinde oyun objeleri iki kategoriye ayrılır. Eğer oyun objesi oyun alanında hareket halinde ise dinamik oyun objesi, oyun alanında hareketsiz ve sabit ise statik oyun objesi olarak tanımlanır.

2.2 Unity 3D Hiyerarşi Penceresi

Oyun tasarımı yapılırken kullanılan oyun objelerini listelemek ve düzlenmek için kullanılan penceredir. Şekil 2.1'de "B" bölümünde gösterildiği gibi arayüzde sol kısımda bulunmaktadır. Bu yapı sayesinde oyunda kullanılan kamera, ışık ve üç boyutlu objeler belirli sıraya göre dizilir. Şekil 2.3'te oyun motorunda bulunan hiyerarşi penceresi sunulmuştur.



Şekil 2.3: Hiyerarşi penceresi.

Hiyerarşi penceresinde üç boyutlu nesneler (küp, küre, silindir, kapsül vs.) birbirinde bağımsız ve bağımlı olarak listelenir. Eğer oyun objeleri listelenirken birbirine bağımlı olarak gerçekleştirilirse bu yapı Ebeveyn-Evlat ilişkisi olarak tanımlanır. Bu yapıda bir tane ebeveyn oyun objesi ve bu objenin hiyerarşi altında bulunan bir veya birden fazla sayıda tanımlı evlat oyun objeleri bulunmaktadır. Bu ilişki ile evlat oyun objeleri oyun alanı içerisinde hareketlerini ebeveyn oyun objesine bağımlı olarak gerçekleştirir. Konumları ise ebeveyn oyun objesine göre tanımlanır.

2.3 Unity 3D Araç Kutuları

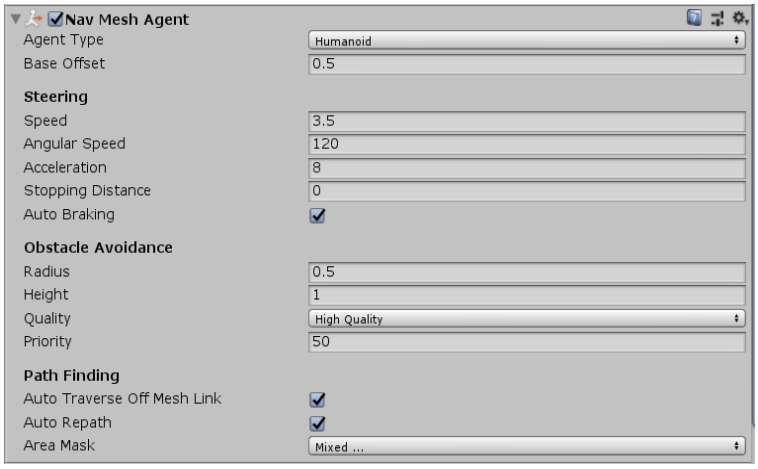
Unity 3D'de oyun objelerine çeşitli özellikler kazandıran araç kutuları bulunmaktadır. Tez kapsamında tasarlanan kaçma-kovalama oyununda bulunan kovalayan ajanlar, kaçan ajan ve oyun alanı tasarımında kullanılan araç kutuları açıklanmaya çalışılmıştır. Birinci kısımda kontrol araç kutusu hakkında bilgi verilip kullanılan çeşitli elemanların özellikleri açıklanacaktır. İlerleyen kısımda ise kovalayan ajanlara yön bulma, otonom hareket etme ve engellerden kaçma özelliği kazandıran navigasyon araç kutusundan bahsedilip sahip olduğu özellikler hakkında bilgi verilecektir.

2.3.1 Kontrol araç kutusu

Oyun motorunun temel araç kutusu olan kontrol araç kutusu bünyesinde çeşitli özelliklere sahip elemanlar bulundurmaktadır. Tez kapsamında kaçma-kovalama oyununda bulunan kovalayan ajanlar, kaçan ajan ve çevre tasarımı için navigasyon ağ ajan elemanı, dönüşüm elemanı, sert gövde elemanı, çarpışma elemanı ve kod elemanı kullanılmıştır.

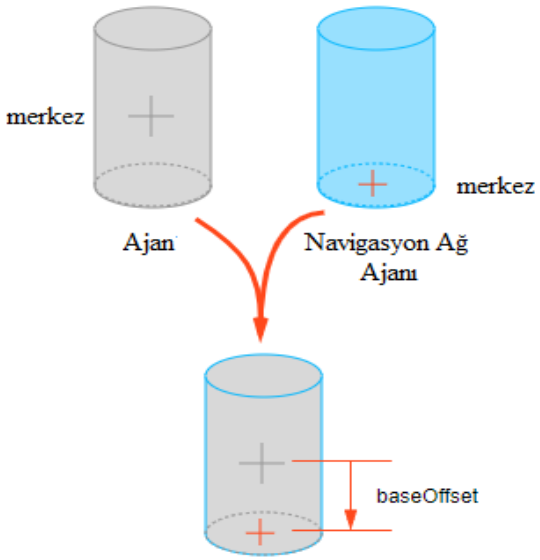
2.3.1.1 Navigasyon ağ ajan elemanı

Bu eleman ile kovalayan ajanlar oyun alanı içerisinde otonom olarak hareket edebilme ve engellerde kaçma özelliğine sahip olmaktadır. Ajanların otonom hareketi ve engellerden kaçması için oyun alanı navigasyon ağları ile örülmesi gerekir. Bu sebeple navigasyon ağ ajan elemanı navigasyon ağları ile birlikte kullanılmaktadır. Kovalayan ajanlara navigasyon ağ ajan elemanı ekleyip, oyun alanı navigasyon ağları ile taranmaz ise ajanlar otonom hareket edemez ve engellerden kaçamaz. Şekil 2.4'te navigasyon ağ ajan elemanı gösterilmiştir. Bu eleman yapısına sahip olan ajanların çizgisel hızı için "Speed" ve açısal hızı için "Angular Speed" bölümlerine girilen sayı ile belirlenmektedir. Bu sayı değerleri çizgisel hız için m/s, açısal hız için $°/s$ birimindendir. Ayrıca girdiğimiz belirli bir sayı değerine göre engellerden kaçma, frenleme ve hızlanma özelliklerine sahip olmaktadır. "Auto Repath " özelliği aktif hale getirildiğinde ise ajanın otonom hareketi için tanımlanan yolun başlangıç noktasından başlayıp bitiş noktasına geldiğinde ajan tekrar takip ettiği yolun başlangıç noktasına gelerek hareketine kaldığı yerden devam etmektedir. Bu özellik kullanılmadığında ise ajan tanımlanan yolu sadece bir kez tamamlar.



Şekil 2.4: Navigasyon ağ ajan elemanı.

Navigasyon ağ ajan elemanın boyutu, yarıçapı ve yüksekliği tanımlandığında Şekil 2.5'te görünen mavi renkli navigasyon ağ ajanı oluşur. Mavi renkli navigasyon ağ ajanı tanımlı olduğu ajan ile birlikte hareket eder.



Şekil 2.5: Navigasyon ağ ajanı.

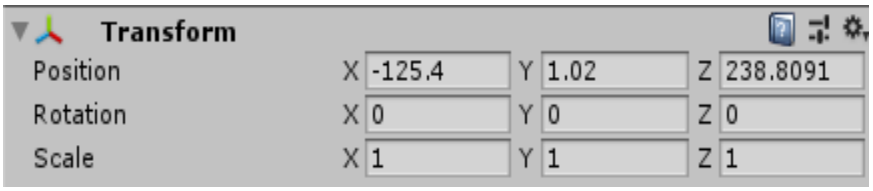
Şekil 2.5'te gösterilen resimde ajanın merkezi orta noktada, navigasyon ağ ajanın merkezi ise alt kısımda bulunmaktadır. İki ajan tek bir gövde üzerinde birleştiğinde merkezlerinin yerleri değişmemektedir. Merkezleri arasındaki mesafe ise navigasyon ağ ajan elemanı parametrelerinde bulunan "baseOffset" değişkenine girilen sayı değerine göre değişmektedir. Girilen sayı değerleri Unity 3D'de metre cinsindendir.

Navigasyon ağ ajanının merkezinin taban kısmında bulunmasının nedeni oyun alanı içerisinde tanımlanan navigasyon ağlarıyla etkileşime girerek, kovalayan ajana oyunda otonom hareket etme, hedef takibi ve engellerden kaçma özelliğini kazandırmasıdır.

2.3.1.2 Dönüşüm elemanı

Dönüşüm elemanının sahip olduğu özelliklerden ilki olan "Position" ile oyun objesi dik koordinat ekseninde hareket ettirilebilir. Girdiğimiz sayı değerine göre oyun objesi dik koordinat sisteminde x, y ve z koordinatlarında hareket eder. Girilen sayı değeri metre cinsindendir. "Rotation" parametresi ile oyun objesi x, y ve z koordinat ekseninde tanımlanan sayı değerine göre dönüş yapmaktadır. Bu alana girilen sayılar derece cinsindendir. Dönüşüm elemanın son özelliği olan "Scale" ise oyun objelerinin oyun ortamında boyutunu ayarlanmaktadır. "Scale" parametresinde girdiğiniz değer

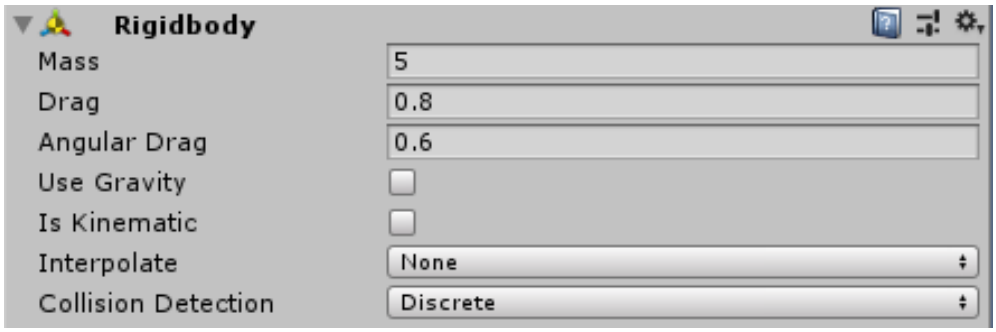
Unity 3D üzerinde metre cinsindendir. Örneğin “Scale” kısmında tüm parametrelere 1 değeri girildiğinde bu x, y ve z ekseninde 1 metre uzunluğa sahip üç boyutlu bir oyun objesi oluşur. “Scale” parametresinde bulunan x, y ve z kutucuklarından herhangi birine sıfır değeri girilirse iki boyutlu oyun objesi oluşur. Şekil 2.6’da dönüşüm elemanının içyapısı sunulmuştur.



Şekil 2.6: Dönüşüm elemanı.

2.3.1.3 Sert gövde elemanı

Unity 3D oyun motoru “Nvidia” firması tarafından geliştirilen “Physic X” adlı fizik motorunu kullanmaktadır. Sert gövde elemanı bu fizik motorunu kullanarak ajanlara ve oyun alanına çeşitli özellikler kazandırmaktadır. Bu eleman sayesinde ajanlar kg biriminde kütleye sahip olmaktadır. Oyun alanına sürtünme ve yer çekimi kuvveti eklemektedir. Ayrıca sert gövde elemanı ile ajanlara kuvvet ve tork uygulanarak hareket ettirilir. Şekil 2.7’de sert gövde elemanı sunulmuştur.



Şekil 2.7: Sert gövde elemanı.

Dönüşüm elemanı ile sert gövde elemanı oyun alanında bulunan oyun objelerini hareket ettirme özelliğine sahiptirler. Ancak aralarındaki fark ise sert gövde elemanı fizik motoru ile oyun objelerine kuvvet ve tork uygulayarak hareket ettirirken, dönüşüm elemanı fizik motorunu kullanmayıp yapısında bulunan (“Position”, “Rotation”, “Scale”) parametrelere girilen sayı değerine göre oyun objelerini hareket ettirmektedir. Sert gövde elemanı dinamik oyun objeleri olan ajanların hareketi için,

dönüşüm elemanı ise statik oyun objeleri olan duvar ve engellerin hareketinde kullanılmıştır.

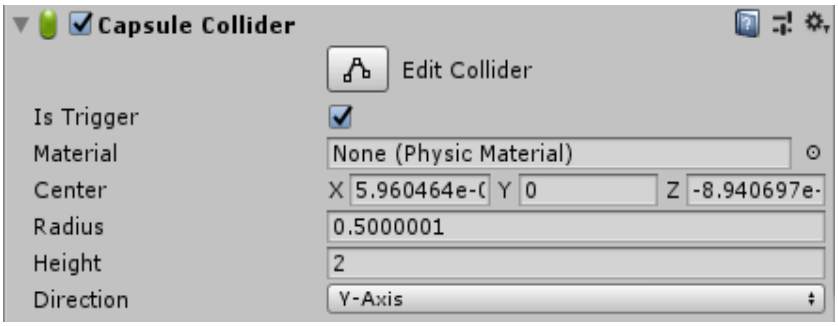
Sert gövde elemanında fizik motorunu kullanmak için "Is Kinematic" yapısının etkileştirilmemesi gerekir, eğer etkileştirilirse fizik motoru kullanılmaz. Oyun objelerine fizik motoru etki etmediği için oyun içinde hareketleri yalnızca dönüşüm elemanı ile yapabilmektedir.

Sert gövde eleman özelliğini aktif kullanan ajanın hiyerarşisi altında oyun objesi tanımlandığında Ebeveyn-Evlat ilişkisi gereği çocuk oyun objesi de fizik motorunu aktif şekilde kullanmaya başlar. Sert gövde elemanı içerisinde bulunan parametre değerleri de çocuk oyun objesi için geçerlidir.

2.3.1.4 Çarpışma elemanı

Oyun motorunda bulunan bu eleman ile ajanlara, duvar, engel ve zemin olarak tanımlanmış oyun objelerine fiziksel çarpışma özelliği kazandırılır. Yapısında kapsül, küp ve küre formunda çarpışma elemanı bulunmaktadır. Bu eleman da sert gövde elemanı gibi fizik motorunu kullanmaktadır.

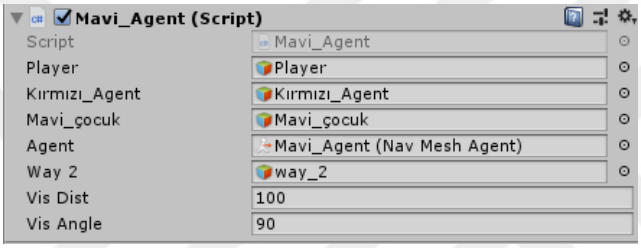
Kovalayan ve kaçan ajanlar yapılarında çarpışma elemanı ve sert gövde elemanı bulundurdukları için dinamik çarpıştırıcı, hareketsiz olarak tanımlanmış duvar, zemin ve engel gibi oyun objeleri sert gövde elemanına sahip olmayıp çarpışma elemanına sahip oldukları için statik çarpıştırıcı olarak tanımlanmıştır. Statik çarpıştırıcılar, dinamik çarpıştırıcılar ile etkileşime girebilir, ancak sert gövde elemanı yapılarında bulunmadığı için hareket edemezler. Oyun içerisinde bu statik çarpıştırıcılar sabit konumda olmalıdır. Eğer statik çarpıştırıcı olarak tanımlamış oyun objeleri hareket ederse fizik motorunun hatalı hesaplama yapmasına ve hata üretmesine neden olur. Şekil 2.8'de oyun motorunda bulunan çarpışma elemanı gösterilmiştir.



Şekil 2.8: Çarpışma elemanı.

2.3.1.5 Kod elemanı

Unity 3D üzerinde C# ve java olmak üzere iki farklı programlama dilini kullanılmaktaydı. Fakat son yıllarda java programlama diline desteğini keserek sadece C# programlama dilinde kodlamaya imkân sağlamaktadır. Bu nedenle bünyesinde sadece C# kütüphaneleri bulunmaktadır. Geliştiriciler C# programlama dilini ve C# kütüphanelerini kullanarak oyun objelerine belirledikleri amaçlar doğrultusunda kontrol edebilmekte ve algoritma geliştirmektedirler. Şekil 2.9'da oyunda bulunan mavi ajan için yazılmış C# kod eleman yapısı gösterilmiştir. Yazılan kod ile ajanların hareketlerini kontrol edebilme, diğer ajanlar ile haberleşmesini sağlama, strateji geliştirme ve ajanlar için yapa zekâ sistemi oluşturabilmektedir.

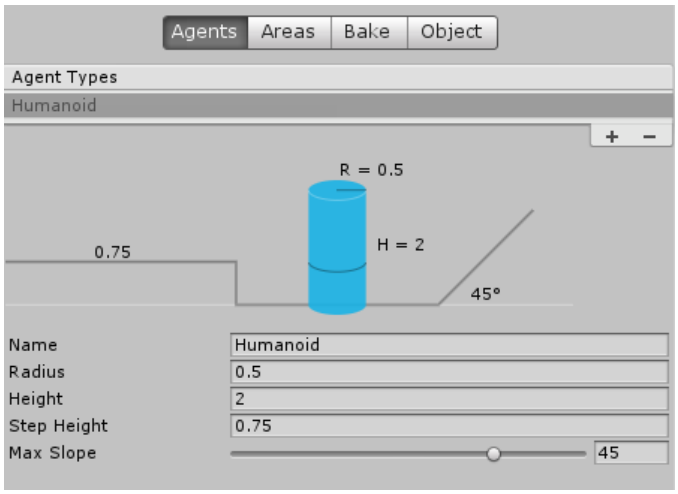


Şekil 2.9: Kod elemanı.

2.3.2 Navigasyon araç kutusu

Bu araç kutusu ile kovalayan ajanlara yön tayini, engellerden kaçma ve belirlenen hedef doğrultusunda takip etme özelliği kazandırılmaktadır. Sahip olduğu çeşitli özelliklerdeki elemanlar ile oyun alanında navigasyon ağı oluşturmaktadır. Şekil 2.10'da gösterilen navigasyon araç kutusu içerisinde "agent","areas","bake" ve "object" kısımları bulunur.

"Agent" kısmında oyun alanında navigasyon ağları üzerinde hareket etmesi gereken ajanlar için gerekli parametre değerleri(yarıçap değeri, yüksekliği) belirlenir. "Bake" kısmı oyun alanına navigasyon ağlarını oluşturmak için kullanılmaktadır. Navigasyon ağlarını yüzey üzerinde oluşturulmadan önce yüzeye "Navigation static" özelliği kazandırılmalıdır. Navigasyon ağları "Navigaiton static" olarak tanımlı yüzeylerde oluşup, "Static" tanımlı olmayan yüzeylerde oluşmamaktadır.



Şekil 2.10: Navigasyon araç kutusu.

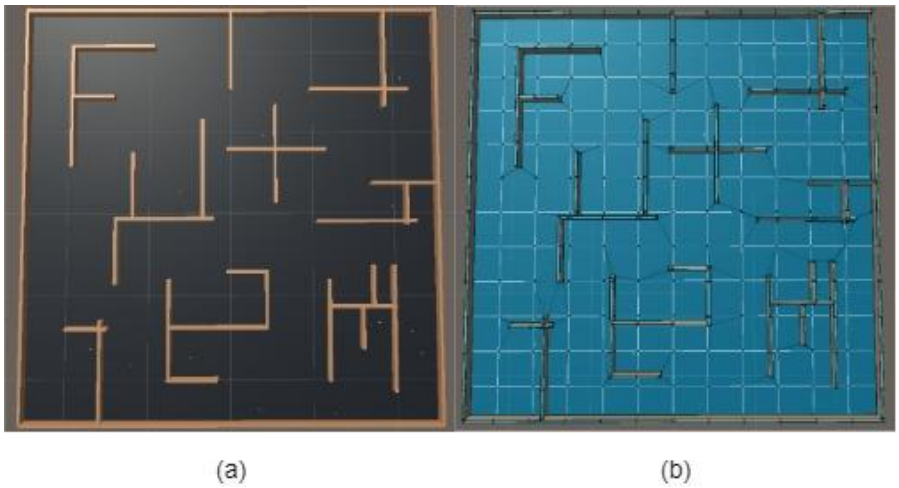
Navigation araç kutusunda bulunan "areas" bölümünde navigasyon ağları üzerinde temel olarak yürünebilir, yürünemez ve zıplanabilir olmak üzere üç tip standart alan tanımlıdır. Tasarımcı oyunun amacı doğrultusunda çeşitli alan tanımlamaları yapabilir. Tanımlanan tüm alanların belli "cost" (maliyet, ağırlık) değeri bulunmaktadır. Ajanlar maliyet değerleri düşük olan yüzey üzerinde hareket etmektedir. En düşük değerli yüzey seçilirken A* algoritmasını kullanılmaktadır. Şekil 2.11 üzerinde navigasyon araç kutusu içerinde bulunan "areas" bölümü gösterilmiştir.



Şekil 2.11: Navigasyon araç kutusunda "Areas" bölümü.

2.3.2.1 Navigasyon Ağları

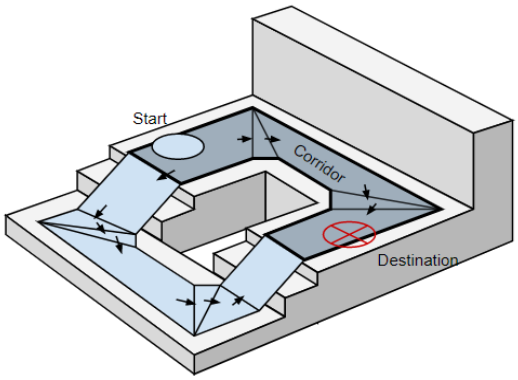
Navigation araç kutusu içerisinde "Bake" özelliği kullanılarak "Navigation static" olarak tanımlanmış yüzey üzerinde navigasyon ağları kareler şeklinde oluşur. Bu ağ sayesinde ajanların otonom hareket etmesi, engellerden kaçması ve belirlenen hedefe doğrultusunda bir yol çizerek ilerlemesi sağlanır. Navigasyon ağlarının oyun alanı içerişindeki görünümü Şekil 2.12'de sunulmuştur.



Şekil 2.12: Unity 3D oyun motorunda (a)navigasyon ağsız oyun alanı (b)navigasyon ağlı oyun alanı.

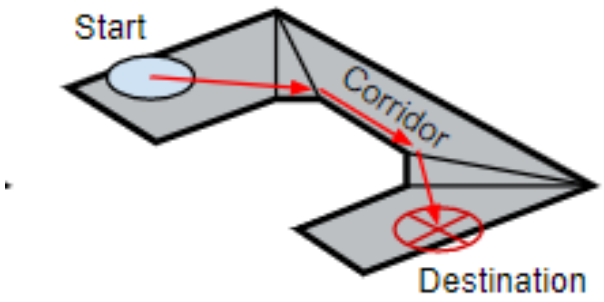
Şekil 2.12.a'da oyun alanında siyah renk ile gösterilen alan oyunun zeminini, kahverengi ile gösterilen oyun objeleri ise engelleri temsil etmektedir. Şekil 2.12.a'da oyun alanı içerisine navigasyon ağlarının eklemeden ve Şekil 2.12.b'de ise eklendikten sonraki halleri gösterilmiştir. Burada navigasyon ağları oyun alanı içerisinde zemin üzerine kare şeklinde ve oyun içerisinde bulunan duvar ve engelleri dikkate alarak oluşmaktadır. Duvar oyun objelerinin sadece üst bölgelerinde navigasyon ağları oluşmaktadır. Duvar oyun objesinin yan kısımlarında oluşmamasının nedeni ise navigasyon ağlarının tarama işlemini yatay olarak gerçekleştirmesindendir.

Şekil 2.13'te sunulan görsellerde ajanların oyun alanı içerisinde navigasyon ağlarını kullanarak nasıl hareket edecekleri açıklanmaya çalışılmıştır. Oyun alanı öncelikle "Navigation static" olarak tanımlanıp, navigasyon araç kutusu kullanılarak navigasyon ağı oluşturulmuştur. Ajanın bu alan içerinde hareket etmesi için gerekli kontrol araç kutusu elemanları eklenmiştir.



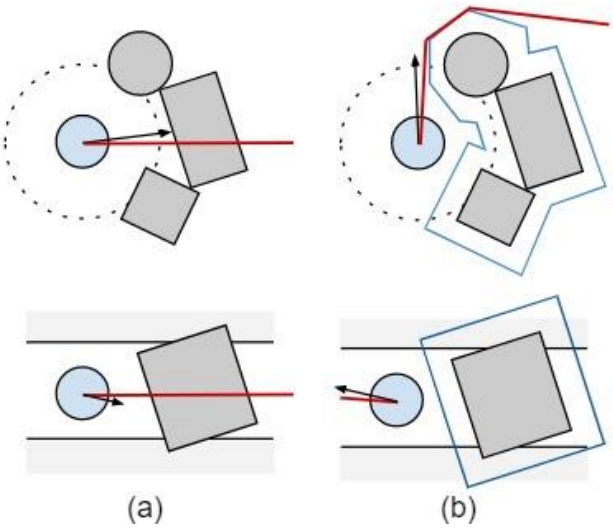
Şekil 2.13: Navigasyon ağ örneği.

Ajan oyun alanında bulunduğu kareden başlayarak hedef noktasına kadar olan tüm karelere değer vererek navigasyon ağını taramaktadır. En kısa yolu bulduğunda ise hedefe doğru bir koridor oluşturarak ilerlememeye başlar. Hedefe ulaşmak için çizdiği en kısa mesafe Şekil 2.14'te gösterilmiştir.



Şekil 2.14: Ajanın hedefe giderken takip ettiği en kısa yol.

Oyun bölgesinde ajanlar herhangi bir engel ile karşılaştığında engellerden kaçınması için navigasyon ağlarından yararlanır. Oyun alanına navigasyon ağları ile taranmadan önce engel olarak yerleştirilen oyun objeleri "Navigation static" olarak tanımlanır, daha sonrasında navigasyon ağları engelleri içine almadan oyun alanı içerisine yerleştirilir. Böylece ajanlar oyun alanı içerisinde engellerin varlığını tespit edip, engellere çarpmadan hedefine gider. Bahsedilen bu yapı Şekil 2.15'te gösterilmiştir. Şekil üzerinde açık mavi çember ajanı, gri renkli geometrik şekiller ise engelleri temsil etmektedir.



Şekil 2.15: Navigasyon ağları üzerinde (a)"Static" tanımlı engeller (b) "Navigation static" tanımlı engeller.

Eğer oyunda bulunan engeller "Navigation static" olarak değil, "Static" olarak tanımlanırsa oyun alanında navigasyon ağları engelleri dikkate almadan oluşacaktır. Şekil 2.15.a'da ki ajan oyun alanındaki engelleri "Static" olarak tanımlandığı için tespit edemeyip gideceği hedefe engelleri dikkate almadan en kısa yolu çizer. Yolu üzerinde bir engelle karşılaştığında ise engeli fark edemeyip çarpmakta, hatta hedefine gidememektedir.

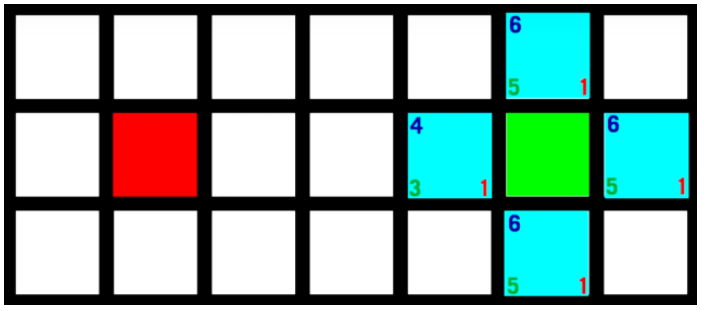
Şekil 2.15.b' de ise oyun alanı ve engeller "Navigation static" olarak tanımlanırsa navigasyon ağları engelleri dikkate alarak oluşacaktır. Böylece navigasyon ağı üzerinde hareket eden ajan engellere çarpmadan en kısa yolu oluşturur.

2.3.2.2 A* Algoritması

Temelinde düğümler halinde tanımlanmış bir kapalı alan içerisinde en kısa yolu bulma algoritması olan A* algoritması genel olarak bilgisayar oyunlarında[23], robotik sistemlerde[24] ve coğrafi bilgi sistemleri[25] gibi benzeri alanlarda kullanılmaktadır. A* algoritmasının matematiksel formülü denklem 2.1 'de gösterilmiştir.

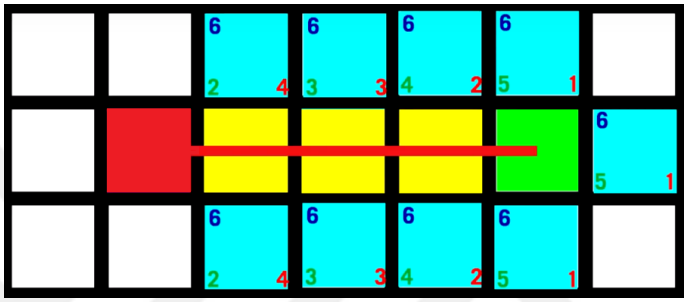
$$f(x) = g(x) + h(x) \tag{2.1}$$

Burada $f(x)$ durum fonksiyonu olup, başlangıç noktasından hedef noktasına kadar olan mesafedeki maliyet hesabını yapar. Sayısal değerinin(ağırlık değeri) az olması, o seçeneğin daha az maliyetli olduğunu ve bir sonraki adımda seçilmesi gereken yol olduğunu gösterir. $g(x)$ fonksiyonu ise başlangıç durumuna göre maliyetini gösterir. Denklem 2.1'de yer alan son fonksiyon olan $h(x)$ ise hedef noktasına göre yapılan sezgisel tahminin maliyetini gösterir. Şekil 2.16'da A* algoritması çalışma prensibi sunulmuştur.



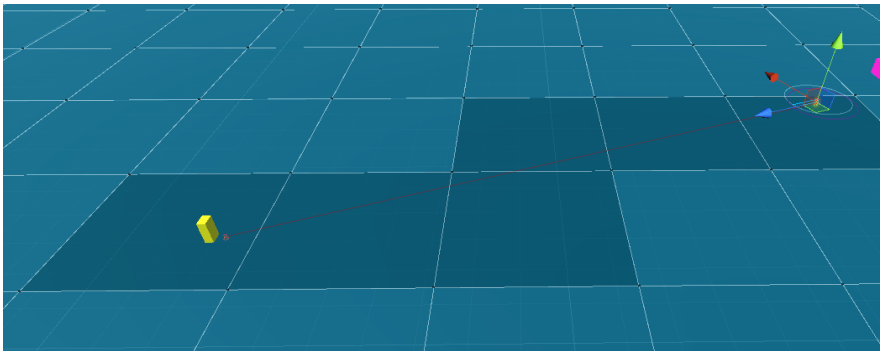
Şekil 2.16: A* algoritması örneği.

Şekil 2.16'da gösterilen resimde yeşil blok başlangıç noktası, kırmızı blok ise gidilmesi gereken hedef noktasıdır. A* algoritması çevresinde bulunan komşu karelere gideceği noktaya göre belirli maliyet değerleri vermektedir. Şekil 2.16'da görülen kırmızı rakamlar denklem 2.1'de belirtilen $g(x)$ fonksiyonu, yeşil rakamlar ise $h(x)$ fonksiyonunu temsil etmektedir. Şekil 2.16'da bulunan mavi rakamlar ise toplam maliyet olup $f(x)$ fonksiyonunu temsil etmektedir. A* algoritması bu matematiksel hesapları yaparak en az maliyetli yolu seçip hedef noktası olan kırmızı bloğa ulaşacaktır. Şekil 2.17'de A* algoritması örneğinde anlatılan yeşil bloğun kırmızı bloğa ulaşırken takip etmesi gereken en kısa yol gösterilmiştir.



Şekil 2.17: A* algoritması örneğinde son durum.

Tez kapsamında A* algoritması Unity 3D üzerinde yol bulma amacıyla oyunda yer alan kovalayan ajanlar tarafından kullanılmaktadır[9]. Bu algoritma oyun motorunda yer alan navigasyon ağ ajan elemanı ve navigasyon ağları ile birlikte kullanılmakta olup, kovalayan ajanlara belirlenen hedefe ulaşması için en kısa yolu hesaplar. Şekil 2.18'de Unity 3D üzerinde kovalayan ajan tarafından A* algoritmasının kullanımı gösterilmiştir.



Şekil 2.18: Unity 3D'de A* algoritmasının kullanımı.

Şekil 2.18'de sağ üst kısımda bulunan kovalayan ajan oyun alanı içerisindeki navigasyon ağlarını ve A* algoritmasını kullanarak hedefi olan sarı bloğa en kısa yolu

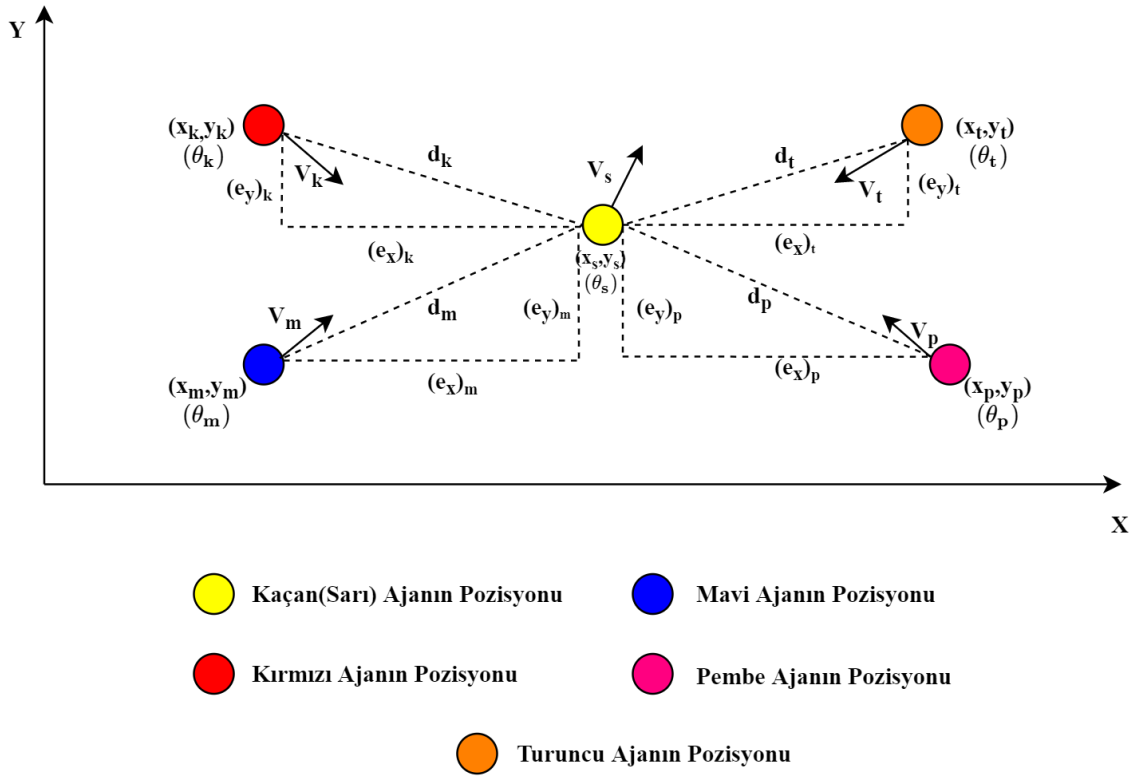
hesaplayarak ilerlemektedir. Resim üzerindeki koyu renkli mavi kareler A* algoritmasıyla hesaplanarak kullanılması gereken kareleri, kırmızı-çizgi ise hedefe giden en kısa yolu temsil etmektedir. Kovalayan ajan kırmızı-çizgiyi takip ederek hedefine ulaşmaktadır. Eğer ajanın gideceği hedef hareket halinde ise, A* algoritması yapılan hesabı güncelleyerek en kısa mesafeyi bulmakta ve takip edilen kırmızı-çizgiyi güncellenmektedir.

3. KAÇMA-KOVALAMA OYUN TASARIMI

Tezin bu bölümünde, kaçma-kovalama oyun modeli tanıtılıp, kaçma-kovalama oyununun kontrol teorisi açısından nasıl ele alınacağı açıklanacaktır. İkinci kısımda Unity 3D oyun motoru üzerinde tasarlanan kaçma-kovalama oyunu, oyun alanı tasarımı, oyunda bulunan kovalayan ajanların ortak özellikleri ve her bir kovalayan ajanın kendine özel kovalama stratejileri izah edilmiştir.

3.1 Kaçma-Kovalama Oyun Modeli ve Kontrol Problemi

Kaçma-kovalama oyununun temel amacı kovalayan ajanların kaçan ajanı en kısa sürede yakalamasıdır. Şekil 3.1'de tez kapsamında tasarladığımız kaçma-kovalama oyun modeli gösterilmiştir. Bu modelde kaçan ajan sarı daire ile gösterilmiş olup, sahip olduğu koordinat eksenleri (x_s, y_s) ile çizgisel hız ve yön açısı sırasıyla V_S ve θ_s olarak tanımlanmıştır.



Şekil 3.1: Kaçma-kovalama oyun modeli.

Oyunda bulunan kovalayan ajanlar ise sahip oldukları renklere göre isimlendirilmiştir. Örneğin kırmızı ajanın sahip olduğu koordinat eksenleri (x_k, y_k) ve sahip olduğu çizgisel hızı ve yön açısı sırasıyla V_k ve θ_k ile temsil edilmektedir. Diğer kovalayan ajanların sahip oldukları koordinat eksenleri, çizgisel hızı ve yön açısı indisleri isimlerinin ilk harfleri kullanılarak tanımlanıp Şekil 3.1'de sunulmuştur.

Kaçma-kovalama senaryosu oluşturulurken kovalayan ajanların çizgisel hızı, kaçan ajanın çizgisel hızından büyük $(V_i > V_s)$, fakat kovalayan ajanın manevra kabiliyetini kısıtlamak amacı ile yön açısı kaçan ajanın yön açısından küçük $(\theta_i < \theta_s)$ olacak şekilde belirlendi. Çizgisel hız ve yön açısı büyüklüğünde bulunan i indisinin yerine k, m, t ve p indisleri yerleştirilirse sırasıyla kırmızı, mavi, turuncu ve pembe ajanların çizgisel hız ve yön açısını temsil eder.

Kaçma-kovalama oyun modelinin genel kontrol blok diyagramı Şekil 3.2'de verilmiştir. Kontrol blok diyagramı dört bloktan oluşmaktadır; Bunlar kaçan ajan, mesafe ve açı üreteci, P kontrolör ve son olarak kovalayan ajan bloğudur. Mesafe üreteci her an için gerekli mesafeyi, açı üreteci ise kovalayan ajan için gerekli olan açı değerini hesaplar. P kontrolör ise her an için kovalayan ajana çizgisel hız sağlar. Blok diyagramında i indisi yerine m ve t indisleri yerleştirilse mavi ve turuncu ajan için bu blok diyagramı sağlanmış olur.



Şekil 3.2: Kaçma-kovalama oyununun genel kontrol blok diyagramı.

Oluşturulan kontrol yapısı detaylandırılırsa; kaçma-kovalama oyununda kovalayan ajan kaçan ajan ile temas ettiğinde yakalama gerçekleşir. Bu bağlamda, öncelikle kaçan ajan ile kovalayan ajan arasındaki mesafeyi oluşturmak için aşağıdaki denklemleri tanımlamamız gerekir.

$$e_x(t_{m+1})_i = x_s(t_{m+1}) - x_i(t_{m+1}) \tag{3.1}$$

$$e_y(t_{m+1})_i = y_s(t_{m+1}) - y_i(t_{m+1}) \tag{3.2}$$

$t_m = mT$ Burada m örneği, T ise örnekleme zamanını temsil etmektedir.

Denklem 3.1 ve 3.2 de gösterilen $e_x(t_{m+1})_i$ ve $e_y(t_{m+1})_i$ kaçan ajan ve kovalayan ajan arasındaki anlık t_{m+1}'deki mesafe farklarıdır. Bundan sonra, kovalayan ajan

tarafından mesafenin en aza indirilmesi amaçlanarak mesafe, mesafe üretecinden aşağıdaki gibi hesaplanır.

$$d_i(t_{m+1}) = \sqrt{e_x(t_{m+1})_i{}^2 + e_y(t_{m+1})_i{}^2} \tag{3.3}$$

Burada, $d_i(t_{m+1})$ kovalayan ve kaçan ajan arasındaki mesafedir. Bu mesafe minimize edilecek bir hata sinyali olarak görülebilir ve böylece takip eden problem kontrol teorisi çerçevesinde ele alınabilir. Bu amaçla, Şekil 3.2'de gösterildiği gibi, denklem 3.3'teki mesafeyi en aza indirmek için bir P kontrolör tasarlanabilir. P kontrolör çıkışında ise kovalayan ajan için gerekli olan hız değeri elde edilmiş olur.

$$V_i(t_{m+1}) = K_p\ d_i(t_{m+1}) \tag{3.4}$$

Burada, K_p oransal kazançtır. Kırmızı ajanda ise kaçan ajanı kovalaması için gerekli çizgisel hız değeri için P kontrolör yerine PD kontrolör tasarlanmıştır.

$$V_k(t_{m+1}) = K_p\ d_k(t_{m+1}) + K_d\big(d_k(t_{m+1}) - d_k(t_m)\big)/T \tag{3.5}$$

Burada, K_p oransal kazanç, K_d ise türevsel kazançtır. Bu tezin geri kalan kısmında, denklemlerin basit ve kısa görünümü için t_{m+1} argümanı kaldırılmıştır.

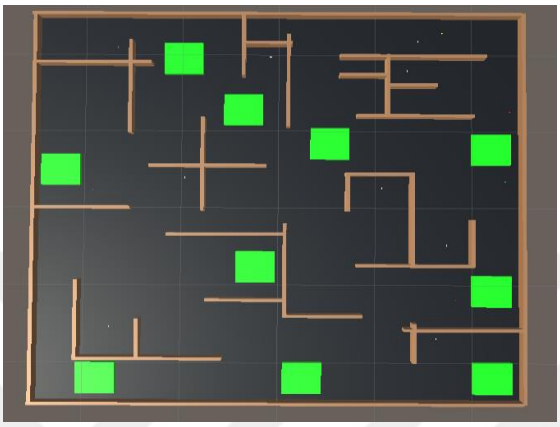
Teoride, kaçan ajanın pozisyonu ile kovalayan ajanın pozisyonunu birleştirmek için sonsuz sayıda olası yörünge vardır. Kaçma-kovalama oyun tasarımı Unity 3D oyun motoru üzerinde yapıldığı için kovalayan ajanın yörüngesini kaçan ajanın konumuna göre ayarlamasında oyun alanında bulunan navigasyon ağları ve A* algoritması kullanılarak gerçekleştirilmekte olup, kovalayan ajanın yön açı değeri aşağıdaki gibi hesaplanır.

$$\theta_i = atan\left((e_y)_i/(e_x)_i\right) \tag{3.6}$$

3.2 Unity 3D Oyun Motorunda Kaçma-Kovalama Oyun Tasarımı

Tez kapsamında kaçma-kovalama oyunu Unity 3D oyun motoru üzerinde tasarlanmıştır. Oyun içerisinde insan kullanıcı tarafından kontrol edilen bir adet kaçan ajan ve dört adet kovalayan ajan bulunmaktadır. Kovalayan ajanların ortak özellikleri ise Unity 3D içerisinde bulunan kontrol araç kutusu ile çeşitli elemanlara sahip olma,

otonom hareket etme ve görüş hattına sahip olmalarıdır. Kovalayan ajanların kaçan ajanı yakalama stratejileri birbirlerinden farklıdır. Oyunun amacı ise oyun alanı içerisinde bulunan beyaz oyun objelerinin (kâğıtların) kaçan ajan tarafından kovalayan ajanlara yakalanmadan toplanmasıdır. Eğer ajanlarla temas edilirse yakalanmış kabul edilip toplanan kâğıtlar sıfırlanacak ve oyun yeniden başlayacaktır. Oyun alanı içerisindeki tüm kâğıtlar toplandığında oyun sona erecektir. Şekil 3.3'te Unity 3D üzerinde tasarlanmış oyun alanı gösterilmektedir.



Şekil 3.3: Unity 3D'de kaçma-kovalama oyun alanı.

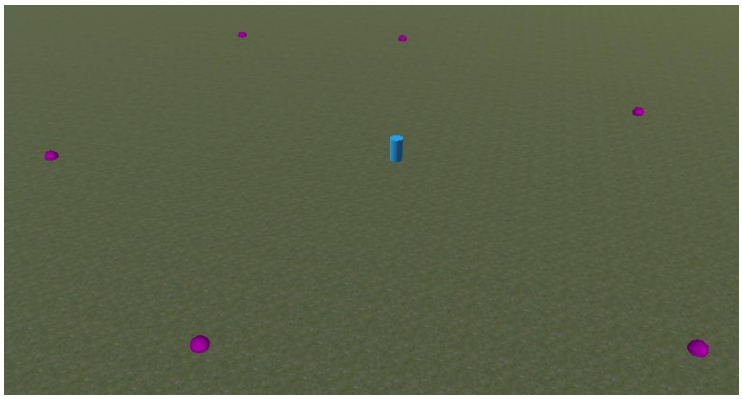
Oyun alanı tasarımı için oyun motoruna birim uzunlukta küp eklenip x ekseninde $50\ m$, y ekseninde $1m$ ve z ekseninde $50\ m$ uzunluk değerleri belirlenerek oyun zemini oluşturuldu. Renk olarak siyah renk seçildi. Oyun alanının konumu oyun motorunun merkezinde olup, $(0m, 0m, 0m)$ konumundadır. Üzerinde hareket eden ajanlarla etkileşime girme ve ajanların zeminden düşmesini engellemek için bünyesine çarpışma elemanı eklenmiştir. Zemine ek olarak oyun alanı tasarımı için yine küp oyun objelerinden duvarlar ve engeller oluşturulmuştur. Şekil 3.3'te kahverengi ile gösterilen duvarlar ve engellerin boyutları oyunda bulunan ajanların boyutlarından büyük olacak şekilde tasarlanmıştır. Duvarlar ve engeller x ekseninde $4\ m$, y ekseninde $20\ m$ ve z ekseninde oyundaki engel tasarımına göre $50\ m$ ile $150\ m$ arasında oluşturulmuştur. Duvarlar ve engellerin yapılarında da çarpışma elemanı bulunmaktadır. Zemin, duvarlar ve engellerin yapılarında sadece çarpışma elemanı olup sert gövde elemanına sahip olmadığı için statik çarpıştırıcı olarak tanımlanır. Böylelikle ajanlar duvarlara ve engellere belirli hız değeri ile çarptığında duvar ve engeller hareket etmemekte ajanlar yapılarında bulundurdukları sert gövde elemanı sayesinde çarptıkları kuvvet kadar oyun alanı içerisine geri dönmektedirler.

Oyun alanı tasarımının ilk aşamalarında ajanlar sadece belirlediğimiz yüzey üzerinde hareket etmekte olup duvarlar ve engeller bulunmamaktaydı. Duvarlar ve engellerin bulunmaması kaçan ajanın oyun alanının dışına çıkmasına ve kaçan ajanın kovalayan ajanlardan saklanamaması gibi çeşitli problemleri beraberinde getiriyordu. Bu sorunları ortadan kaldırmak için duvarlar ve engeller oyun alanına eklemiştir.

Şekil 3.3' te gösterilen yeşil alanlar ise ölüm bölgesi olarak tasarlanmıştır. Kaçma-kovalama oyununda kaçan ajan bu bölge ile temas ettiğinde oyun sona erecek, oyun tekrar başladığında bu yeşil alanların konumu ve boyutu rastgele bir şekilde oyun alanında konumlanacaktır.

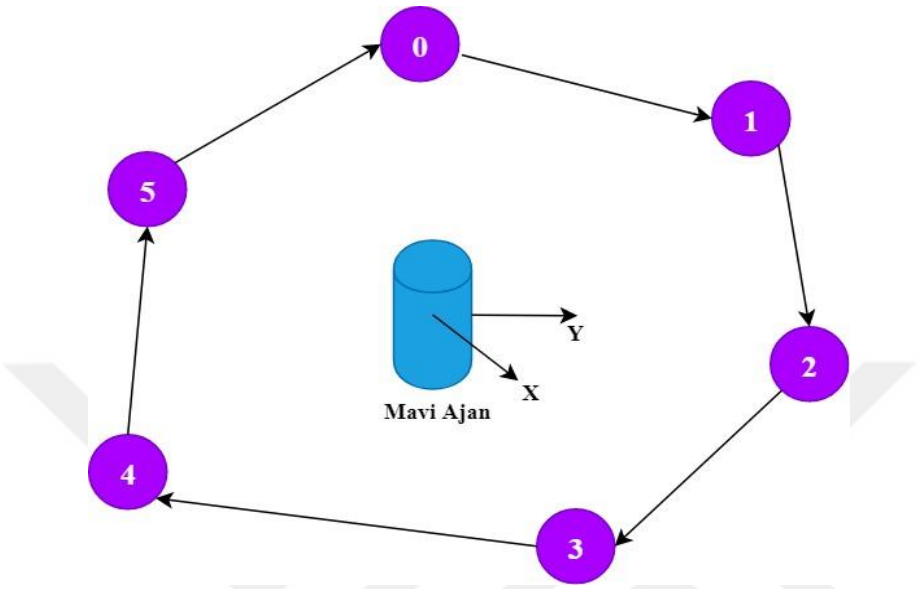
3.2.1 Unity 3D'de yol nokta yapısı

Yol takibi bilgisayar oyunlarında[26], GPS[27], robotik[28] ve lojistik gibi alanlarda önemli bir bileşendir. Tez kapsamında oyun alanı içerisinde kovalayan ajanların otonom hareketini sağlamak için yol takip sistemi, yol nokta yapısı[29] ile oluşturulmuştur. Yol nokta yapısı oyun objelerini kullanarak (küre oyun objesi) oyun alanı içerisinde kovalayan ajanların izleyeceği patikalar oluşturarak gerçekleştirilmiştir. Kovalayan ajanlar otonom hareketi sırasında patikada takip edeceği oyun objelerinin kararını ise Unity 3D'de bulunan etiket yapısı sayesinde karar vermektedir. Patikada bulunan oyun objelerinin hepsine aynı etiket tanımlanıp bir dizi oluşturulur, kovalayan ajanlar ise kendi hafızalarında bu diziye sahip olduğu için sırasıyla bu oyun objelerini takip eder. Ajan en son dizide bulunan oyun objesine geldiğinde tekrar sıfırıncı dizide bulunan oyun objesine gitmesi için navigasyon ağ ajan elemanında bulunan "AutoRepath" özelliğini kullanarak hareketine devam eder. Dizi şeklinde oluşturulan yol nokta yapısı oyun alanına keyfi olarak yerleştirilir. Şekil 3.4'te Unity 3D'de tasarlanan yol nokta yapısı sunulmuştur.



Şekil 3.4: Unity 3D'de yol nokta yapısı.

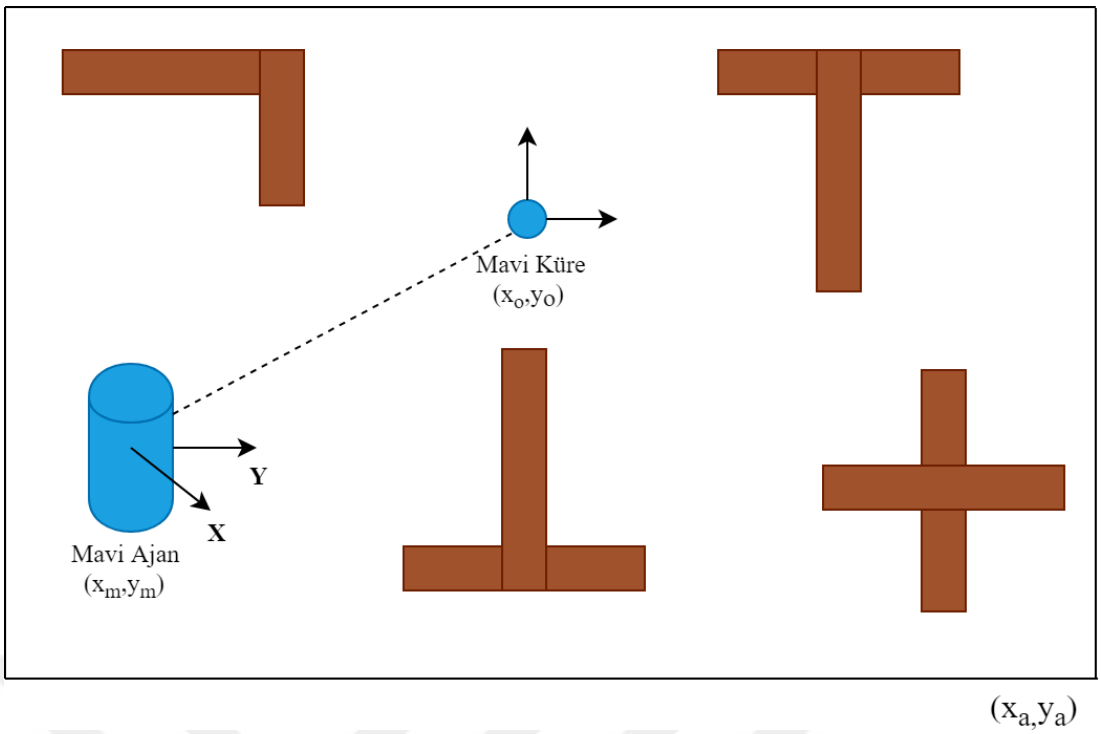
Şekil 3.4'te mor küreler ajanın yol takibi için izleyeceği yol nokta yapısını, mavi silindir ise oyun da bulunan mavi ajanı temsil etmektedir. Mavi ajan mor kürelere doğru sırasıyla hareket eder. Böylelikle yol nokta yapısı ile kovalayan ajanlar oyun alanı içerisinde otonom hareket etmektedir. Şekil 3.5'te farklı bir açıdan açıklanmaya çalışılmıştır.



Şekil 3.5: Yol nokta yapısı.

Şekil 3.5'de gösterilen mor çemberler yol nokta yapısını, mavi silindir ise mavi ajanı temsil etmektedir. Mavi ajan dizi şeklinde tanımlanan çemberlerde bulunan sayıları küçükten büyüğe doğru takip etmekte ve en büyük numaralı çembere geldiğinde dizinin başında bulunan çembere yani sıfır numaralı çembere giderek hareketine devam etmektedir. Mavi ajan sıralı oyun objelerine giderken navigasyon ağlarını ve A* algoritmasını kullanarak en kısa yolu denklem 3.3'te gösterildiği gibi hesaplamakta ve mor çemberlere tanımlanan hız değerinde gitmektedir.

Bu yapının getirdiği olumsuzluk ise kendisini dinamik olarak güncellemediği yani sabit kaldığı için kovalayan ajanlar sadece yol nokta yapısı için etiketlenmiş oyun objelerini takip etmekte ve oyun alanının diğer bölgelerine gitmemektedir. Bu olumsuzluğu ortadan kaldırmak için Unity 3D'de bulunan sert gövde elemanı ve çarpışma elemanı kullanılarak dinamik yol yapısı gerçekleştirilmiştir. Bu dinamik yapıda sadece bir tane oyun objesi kullanılmış ve bu oyun objesinin konumu güncellenerek dinamik yol nokta yapısı elde edilmiştir. Şekil 3.6'da dinamik yol nokta yapısı sunulmuştur.



Şekil 3.6: Dinamik yol nokta yapısı.

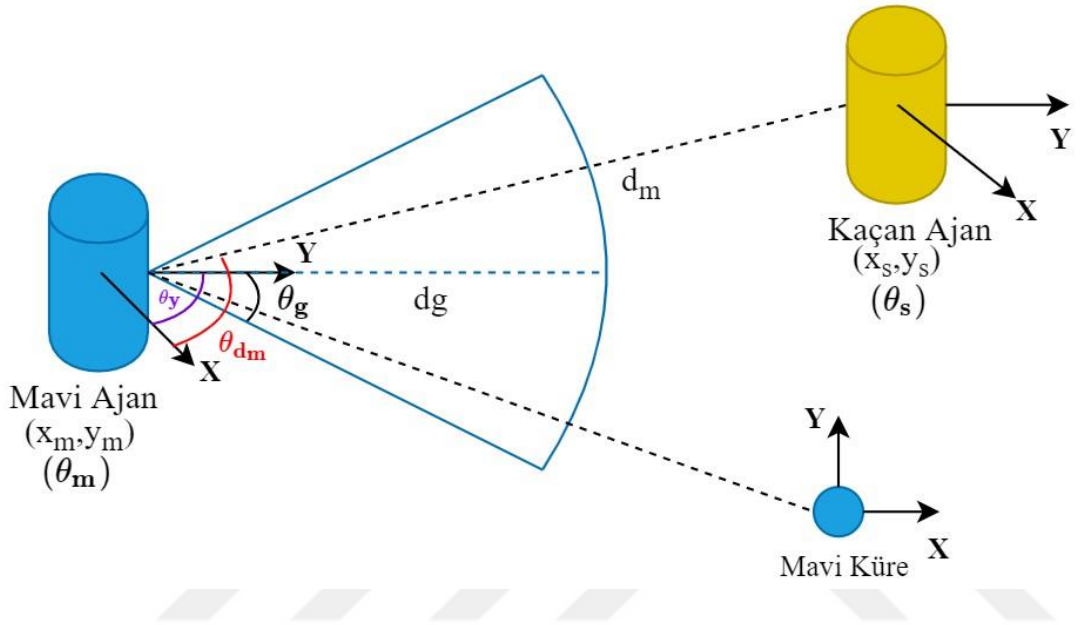
Şekil 3.6'da mavi ajan ve dinamik yol nokta yapısı için tanımlamış mavi küre gösterilmiştir. Şekilde gösterilen kahverengi bloklar ise oyun içerisindeki engelleri temsil etmektedir. Mavi ajanın ve mavi kürenin yapısında sert gövde elemanı ve çarpışma elemanı bulunmaktadır. Mavi ajan navigasyon ağlarını ve A* algoritmasını kullanarak mavi küreye doğru en kısa yoldan harekete geçer. Mavi ajan, mavi küreye temas ettiğinde fizik motoru devreye girip mavi kürenin (x_o, y_o) koordinat değeri için, oyun alanının koordinat değeri olan (x_a, y_a) değerlerinden rastgele iki sayı üretir. Bu üretilen sayılardan birincisi mavi kürenin x ekseni değeri, ikinci üretilen sayı ise y ekseninin değeridir. Bu nedenle mavi ajan mavi küreye her temas ettiğinde, mavi kürenin oyun alanı içerisinde x ve y konumu düzenli olarak güncellenmektedir. Böylece dinamik yol nokta yapısı oluşturulmuş olup mavi ajanın oyun alanı içerisinde tüm bölgelere gitme olasılığı sağlanmış olur.

Eğer mavi kürenin x ve y koordinat değerleri oyunda bulunan engellerin koordinatları ile çakışırsa mavi kürenin engellerin içerisine girmesini engellemek için engellere çarpışma elemanı eklenmiştir. Burada dikkat edilmesi gereken nokta mavi küre dinamik çarpıştırıcı olduğu için çarpışma sonucu oyun alanına geri döner, ancak engeller statik çarpıştırıcı olduğu için hareket edemez, konumu sabit kalır. Tasarlanan bu dinamik yol nokta yapısı oyunda bulunan kırmızı, mavi, turuncu ve pembe ajan için

kullanılmıştır. Böylece dört adet kovalayan ajan oyun alanında rastgele hareket ederek oyun alanında tüm bölgelere gitme olasılığı artırılmıştır.

3.2.2 Unity 3D görüş hattı yapısı

Tasarlanan kaçma-kovalama oyununda kovalayan ajanlara oyun alanı içerisinde belli bir görüş menzili oluşturmak için görüş hattı yapısı oluşturulmuştur. Şekil 3.7'de görüş hattı yapısı sunulmuştur.



Şekil 3.7: Görüş hattı yapısı.

Şekil 3.7'de üzerinde görüş hattı yapısını anlatmak için oyunda bulunan mavi ajan ve kaçan ajan kullanılmıştır. Görüş hattı yapısı temel olarak mavi ajanın sahip olduğu ileri yön ekseninde (y ekseni) üçgen şeklinde tanımlanmış geometridir. Görüş hattı yapısı d_g mesafesi ve θ_g açısına göre belirlenir. d_g mesafesi görüş hattı olarak düşünülen üçgen şeklin yüksekliği, θ_g açısı ise tepe açısıdır.

Şekil 3.7 üzerindeki d_m mesafesi iki ajan arasındaki mesafe olup, oyun içindeki hareketine göre denklem 3.7 ile hesaplanmaktadır. Denklemde i indisi yerine k, m, t ve p indisleri yerleştirildiğinde her bir kovalayan ajan ile kaçan ajan arasındaki mesafe hesaplanmış olur.

$$d_i = \sqrt{(x_s - x_i)^2 + (y_s - y_i)^2} \tag{3.7}$$

Şekil 3.7 üzerinde gösterilen θ_g açısı ise d_m mesafesi ile ajanın ileri yön koordinatı olan y ekseni arasında kalan açı değerine eşittir. θ_g açı değeri hesabı ise denklem 3.8' de gösterilmiştir.

$$\theta_g = tan^{-1}\left(\frac{M1 - M2}{1 + M1M2}\right) \tag{3.8}$$

3. 8 denkleminde yer alan M_1 kovalayan ajanın y ekseninin eğimi olup, x ekseni ile yaptığı θ_y açı değerinin tanjantına eşittir.

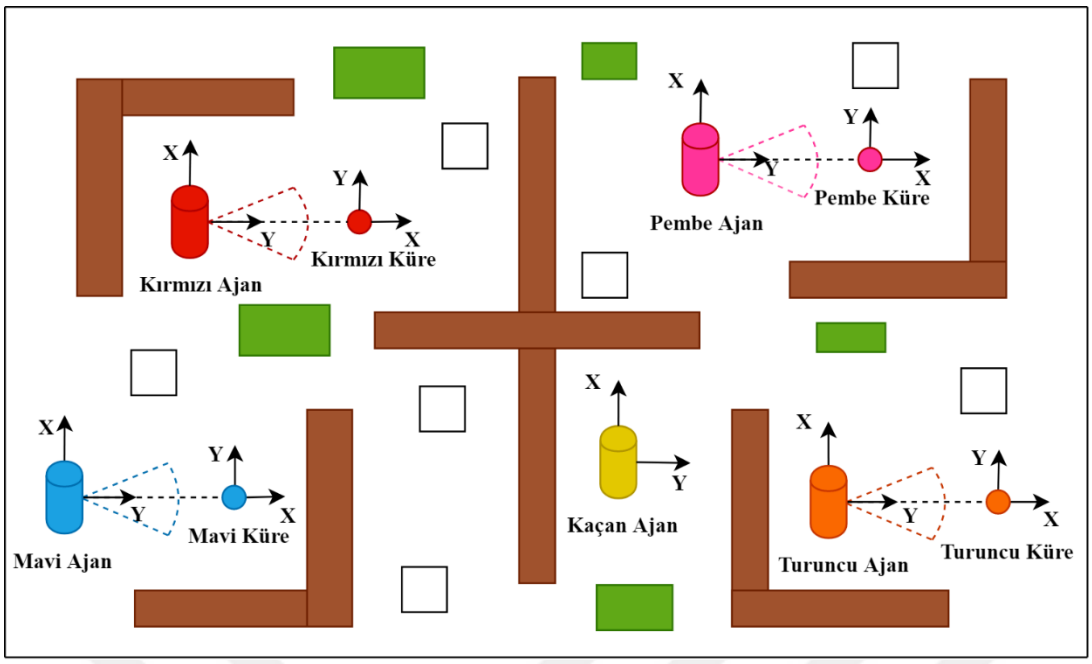
$$M_1 = \tan(\theta_y) \tag{3.9}$$

M_2 ise d_m doğrultusunun eğimini olup, x ekseni ile yaptığı θ_{d_m} açısının tanjant değerine eşittir.

$$M_2 = \tan(\theta_{d_m}) \tag{3.10}$$

Oyun bulunan tüm ajanlar için $d_g = 60m$ ve $\theta_g = 30°$ olarak belirlenip her ajanda aynı menzile sahip görüş hattı yapısı oluşturulmuştur. Eğer kaçan ajan kovalayan ajanların görüş hattına girerse ($d_m \leq 60m$ ve $\theta_g \leq 30°$) kovalayan ajanlar tarafından yakalanmaya çalışılacaktır. Kaçan ajan bu görüş hattı içerisinde bulunmadığı takdirde ($d_m > 60m$ veya $\theta_g > 30°$) kovalayan ajanlar dinamik yol nokta yapısını kullanarak oyun alanı içerisinde hareketine devam edecektir. Oyunda bulunan kâğıtlar kaçan ajan tarafından toplanmaya başladıkça kovalayan ajanların görüş hattı yapısındaki θ_g açısında, çizgisel ve açısal hızlarında artış meydana gelecektir. Böylece hızı ve görüş hattı artan kovalayan ajanların hareketleri agresifleşmeye başlayacaktır.

Kovalayan ajanlar için tasarlanmış dinamik yol nokta yapısının ve görüş hattı yapısının tüm ajanlarda ortak olduğunu belirtmiştik. Dinamik yol nokta yapısı için her bir kovalayan ajana oyun içerisinde devriye gezmesi için kendi renklerinde küre oyun objeleri oluşturulup oyun alanına eklenmiştir. Görüş hattı yapısı ve dinamik yol nokta yapısı kovalayan ajanlara eklendiğinde Şekil 3.8'daki gibi görünüm elde edilir.



Şekil 3.8: Kaçma-kovalama oyunundaki tüm ajanlar.

Şekil 3.8'de oyun alanı temsili olarak gösterilmiştir. Oyun tasarımının nihai hali Şekil 3.3'te sunulmuştur. Şekil 3.8'de ajanların koordinat eksenlerinin ters olmasının sebebi ajanların ileri yön ekseninin y ekseni olması ve ileri yön hareketlerinin gösterilmek istenmesidir. Şekilde gösterilen kahverengi bloklar engelleri, yeşil dikdörtgen oyun objeleri ölüm alanını ve beyaz kare oyun objeleri ise toplamamız gereken kâğıtları temsil etmektedir. İnsan kullanıcı tarafından kontrol edilen kaçan ajan kovalayan ajanlardan kaçarak kâğıtları toplamaya çalışacaktır. Kovalayan ajanlar ise kendi renkleri doğrultusunda oluşturulan küre oyun objesini takip ederek otonom hareket edecektir. Eğer görüş hattına kaçan ajan girerse kovalamaya çalışacaktır.

Tezin bu kısmına kadar kovalayan ajanların ortak özelliklerinden bahsedilmiştir. İlerleyen kısımlarda dört adet kovalayan ajana özel kovalama stratejileri geliştirilmiştir. Kovalayan ajanlara strateji geliştirmek için Unity 3D oyun motorunun özellikleri, "Pacman" oyunundaki hayaletlerin kovalama stratejileri ve bulanık mantık kullanılmıştır.

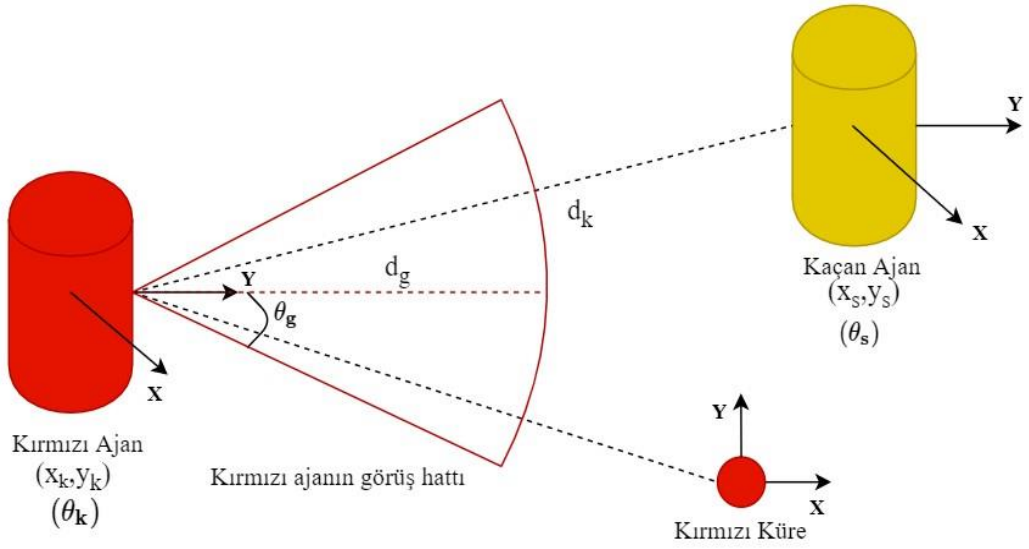
3.2.3 Oyunda bulunan ajanlar

Unity 3D oyun motorunda tasarlanan kaçma-kovalama oyununda bulunan tüm ajanlar silindir oyun objesi kullanılarak tasarlanmıştır. Bu ajanlara çeşitli renkler eklenip, isimleri sahip olduğu renklere göre belirlenmiştir. Oyunda bir adet insan kullanıcı tarafından kontrol edilen kaçan ajan (sarı ajan) ve dört adet kovalayan ajan (kırmızı

mavi, turuncu ve pembe ajan) bulunmaktadır. Kovalayan ajanlardan kırmızı, mavi ve turuncu ajana strateji geliştirmek için Unity 3D oyun motorunun özellikleri ve "Pacman" oyunundaki hayaletlerin yakalama stratejileri kullanılmıştır. Pembe ajana strateji geliştirmek için bulanık mantık kullanılmıştır.

3.2.3.1 Birbirleri ile haberleşen ajanlar: Kırmızı, mavi ve turuncu ajan

Oyunda bulunan kırmızı ajan için strateji geliştirilirken "Pacman" oyununda kırmızı hayalet olan "Blinky'nin" "Pacman'ı" yakalamak için kullandığı strateji tercih edilmiştir. "Blinky" "Pacman'ı" yakalamak için en kısa yolu tercih ederek "Pacman'ı" yakalamaya çalışmaktadır. Kırmızı ajan Unity 3D elemanları olan navigasyon ağ ajan elemanı, navigasyon ağları ve A* algoritmasını kullanarak kaçan ajanı en kısa yoldan yakalamaya çalışır. Eğer kaçan ajan, kırmızı ajanın görüş hattına girerse onu yakalamaya çalışmaktadır. Kaçan ajan görüş hattından çıktığında ise kırmızı ajan kovalamayı bırakıp, dinamik yol nokta yapısını kullanarak otonom hareket eder. Şekil 3.9'da kırmızı ajanın strateji yapısı gösterilmiştir.

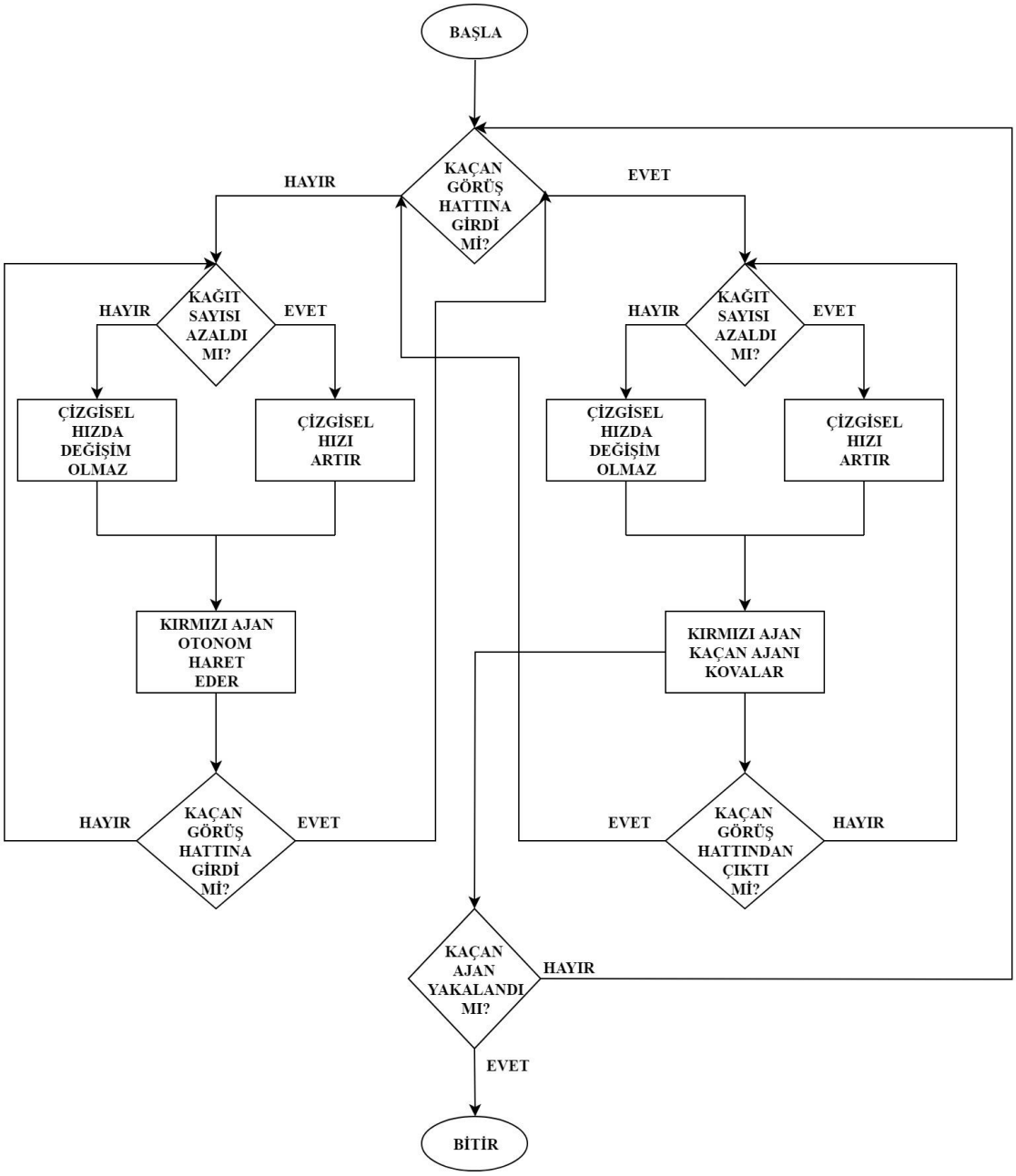


Şekil 3.9: Kırmızı ajan strateji yapısı.

Şekil 3.9'da gösterilen kırmızı ajan ile kaçan ajan arasındaki d_k mesafesi denklem 3.7'de i indisi yerine k indisi yerleştirilerek bulunur. Bu mesafe $60m$'ye eşit ve altında düştüğünde $(d_k \leq 60m)$ ve $\theta_g \leq 30°$ olduğunda kırmızı ajan kaçan ajanı kovalamaya başlayacaktır. Eğer mesafe $60m$ üzerine çıkarsa veya açı değeri $30°$ büyük olursa kırmızı ajan şekil üzerinde gösterilen kırmızı küreye doğru hareket ederek oyun içerisinde otonom hareketine devam edecektir.

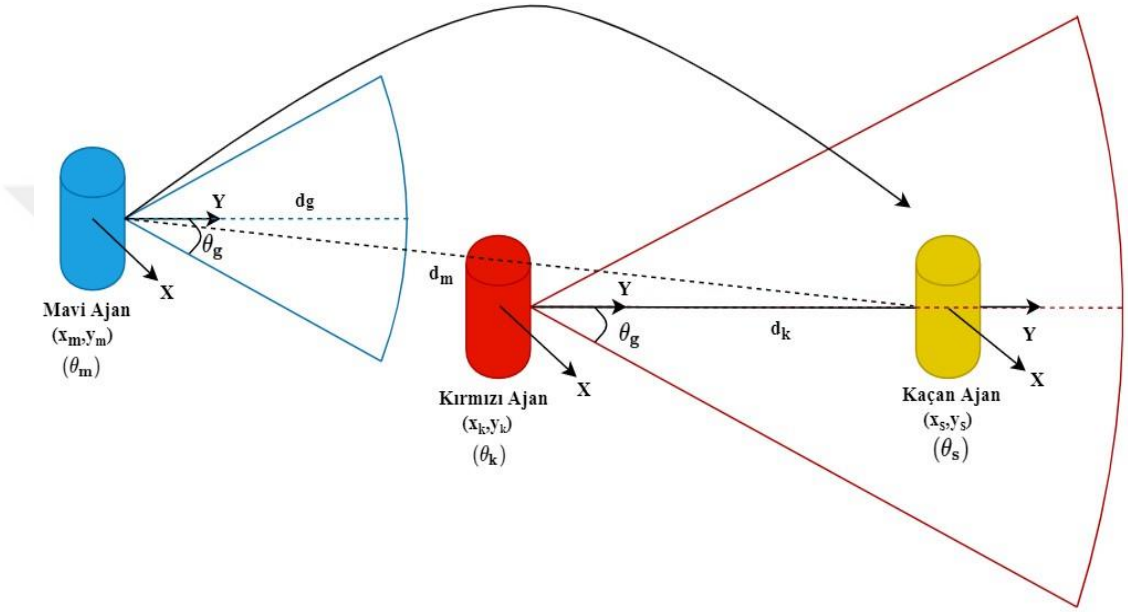
Ayrıca oyunda bulunan kâğıtlar kaçan ajan tarafından toplandıkça kırmızı ajanın çizgisel hızı (V_k) artmaya başlayacaktır.

Kırmızı ajanın sahip olduğu elemanlar ise navigasyon ağ ajan elemanı haricinde dönüşüm elemanı, sert gövde elemanı ile fizik motorundan faydalanma, $m_k = 1\ \text{kg}$ kütleye sahip olma ve yer çekimi kuvvetinden etkilenmedir. Çarpışma elemanı olarak kapsül çarpışma elemanına sahiptir. Kod elemanı ile kırmızı ajana kovalaması için gerekli strateji geliştirilmiştir. Şekil 3.10'da ise geliştirilen stratejinin akış diyagramı sunulmuştur.



Şekil 3.10: Kırmızı ajan stratejisinin akış diyagramı.

Mavi ajan için kırmızı ajanın sahip olduğu strateji yapısı temel alınarak strateji geliştirilmiştir. Mavi ajanın sahip olduğu stratejide kaçan ajan mavi ajanın görüş hattına girdiğinde, A* algoritmasını kullanarak sahip olduğu çizgisel hız ile yakalamaya çalışacaktır. Kaçan ajan görüş hattından çıktığında ise mavi ajan oyun alanı içerisinde mavi küreyi takip etmektedir. Mavi ajan, mavi küre ile her temas ettiğinde mavi kürenin konumu oyun alanının sahip olduğu koordinat değerleri arasında güncellenecektir. Böylece, mavi ajanın oyun alanı içerisinde tüm bölgelere gitme ihtimali olacaktır. Bahsedilen bu yapı Şekil 3.11'de sunulmuştur.

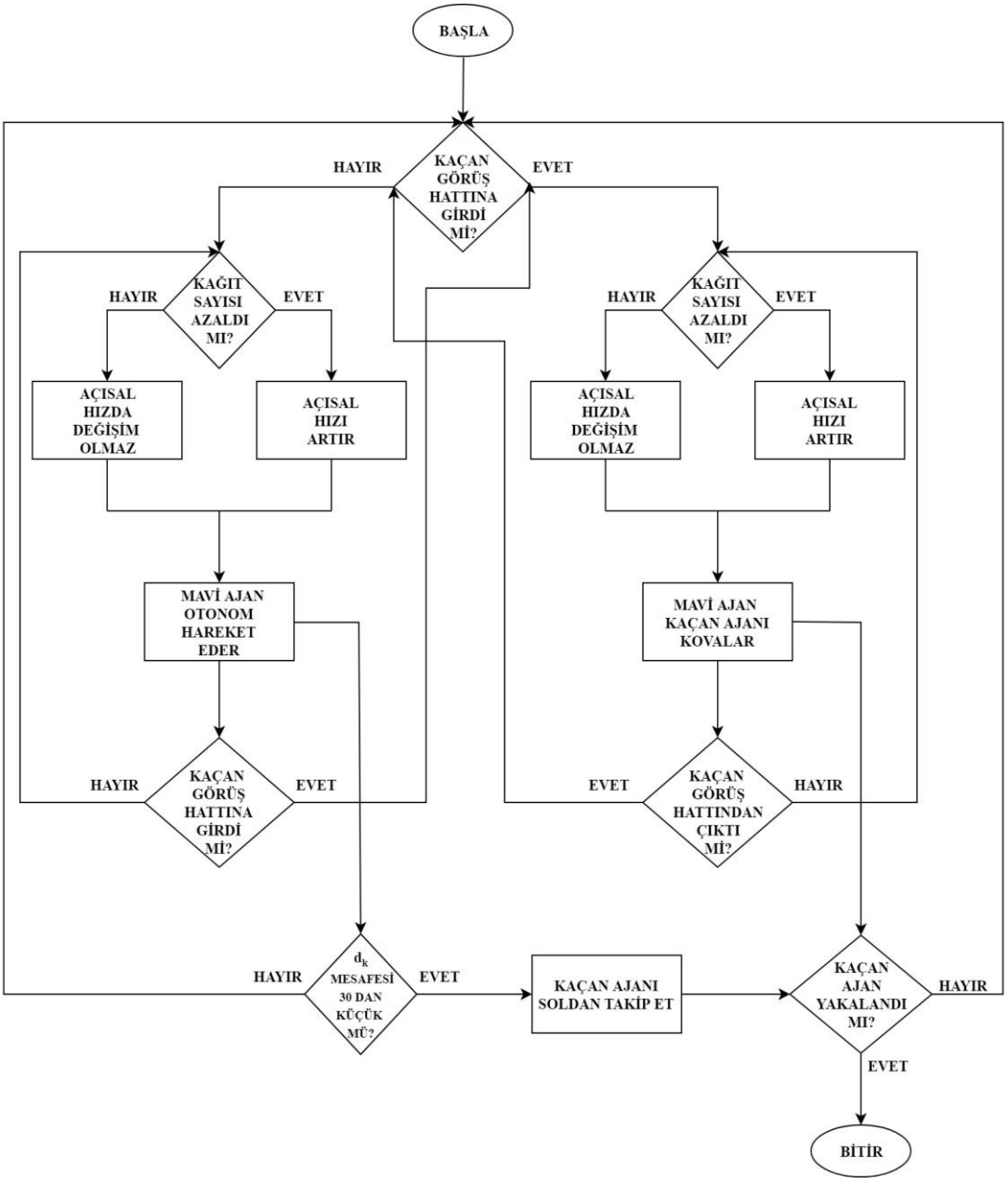


Şekil 3.11: Mavi ajan strateji yapısı.

Mavi ajan ayrıca kırmızı ajan ile haberleşerek, kırmızı ajan ile kaçan ajan arasındaki d_k mesafesini hafızasında bulundurmaktadır. Kaçan ajan ile kırmızı ajan arasındaki mesafe belli bir değerin altına düştüğünde ($d_k < 30\ m$) ise kırmızı ajan kaçan ajanı arkadan, mavi ajan ise soldan yakalamaya çalışacaktır. Soldan yakalama için Ebeveyn-Evlat ilişkisinden yararlanılarak kaçan ajanın hiyerarşisine mavi renkli bir oyun objesi sol kısmına yerleştirildi. Böylece mavi oyun objesi kaçan ajanın hiyerarşisine girdiği için kaçan ajanın tüm özelliklerine sahip olmuş olur ve hareketi kaçan ajana bağımlı hale gelmiştir.

Mavi ajan kaçan ajanı soldan takibe başladığında aralarındaki d_m mesafesi belirli süre sonra azalmaya başlar, böylece kaçan ajan mavi ajanın da görüş hattına girmiş olup iki ajan tarafından kovalanır. Oyun alanı içerisinde kâğıt sayısı azaldığında mavi ajanın açısal hızı artmaktadır.

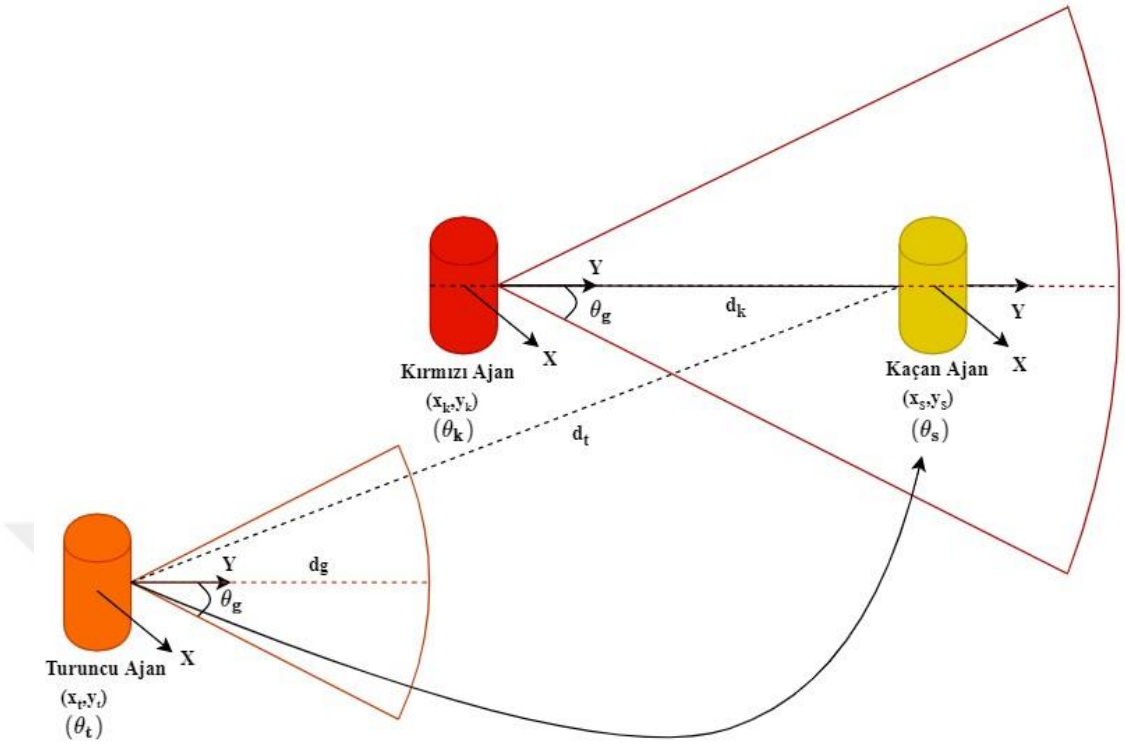
Mavi ajanın sahip olduğu elemanlar ise yukarıda bahsedildiği gibi otonom hareketi için navigasyon ağ ajan elemanı, dönüşüm elemanı, sert gövde elemanı ile $m_m = 1\,kg$ kütleye sahip olup, yer çekimi kuvvetinden etkilenmektedir. Oyun alanında bulunan tüm oyun objeleri ile etkileşime girmek için kapsül çarpışma elemanına ve yukarıdaki anlatılan stratejileri geliştirmek için kod elemanına sahiptir. Şekil 3.12'de ise mavi ajan için geliştirilen stratejinin akış diyagramı gösterilmiştir.



Şekil 3.12: Mavi ajan stratejisinin akış diyagramı.

Turuncu ajanın sahip olduğu stratejide; kaçan ajan turuncu ajanın görüş hattına girdiğinde A* algoritmasını kullanarak sahip olduğu çizgisel hız ile kaçan ajanı yakalamaya çalışacaktır. Kaçan ajan görüş hattından çıktığında ise turuncu ajan oyun

alanı içerisinde tanımlı turuncuyu küreye doğru hareket ederek oyun alanı içerisinde otonom hareket edecektir. Şekil 3.13'te turuncu ajanın stratejisi sunulmuştur.

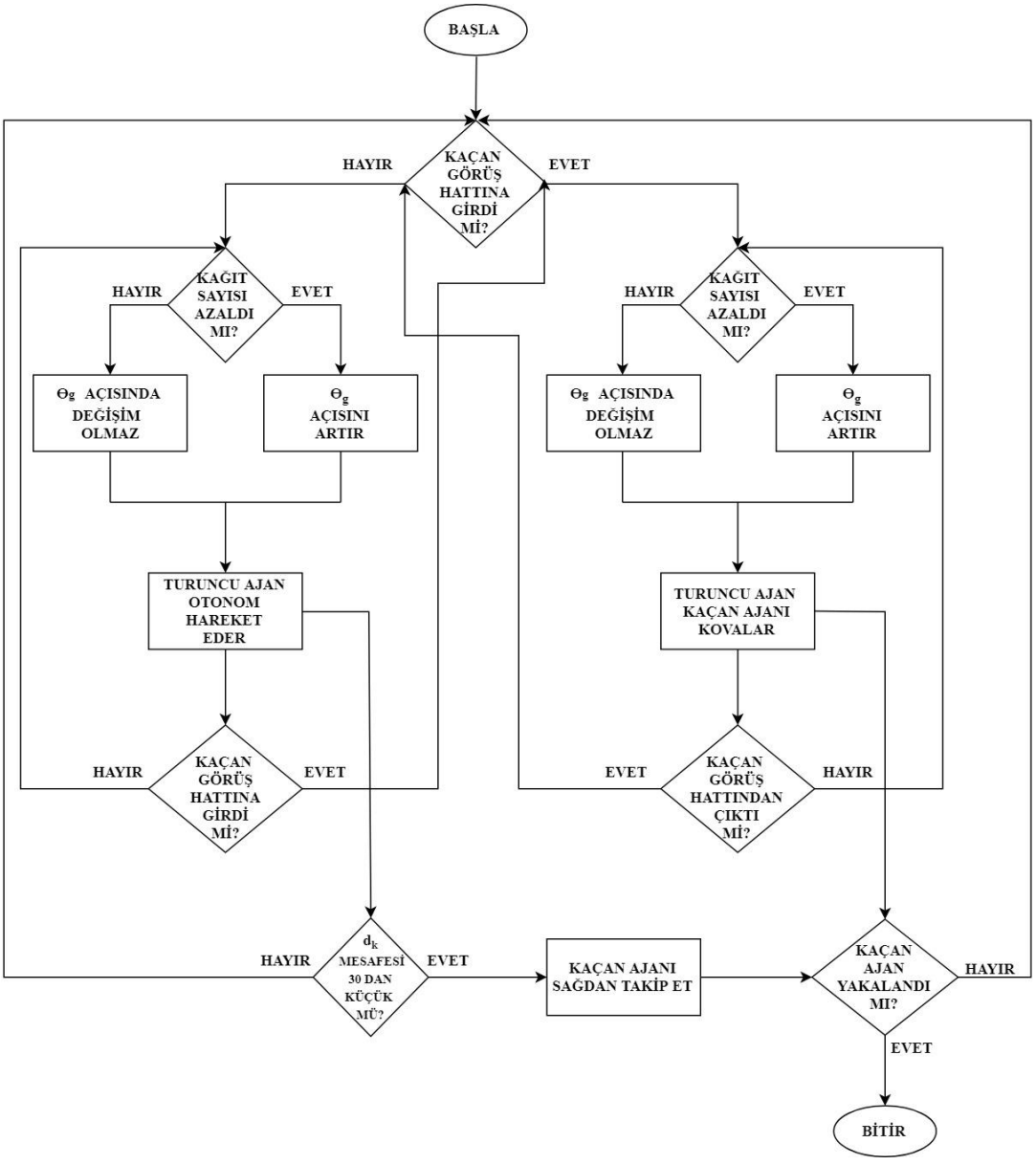


Şekil 3.13: Turuncu ajan stratejisi yapısı.

Turuncu ajan, kırmızı ajan ile haberleşerek kaçan ajan ile kırmızı ajan arasındaki d_k mesafesi hakkında bilgi sahibi olmaktadır. Eğer d_k mesafesi belirli değerin altına düşerse ($d_k < 30m$) ise turuncu ajan kaçan ajanı sağdan takip etmeye başlayacaktır. Turuncu ajanın kaçan ajanı sağdan takip etmesi için Ebeveyn-Evlat ilişkisinden yararlanılarak kaçan ajanın gövdesine $+x$ yönünde turuncu renkli oyun objesi yerleştirilmiştir. Böylelikle turuncu ajan bu objeye doğru hareket ederek kaçan ajanı sağ taraftan takip eder. Turuncu ajan ile kaçan ajan arasındaki d_t mesafesi azaldığı ($d_t < 60m$) için kaçan ajan turuncu ajanın görüş hattına girer ve ajan tarafından yakalanmaya çalışılır.

Eğer d_k mesafesi $30m$ den büyük olursa turuncu ajan yapacağı aksiyonu d_t mesafesine göre karar verecektir. Bu mesafe $60m$ ye eşit ve altında ise kaçan ajanı görüş hattından çıkana kadar kovalar, $d_t > 60m$ olduğunda ise turuncu ajan oyun alanında otonom olarak hareket eder. Ayrıca oyun alanı içerisindeki kâğıt sayısı azaldıkça turuncu ajanın sahip olduğu görüş hattı açısı (θ_g) artmaktadır. Bu açı değeri arttıkça turuncu ajanın görüş alanı büyümektedir.

Turuncu ajanın sahip olduğu elemanlar ise yukarıda bahsedildiği gibi otonom hareketi için navigasyon ağ ajan elemanı, dönüşüm elemanı, sert gövde elemanı ile $m_t = 1\,kg$ kütleye sahip olup, yer çekimi kuvvetinden etkilenmektedir. Oyun alanında bulunan tüm oyun objeleri ile etkileşime girmek için kapsül çarpışma elemanına ve yukarıda anlatılan stratejileri geliştirmek için kod elemanına sahiptir. Şekil 3.14'te turuncu ajanın sahip olduğu stratejinin akış diyagramı sunulmuştur.



Şekil 3.14: Turuncu ajan stratejisinin akış diyagramı.

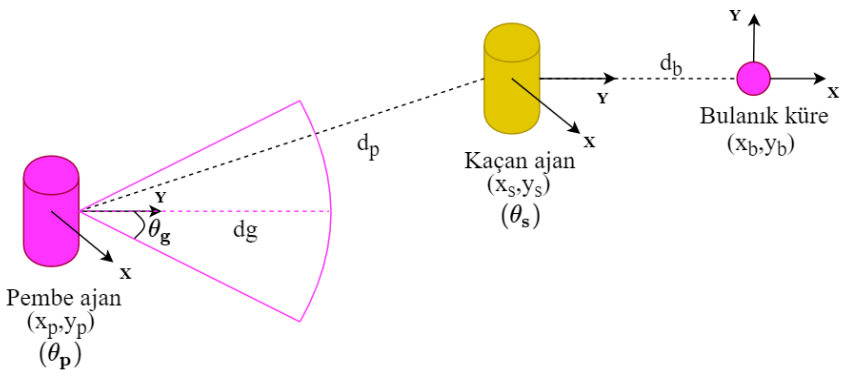
Tasarladığımız kırmızı, mavi ve turuncu ajan ile oyun motorunun sağladığı özellikler doğrultusunda bir kovalama stratejisi geliştirdik. Bu strateji ile her kovalayan ajan görüş hattına giren kaçan ajanı kovalayacaktır. Ayrıca mavi ve turuncu ajan için, kırmızı ajan ile koordinat haberleşmesi sağlanmıştır. Böylece $d_k < 30m$ olduğu

takdirde, mavi ajan soldan ve turuncu ajan sağdan takip ederek kaçan ajanı yakalamaya çalışacaktır.

3.2.3.2 Bulanık mantık tabanlı ajan: Pembe ajan

Bulanık mantık tabanlı ajan olan pembe ajan kendisi için tanımlanmış pembe küre ile oyun içerisinde otonom olarak hareket etmekte ve görüş hattı yapısı sayesinde kaçan ajanı kovalamaktadır.

Pembe ajana strateji geliştirirken sahip olduğu görüş hattı yapısı kullanılmıştır. Diğer kovalayan ajanlar görüş hattına kaçan ajan girdiğinde yakalamak için P veya PD kontrolör yapısını kullanarak kovalamaktadır. Ancak kovalayan ajan ile kaçan ajan arasındaki mesafe değiştiğinde mavi ve turuncu ajan aynı hızla kovalamakta, kırmızı ajan ise kaçan ajanla olan mesafesini azaltamamaktadır. Pembe ajanda tasarlanan strateji ile diğer ajanlarda ortaya çıkan bu sorunun ortadan kaldırılması amaçlanmıştır. Geliştirilen stratejide kaçan ajan, pembe ajanın görüş hattına girdiğinde kovalaması gereken hıza aralarındaki mesafeye göre karar vermesi gerekir. Pembe ajan ile kovalayan ajan arasındaki d_p mesafesi azaldığında V_p hızı da azalmalı, d_p mesafesi arttığında V_p hızı da artmalıdır. Diğer bir deyişle pembe ajanın, insanın düşünme ve karar verme davranışını taklit ederek V_p hızına karar vermesi gerekir. Kaçan ajan, pembe ajanın görüş hattına girdiğinde diğer ajanlarda olduğu gibi A* algoritmasını kullanarak en kısa yoldan kovalayacaktır, ancak pembe ajan, kaçan ajanı sabit bir hızla değil bulanık mantık yöntemini kullanarak hızını kontrol ederek kovalayacaktır. Pembe ajanın kaçan ajanı yakalaması için Ebeveyn-Evlat ilişkisi ile ileri yön koordinatında bir oyun objesi eklenmiştir. Pembe ajan kaçan ajanda bulunan bulanık kürenin konumu (d_b) ve V_p hızını d_p değişime göre kontrol ederek kaçan ajanı yakalayacaktır. Bahsedilen yapı Şekil 3.15'te sunulmuştur.



Şekil 3.15: Pembe ajan strateji yapısı.

Şekil 3.15'te gösterilen d_p uzaklığı ise denklem 3.7'de sunulan denklemde i indisi yerine p indisi yerleştirilerek hesaplanmış olur.

Bulanık küre Ebeveyn-Evlat ilişkisi doğrultusunda kaçan ajanın hiyerarşisi altına girmiştir. Bulanık küre hareketine ise kısıtlama getirilerek kaçan ajanın ileri yön doğrultusu olan y ekeninde hareket etmesi sağlanmıştır. Bulanık kürenin x ekseni ise kaçan ajan ile aynı koordinat değerine sahip olup sabittir, ancak y ekseni pembe ajan için tasarlanan bulanık sistem çıkışına göre değişmektedir.

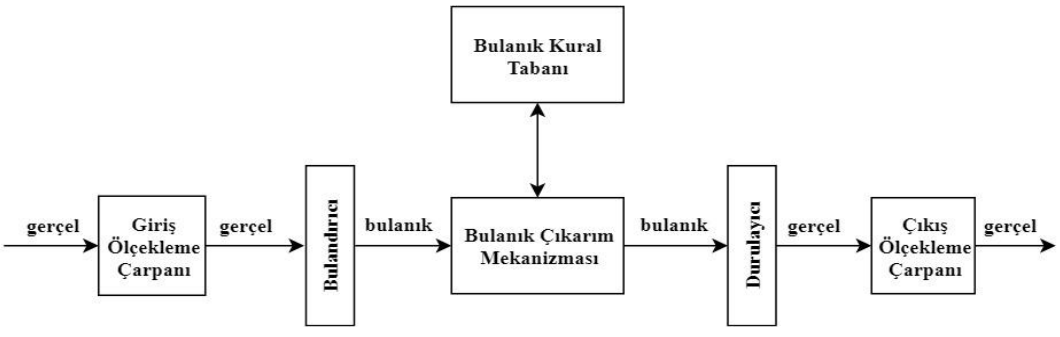
$$(x_b, y_b) = \left(x_b = x_p, y_b = y_p + d_b\right) \tag{3.11}$$

Pembe ajanın sahip olduğu elemanlar ise otonom hareketi için navigasyon ağ ajan elemanı, dönüşüm elemanı, sert gövde elemanı ile $m_p = 1\ kg$ kütleye sahip olup, yer çekimi kuvvetinden etkilenmektedir. Oyun alanında bulunan tüm oyun objeleri ile etkileşime girmek için kapsül çarpışma elemanına sahiptir. Ayrıca kod elemanı ile bulanık mantık sistemler geliştirilmiştir.

Pembe ajan için strateji geliştirme

Pembe ajana strateji geliştirmek için bulanık mantık kullanılmıştır. Bulanık mantık ile tasarlanan iki adet bulanık sistem iki girişli ve tek çıkışlı olup, pembe ajanın V_p çizgisel hızı ve d_b mesafesini kontrol etmektedir.

Bulanık kümeler ve bunlarla ilişkisi olan matematiksel yapı içeren sistemlere bulanık sistem denir [30]. Bulanık sistemlerin çok büyük kısmı "EĞER-O HALDE" kuralları ile tanımlanır[30]. Bu tür sistemler, "kural tabanlı bulanık sistemler" olarak adlandırılır. Genel olarak kontrol sistem tasarımı [31], robotik [32,33], görüntü işleme [34], karar verme [35], bilgisayar oyunları[36-40], modelleme [41] ve kelimelerle hesaplama [42] gibi çeşitli alanlarda kullanılır. Bir bulanık sistemin içyapısında bulunan birimler Şekil 3.16'da gösterilmiştir.



Şekil 3.16: Bulanık sistemin içyapısı.

Bir bulanık sistemin içyapısında dört adet birim bulunmaktadır. Bunlar sırasıyla Bulandırıcı, Bulanık Çıkarım Mekanizması, Bulanık Kural Tabanı ve Durulayıcı birimleridir. Bulandırıcı, gerçel değerli sistem girişini bulanık kümelere dönüştüren ve böylece bunların bulanık çıkarım mekanizmasında değerlendirilmesine olanak sağlayan birimdir. Bulanık çıkarım mekanizması bulanık kümeleri bulanık kural tabanında tanımlanan "EĞER-O HALDE" kuralları doğrultusunda çıkışında bulanık kümeler üretir. Elde edilen bulanık kümeler durulaştırma birimine giriş olarak verilip çıkışta gerçel sayılara dönüştürülür.

Bulanık sistemlerde değişkenler arasındaki ilişkiler bulanık EĞER öncül (antecedent) önerme O HALDE sonuç (consequent) önerme şeklindeki "EĞER – O HALDE" bulanık kuralları ile ifade edilir. Bu kuralın genel biçimi şu şekildedir:

EĞER öncül önerme ise O HALDE sonuç önerme.

Öncül önerme, her zaman "x, A'dır" şeklinde bir bulanık önermedir. Burada x dilsel değişken ve A ise bir dilsel terimdir. Önermenin doğruluk değeri (0 ile 1 arasında gerçel sayı) x ve A arasındaki uyumun (benzerliğin) derecesine bağlıdır. Sonuç önermeye bağlı olarak iki ana tip kural tabanlı model oluşturulur:

i) Mamdani Bulanık Model: Öncül ve sonuç önermelerin her ikisi de bulanık önermelerdir[43,44].
ii) Takagi-Sugeno-Kang Bulanık Model: Öncül bulanık önerme, sonuç ise belirgin fonksiyondur[45].

Tez kapsamında, bulanık sistem tasarımı için "Takagi-Sugeno-Kang" bulanık model kullanılmıştır. Bu modeli seçmemizin nedeni "Mamdani" modelde bulunan öncül ve sonuç kısmının bulanık önerme olmasının aksine, "Takagi-Sugeno-Kang" modelde öncüller bulanık olup, sonuç ise belirgin ve kesin bir fonksiyondur. Dolayısıyla bu model hem matematiksel, hem de dilsel beslemeli olarak görülebilir. Denklem 3.12 'de "Takagi-Sugeno-Kang" modelindeki genel kural yapısı verilmiştir.

$$R_i : EĞER\ x = A_i\ ve\ \ y = Bi\ ise\ ,O\ HALDE\ z_i = f_i(x,y) \quad i = 1,2,3 \dots 9 \qquad (3.12)$$

Burada x ve y kesin değerli girişleri , A_i ve B_i dilsel değişkenlerdir. Kuralda yer alan $f_i(x,y)$ ise aynı tip yapıda olup, parametreleri değişken olan çıkışı ve i ise kural sayısını göstermektedir. Bulanık kural tabanında 9 adet kural bulunduğu için i parametresi 9 da sonlandırılmıştır.

Tez kapsamında tasarlanan bulanık sistemde durulaştırma için ağırlıklı ortalama yöntemi seçilmiştir. Bulanık sistemin çıktısı ise aşağıdaki denklem ile hesaplanmıştır. Bulanık sistem çıkışı z_i ise;

$$z_i = \frac{\sum_{i=1}^{p} f_i w_i}{\sum_{i=1}^{p} w_i} \tag{3.13}$$

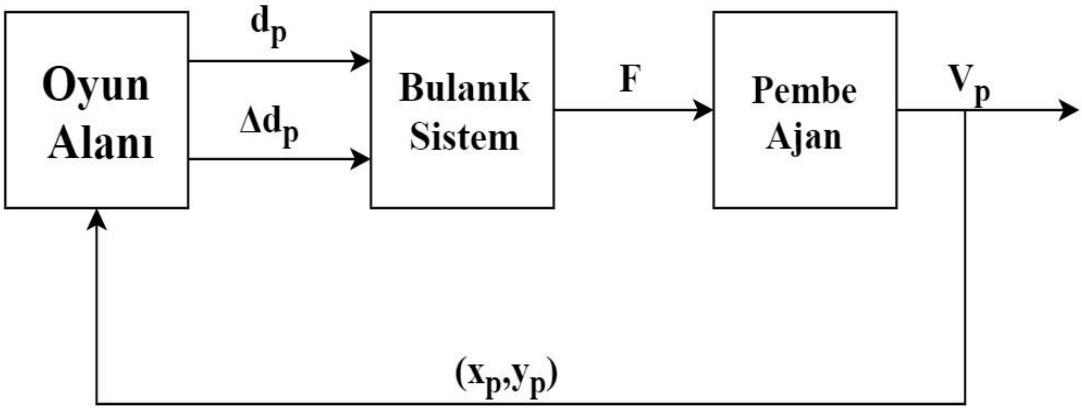
Denklem 3.13'te bulunan f_i her bir kural için elde edilen çıkış değeri, w_i ise giriş değerlerinin ait olduğu üyelik fonksiyonlarının ateşleme değerleri ve p ise toplam kural sayısıdır. Denklem 3.14'te w_i verilmiş olup, μ_i ise girişlerin üyelik fonksiyonuna aitlik derecesini belirtir.

$$w_i = \mu_{A_i}(x)\mu_{B_i}(y) \tag{3.14}$$

V_p hızı için bulanık sistem tasarımı

Pembe ajanın çizgisel hızını kontrol etmek için denklem 3.12'de verilen kural yapısı kullanılarak iki girişli ve bir çıkışlı bulanık sistem tasarımı yapılmıştır. Bulanık kural tabanında ise 9 adet kural bulunmaktadır. Bulanık sistemlerin girişleri Şekil 3.15'te gösterilen pembe ajan ile kaçan ajan arasındaki mesafe olan d_p ve Δd_p'dir. Tasarlanan sistem ile pembe ajan, kaçan ajanı sabit hız yerine aralarındaki mesafenin değişimine göre çizgisel hızını kontrol edecektir.

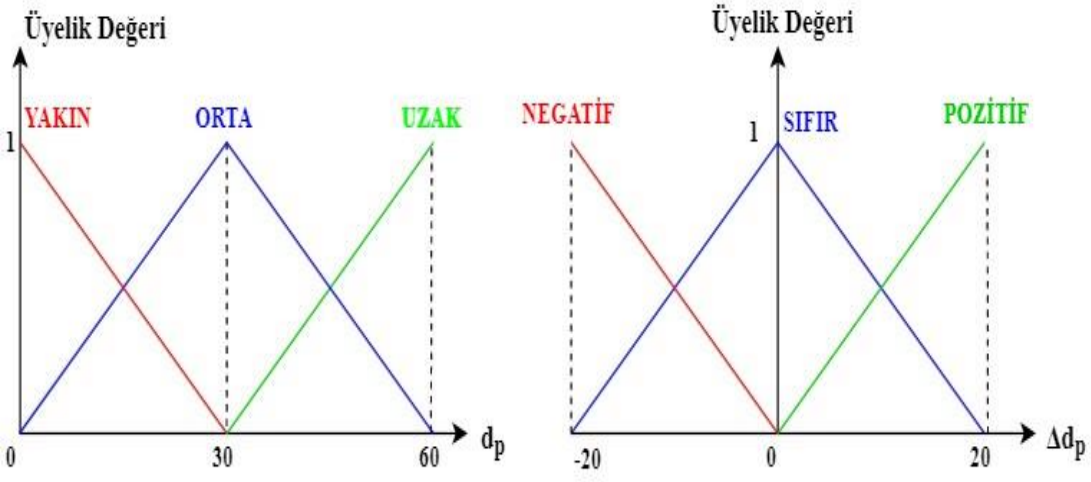
Pembe ajan da tezin ikinci kısmında bahsettiğimiz çeşitli Unity 3D araç kutularına sahiptir. Kontrol araç kutusunda bulunan navigasyon ağ ajan elemanı ve sert gövde elemanlarının kendilerine ait hız değerleri bulunmaktadır. Şekil 2.4'te gösterilen navigasyon ağ ajan elemanında bulunan "Speed" değerine herhangi bir sayısal değer yazarsak, kovalayan ajan navigasyon ağı üzerinde verilen hız değeri ile hareket eder. Şekil 2.7'de gösterilen sert gövde elemanı yapısında ise hız değeri bulunmamaktadır. Ancak bu eleman Unity 3D motorunda bulunan fizik motoru ile birlikte çalışmaktadır. Kütlesi olan pembe ajana kuvvet uyguladığımızda, uygulanan kuvvet doğrultusunda ivme ve hız değerleri elde edilir. Böylelikle tasarlanan bulanık sistem çıkışında pembe ajana uygulamamız gereken kuvvet değeri kontrol edilmektedir. Bu kuvvet değeri fizik motoruyla pembe ajana giriş olarak verilip, çıkışta çizgisel hız değeri elde edilmiştir. Çizgisel hız değeri için tasarlanan kontrol şeması Şekil 3.17'de verilmiştir.



Şekil 3.17: Pembe ajanın çizgisel hız kontrol blok diyagramı.

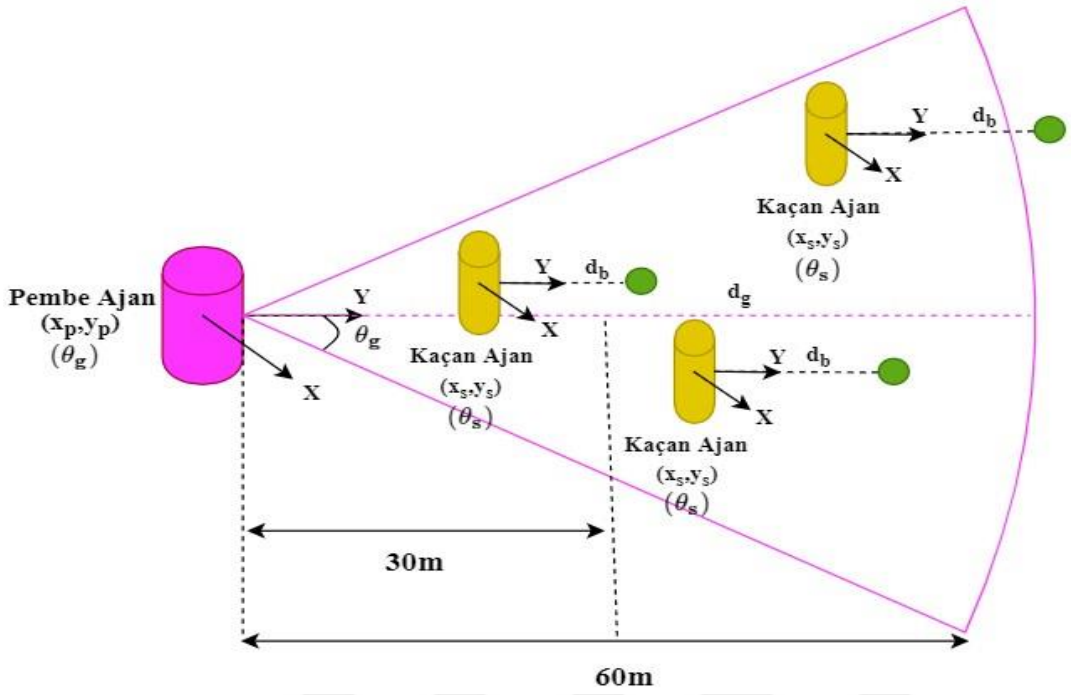
Şekil 3.17'de gösterilen blok diyagramında bulunan oyun alanı bloğu bulanık sisteme iki adet giriş sağlamaktadır. Bu girişler pembe ajan ile kaçan ajan arasındaki d_p mesafesi ve Δd_p'dir. Bulanık sistem çıkışında ise elde ettiğimiz sayı değeri pembe ajana kuvvet olarak etki etmektedir. Pembe ajana giriş olarak kuvvet değeri uygulandığında yapısında bulunan sert gövde elemanı sayesinde çizgisel hıza dönüşmektedir.

Şekil 3.17'de gösterilen bulanık sistemde iki adet giriş ve bu girişlerin üç adet üyelik fonksiyonları bulunmaktadır. Bulanık sistemin birinci girişi olan d_p mesafesinin üyelik fonksiyonlarını görüş hattı içerinde bulunan kaçan ajan ile olan mesafesine göre "YAKIN", "ORTA", "UZAK" dilsel ifadeler olarak tanımlanmıştır. Bulanık sistemdeki ikinci giriş olan Δd_p'nin üyelik fonksiyonları ise "NEGATİF", "SIFIR" ve "POZİTİF" olarak karar verilmiştir. Bu iki girişe ait üyelik fonksiyonları Şekil 3.18'de gösterilmiştir.



Şekil 3.18: Giriş değerlerinin üyelik fonksiyonları.

İki adet giriş ve bu girişlerin ait olduğu üç adet üyelik fonksiyonlar ile bulanık sistemde dokuz adet kural yazılmıştır. Şekil 3.19'da kaçan ajanın, pembe ajanın görüş hattında farklı koordinatlarda bulunması durumuna göre dokuz adet kural oluşturulmuştur.



Şekil 3.19: Kaçan ajanın görüş hattında bulunduğu durumlar.

Pembe ajanın çizgisel hızını kontrol etmek için bulanık kural tabanında oluşturulan tüm kurallar tablo 3.1'de sunulmuştur.

Tablo 3.1: V_p hızı için tanımlanan 3x3 kurallar.

$d_p/\Delta d_p$	NEGATİF	SIFIR	POZİTİF
YAKIN	ÇK	ÇK	K
ORTA	N	N	B
UZAK	N	B	ÇB

Tablo 3.1'de gösterilen 9 adet kurall için çeşitli kişiler tarafından yapılan deneyler sonucunda 5 adet dilsel sonuç elde edilmiştir. Bunlar ÇK: Çok Küçük, K: Küçük, N: Normal, B: Büyük ve ÇB: Çok Büyük dilsel ifadelerle tanımlanmıştır. ÇK değeri 1 N, K değeri 4 N, N değeri 8 N, B değeri 12 N ve ÇB ise 16 N değerine sahiptir.

R_1: *Eğer* d_p *YAKIN ve* Δd_p *NEGATİF ise, O halde F ÇK'dir.*

Yazılan birinci kuralda kaçan ajan ile pembe ajan arasındaki mesafe YAKIN ve mesafenin birim zamanda değişimi NEGATİF ise, çıkışta ÇK dilsel olarak tanımlanan bir kuvvet değeri elde edilir. Bu kural gereği d_p 0 ile 30 metre arasında ve Δd_p NEGATİF yani kaçan ajan pembe ajana yaklaşıyor ise bulanık sistem çıkışında

$1\,N$'luk kuvvet değeri elde edilir. Bu kuvvet $1\,kg$ kütleye sahip pembe ajana uygulamaktadır. Pembe ajana giriş olarak uygulanan bu kuvvet, çıkışta kaçan ajanı kovalaması gereken hız değerine dönüşür. Ortama yer çekimi kuvveti ve 0.08 katsayılı sürtünme kuvveti eklenmiştir. Yapılan deneyler sonucunda yer çekimi ivmesi ve sürtünme kuvveti oyun ortamından kaldırıldığında pembe ajanın hızı giderek artmakta ve kaçan ajanı yakalayamamaktadır.

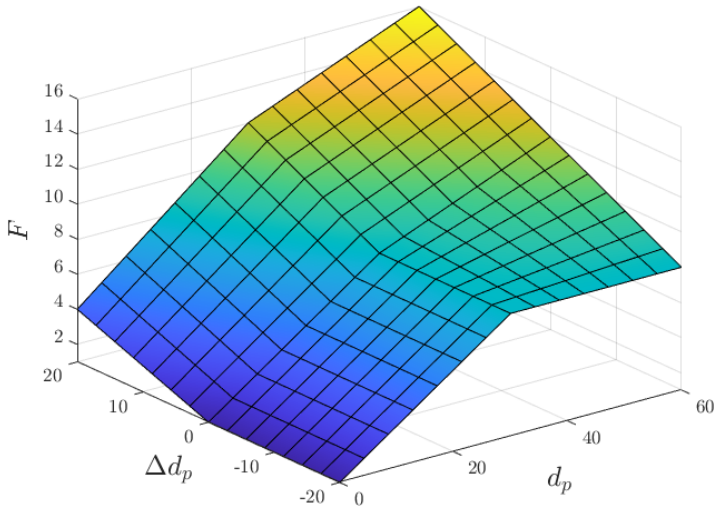
R_2: *Eğer* d_p *ORTA ve* Δd_p *NEGATİF ise, O halde F N'dir*

Bu kuralda ise d_P mesafesi ORTA uzaklıkta ve Δd_p NEGATİF ise çıkışta 8N'luk kuvvet elde edilir. Kaçan ajan ile pembe ajanı arasındaki mesafe 0 ile $60\,m$ arasında ve kaçan ajan, pembe ajana doğru hareket ediyorsa çıkışta $8\,N$'luk kuvvet elde edilir. R_1 kuralında bahsedildiği gibi fizik motoru bu kuvveti pembe ajana uygulayarak çizgisel hıza dönüştürür. Bu kuvvet etkisi ile oluşan çizgisel hız değerine göre pembe ajan kaçan ajanı yakalamaya çalışır.

R_3: *Eğer* d_p *UZAK ve* Δd_p *POZİTİF ise, O halde F ÇB'dir*

Kaçan ajan ile pembe ajan arasındaki mesafe 30 ile $60\,m$ arasında ve mesafenin birim zamanda değişimi POZİTİF ise, diğer bir deyişle kaçan ajan pembe ajandan uzaklaşıyor ise çıkışta $16\,N$ kuvvet oluşur, bu kuvvet pembe ajana fizik motoru ile uygulanarak gerekli çizgisel hız değeri kazandırılmıştır.

Yukarıda çizgisel hız için tasarlanan bulanık sistemin dokuz adet kuralından üç tanesi açıklanmaya çalışılmıştır. Diğer kurallar yapılan deneyler sonucu oluşturulmuştur. Tasarlanan bulanık sistemden elde ettiğimiz yüzey ise Şekil 3.20'de gösterilmiştir.



Şekil 3.20: V_p hızı için elde edilen yüzey.

d_b mesafesi için bulanık sistem tasarımı

Tasarlanan ikinci bulanık sistem ile d_b mesafesi kontrol edilmiştir. Bu sistemin girişleri bir önceki sistemde olduğu gibi d_p ve Δd_p'dir. Her bir girişin üç adet üyelik fonksiyonları olup Şekil 3.18'de gösterilmiştir. İki adet giriş ve sahip oldukları üyelik fonksiyonları ile 9 adet kural denklem 3.12'ye göre oluşturulmuştur. Sistem çıkışında elde edilen gerçel sayı değeri ise d_b mesafesine eşit olup metre cinsindedir.

Çeşitli sayıda kişilerle yapılan deneyler sonucunda bu sistem için çıkışta 5 adet dilsel sonuç elde edilmiştir. Bunlar ÇY: Çok Yakın, Y: Yakın, O: Orta, U: Uzak ve ÇU: Çok Uzak olarak ifade edilmiştir. ÇY ifadesi $0\ m$, Y ifadesi $10\ m$, diğer dilsel ifadeler ise sırasıyla $15, 20, 25\ m$ değerine eşit olmaktadır. Bulanık sistem için oluşturulmuş kuralların bütünü tablo 3.2'de sunulmuştur.

Tablo 3.2: d_b mesafesi için tanımlanan 3x3 kurallar.

$d_p/\Delta d_p$	NEGATİF	SIFIR	POZİTİF
YAKIN	ÇY	ÇY	Y
ORTA	O	O	U
UZAK	O	U	ÇU

R_1: *Eğer* d_p *UZAK ve* Δd_p *POZİTİF ise, O halde* d_b *ÇU'dir.*

Yazılan kuralda d_p mesafe değeri $30\ m$ ile $60\ m$ arasında ve d_p değişimi pozitif ise, d_b $25\ m$ dir. Kaçan ajan pembe ajanın görüş hattına girdiğinde, pembe ajan doğrudan bulanık küreyi yakalamaya çalışacaktır. Bulanık küreyi yakaladığında kaçan ajanı da yakalamış olacaktır. Mesafenin bu kadar büyük seçilmesinin nedeni, yapılan deneyler sonucunda pembe ajanın kaçan ajanı görüş hattından çıkmadan yakalamasına imkân sağlamasıdır. Aralarındaki mesafe büyük olduğu için pembe ajan bu mesafeye göre hız değerini artıracaktır.

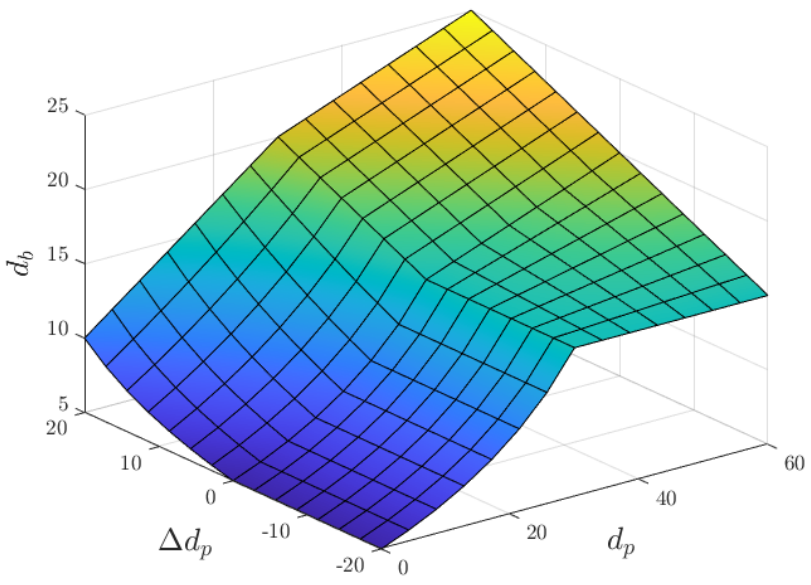
R_2: *Eğer* d_p *UZAK ve* Δd_p *NEGATİF ise, O halde* d_b *O'dir.*

Bu kuralda ise kaçan ajan pembe ajanın görüş hattında uzak mesafede ve kaçan ajan pembe ajana doğru hareket ediyor ise, yapılan deneyler sonucunda d_b $15\ m$ olarak belirlendi. Bu mesafe değerine gitmesi gereken çizgisel hız için fizik motoru pembe ajana $8\ N$'luk kuvvet uygulamaktadır. Bu kuvvet değerinden daha düşük ya da yüksek kuvvet değeri pembe ajana uygulandığında kaçan ajanı yakalayamamaktadır.

R_3: *Eğer* d_p *YAKIN ve* Δd_p *NEGATİF ise, O halde* d_b *ÇY'dir.*

Pembe ajan ile kaçan ajan arasındaki mesafe $0\,m$ ile $30\,m$ arasında ve zamanla d_p mesafesi azalıyor ise d_b $0m$ olarak seçildi. Bu kural gereği bulanık küre kaçan ajanın içinde yer alır. Pembe ajanın, kaçan ajanı yakalaması için hız değeri küçük olması gerekir. Bu nedenle pembe ajana uygulanması gereken kuvvet değerinin $1\,N$ olarak belirlendi. Uygulanan kuvvet değeri yüksek bir değer olursa yakın mesafede pembe ajan kaçan ajanı yakalayamazdı.

Yukarıda d_b için tasarlanan bulanık sistemin dokuz adet kuralından üç tanesi açıklanmaya çalışılmıştır. Diğer kurallar yapılan deneyler sonucu oluşturulmuştur. Tasarlanan bulanık sistemden elde ettiğimiz yüzey ise Şekil 3.21'de gösterilmiştir.



Şekil 3.21: d_b mesafesi için elde edilen yüzey.

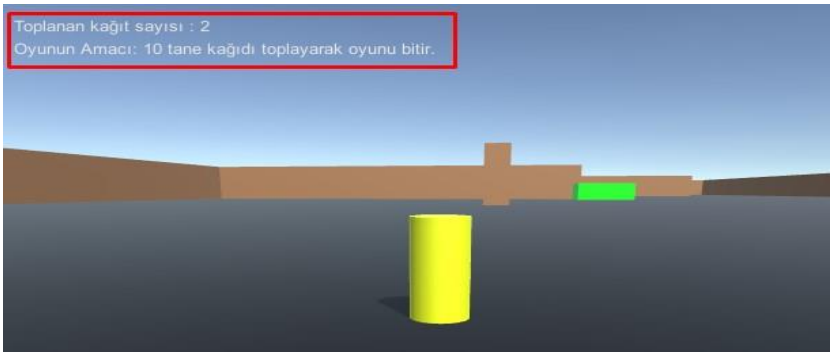
4. UNİTY 3D OYUN ORTAMINDA YAPILAN ÇALIŞMALAR

Bu bölümde, oyun motorunda oluşturduğumuz kaçma-kovalama oyun tasarımında alan seçimi ve tasarımı, oyunun amacı, kaçan ajan tarafından toplanan kâğıtların kovalayan ajanlara etkisi, kovalayan ajanların otonom hareket ederken ve kaçan ajanı takip sırasındaki davranışları incelenip karşılaştırılacaktır.

4.1 Oyun Alanının Oluşturulması ve Amacı

Kaçma-kovalama oyununda, alan tasarımı için yapılan literatür araştırması sonucunda çeşitli şekillerde (üçgen, kare, dikdörtgen) oyun alanı tercih edildi. Bilgisayar oyunun ilk aşamalarında kare oyun alanı oluşturulmuş olup, oyunda bir adet kaçan ve bir adet kovalayan ajan bulunmaktaydı. Geliştirilen stratejiler doğrultusunda kovalayan ajan sayısının artırılması ve oyun alanına engellerin yerleştirilmesi ile boyutu değiştirilerek dikdörtgen boyuta geçilmiştir.

Oyundaki amaç insan kullanıcı tarafından kontrol edilen kaçan ajanın oyun alanında bulunan 10 adet kâğıtları toplamasıdır. Oyun alanında bulunan kâğıtların yapılarında çarpışma elemanı bulunmaktadır. Kaçan ajan kâğıtlar ile temas ettiğinde kâğıtlar oyun alanından silinmektedir. Böylece silinen her kâğıt kaçan ajan tarafından toplanmış kabul edilir. Oyun alanında 10 adet kâğıt toplandığında oyun sona ermektedir. Toplanan kâğıtların sayılarını kullanıcı tarafından takip edilmesi için "User-Interfaces" (UI) oyun objesi kullanılmıştır. Bu oyun objesinin oyun içinde kullanımı şekil 4.1'de gösterilmiştir.



Şekil 4.1: Unity 3D'de kullanıcı arayüzü.

Şekil 4.1'de üzerinde kırmızı dikdörtgen içerisinde gösterilen "User-Interfaces" (UI) oyun objesi sayesinde kullanıcı oyunda topladığı kâğıt sayısını takip edebilmekte ve oyunun amacı hakkında bilgilendirilmektedir. Kâğıt sayısı azaldığında ise kovalayan ajanların çizgisel, açısal hızı ve görüş hattı yapısındaki açı değeri artmaktadır. Kırmızı ajanın çizgisel hızı oyun başladığında $V_k = 60\, m/s$'dir. Kâğıt sayısı azaldığında kırmızı ajanın çizgisel hızı denklem 4.1'de gösterildiği şekilde artış göstermektedir.

$$V_k = V_k + 5b \tag{4.1}$$

Denklemde gösterilen b ise toplanan kâğıt sayısını göstermektedir. Böylece oyunda toplanan kâğıt sayısıyla doğru orantılı olarak kırmızı ajanın çizgisel hızı artmaktadır. Oyun alanında kâğıt sayısı azaldığında ise mavi ajanın açısal hızı artmaktadır. Başlangıçta mavi ajanın açısal hızı $W_m = 90\ °/s$ 'dir. Kâğıt sayısı azaldığında mavi ajanın açısal hızı denklem 4.2'de gösterildiği gibi artmaktadır.

$$W_m = W_m + 10b \tag{4.2}$$

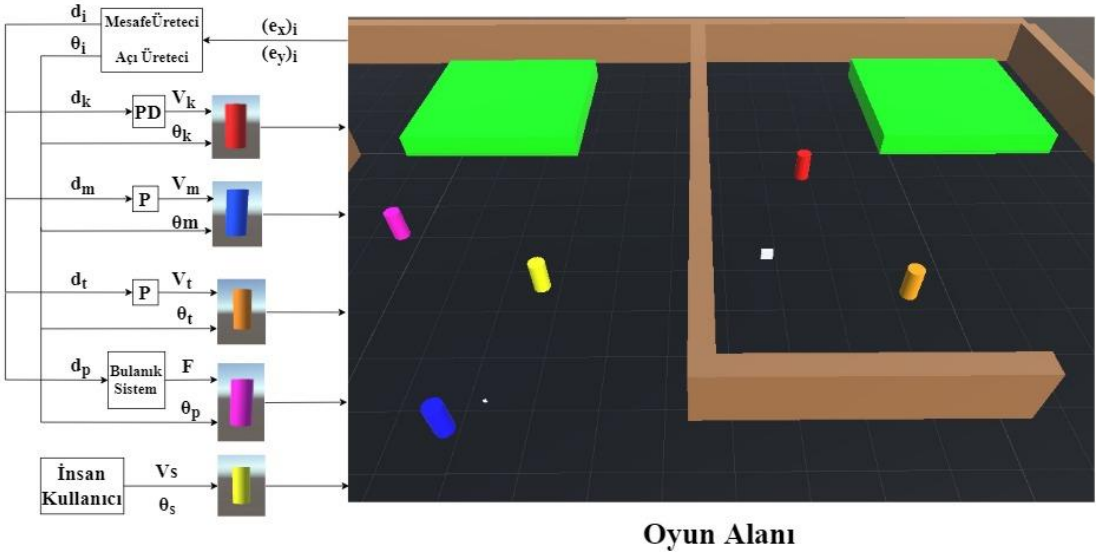
Turuncu ajanın sahip olduğu görüş hattı açısı $\theta_g = 30\,°$ 'dir. Kaçan ajan oyun alanında kâğıtları toplamaya başladıkça turuncu ajanın sahip olduğu görüş hattı açı değeri denklem 4.3'te gösterildiği gibi artmaktadır.

$$\theta_g = \theta_g + 10b \tag{4.3}$$

Böylece kâğıt sayısı azaldıkça kırmızı ajanın çizgisel hızı, mavi ajanın açısal hızı ve turuncu ajanın görüş hattı açı değeri artış göstermekte ve ajanların davranışları agresifleşmektedir.

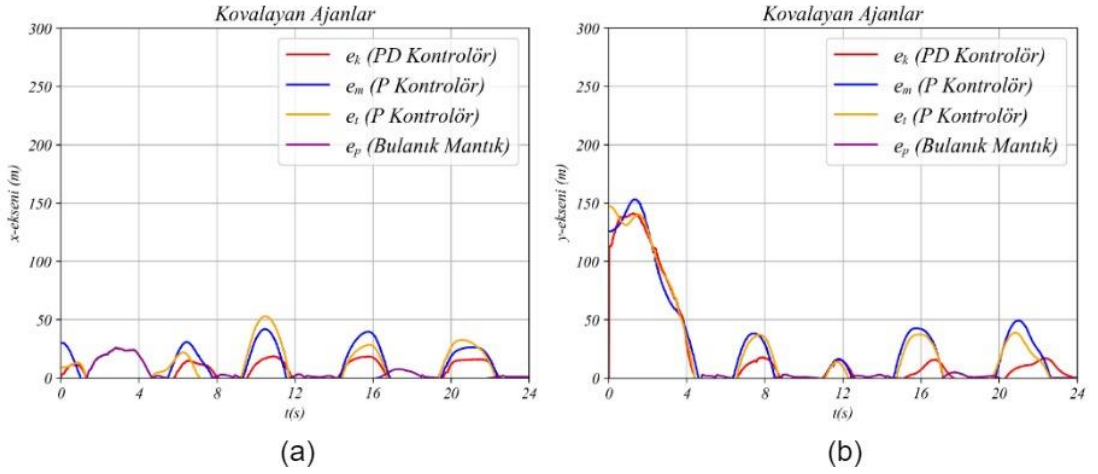
4.2 Kırmızı, Mavi, Turuncu ve Pembe Ajanların Kovalama Davranışları

Şekil 4.2'de gösterilen oyun alanında bulunan kırmızı ajanın yapısında PD kontrolör, mavi ve turuncu ajanın yapısında P kontrolör ve pembe ajanın yapısında bulanık sistem bulunmaktadır. Bu dört ajan oyun alanında otonom hareket ederken kendileri için tanımlanmış çizgisel hızlar ile hareket etmektedir. Eğer kaçan ajan kovalayan ajanların görüş hattına girerse, kovalayan ajanlar kaçan ajanı yapılarında bulunan kontrolör ile yakalamaya çalışmaktadır.



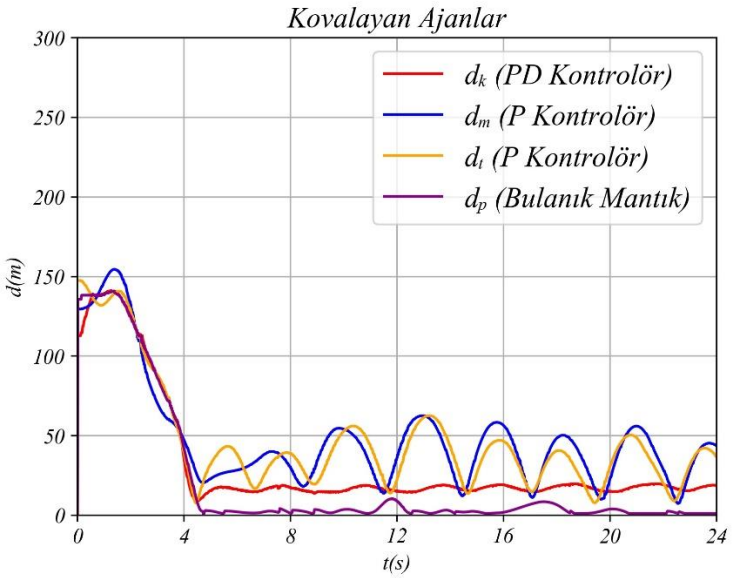
Şekil 4.2: Kaçma-kovalama oyun alanı.

Yapılan kovalama deneyinde kaçan ajan insan kullanıcı tarafından kontrol edilip hareket halindedir. Şekil 4.3'te kovalama deneyi sırasında kaçan ajan ile kovalayan ajanlar arasındaki mesafesinin x ve y eksenlerindeki değişimleri gösterilmiştir.



Şekil 4.3: Kaçan ajan ile kovalayan ajan arasındaki mesafelerin (a) x ekseninde, (b) y eksenindeki değişimleri.

Şekil 4.3.a'da ajanların x ekseni üzerindeki mesafe değişimleri, Şekil 4.3.b'de ise y eksenindeki mesafe değişimleri gösterilmiştir. Bu değişimler sırasıyla e_x ve e_y değerleridir. Bu değerler Şekil 3.2'de gösterilen mesafe üretecine giriş olarak verildiğinde kaçan ajan ile her bir kovalayan ajan arasındaki en kısa mesafe olan d_i bulunur. Şekil 4.4'te ise kaçan ajan ile kovalayan ajanlar arasındaki d_k, d_m, d_t ve d_p mesafe değişimleri gösterilmiştir.

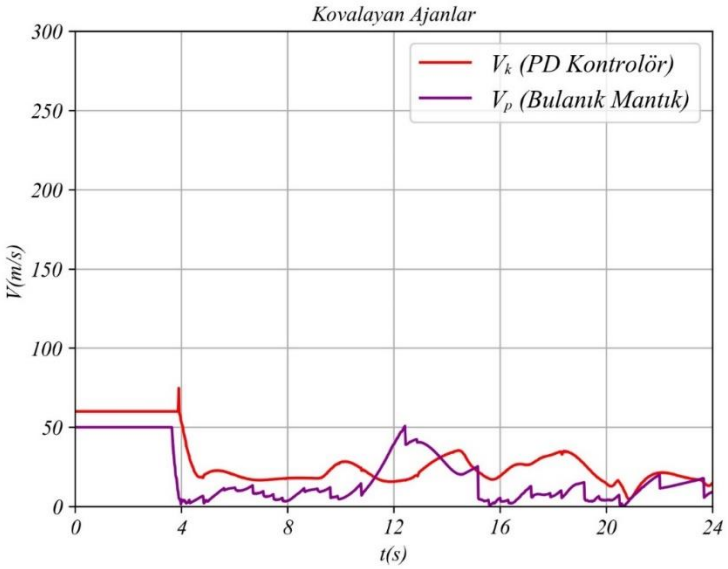


Şekil 4.4: d_k, d_m, d_t ve d_p mesafeleri.

Oyunda yakalama tanımı yüzey temasıdır. Diğer bir deyişle $d_i = 0m$ olmasıdır. Kovalama deneyinde kaçan ajan hareket halinde olup kovalayan ajanlarla arasındaki mesafe zamanla azalıp kovalayan ajanların görüş hattına girmektedir. Kırmızı ajan görüş hattı içerisinde kaçan ajanı yakalamak için PD kontrolör ile V_k hızını kontrol etmektedir. Ancak Şekil 4.4'te görüldüğü gibi $d_k = 0m$ olmamakta ve kaçan ajanı yakalayamamaktadır. Mavi ve turuncu ajan, kaçan ajanı görüş hattında değişim gösteren d_m ve d_t mesafelerinde $V_m = 75\ m/s$ ve $V_t = 70\ m/s$ sabit hızlarla takip etmektedir. Bu nedenle d_m ve d_t mesafeleri Şekil 4.3'te gösterildiği gibi salınımlı bir görünüm elde eder. Mavi ve turuncu ajan, kaçan ajanla yüzey teması gerçekleştiremediği için kaçan ajanı yakalayamamaktadır. Ayrıca mavi ve turuncu ajana kırmızı ajanla konum haberleşmesi sağlandığı için $d_k \leq 30m$ olduğu sürece kaçan ajanı sağdan ve soldan kovalamaya devam etmektedir. Konum haberleşmesi ile mavi ve turuncu ajan kaçan ajanı, görüş hattından çıksa bile tekrar takip edip, görüş hattına kaçan ajan girdiğinde yakalamaya çalışacaktır. Pembe ajan ise kaçan ajanı görüş hattına girdiğinde aralarında değişim gösteren d_p mesafesine göre V_p hızını bulanık mantık ile kontrol ederek kaçan ajanı yakalamaya çalışacaktır. Şekil 4.4'te görüldüğü gibi kaçan ajan ile pembe ajan arasında yüzey teması gerçekleşmiş, diğer bir deyişle $d_p = 0\ m$ şartı sağlandığı için kaçan ajan yakalanmıştır.

Kovalayan ajanlar kaçan ajanı görüş hattı içerisinde kovalarken, kırmızı ajan PD kontrolör ile V_k hızını, Mavi ve turuncu ajan P kontrolör ile V_m ve V_t hızını, pembe

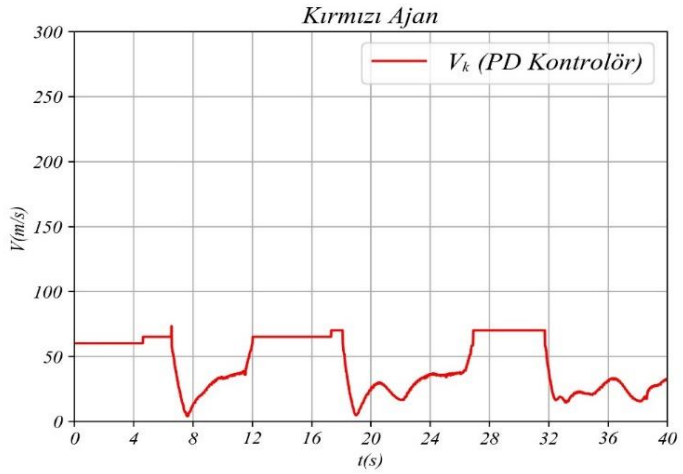
ajan ise bulanık mantık yöntemi ile V_p hızını kontrol etmektedir. Oyun içerisindeki çizgisel hızları Şekil 4.5'te gösterilmiştir.



Şekil 4.5: Kovalayan ajanların çizgisel hızları.

Şekil 4.5'te kırmızı ajan oyun alanı içerisinde otonom hareket ederken $V_k = 60m/s$ hızına sahiptir. Kaçan ajan, kırmızı ajanın görüş hattına girdiğinde PD kontrolör devreye girerek değişim gösteren d_k mesafesine göre V_k hızını kontrol etmektedir. Şekil 4.5'te kırmızı ajan 4. saniyeye kadar oyun ortamında otonom olarak sabit hızla hareket etmektedir. Grafik üzerinde 4. saniyeden sonra kaçan ajan kırmızı ajanın görüş hattına girdiği için çizgisel hızı aralarındaki mesafeye göre değişim göstermektedir. Mavi ve turuncu ajanın çizgisel hızları Şekil 4.5'te gösterilmemiştir. Nedeni ise otonom hareket sırasında ve kaçan ajanı kovalarken sabit $V_m = 70\ m/s$ ve $V_t = 75m/s$ hızla hareket etmesidir. Pembe ajan diğer kovalayan ajanlarda olduğu gibi otonom hareketi sırasında sabit hızla hareket etmektedir. Kaçan ajan pembe ajanın görüş hattına girdiğinde, V_p hızı değişim gösteren d_p mesafesine göre bulanık mantık ile kontrol edilecektir. Şekil 4.5'te 4.saniyeden itibaren kaçan ajan pembe ajanın görüş hattına girmiş olup, bu andan itibaren V_p hızı bulanık mantık ile kontrol edilir.

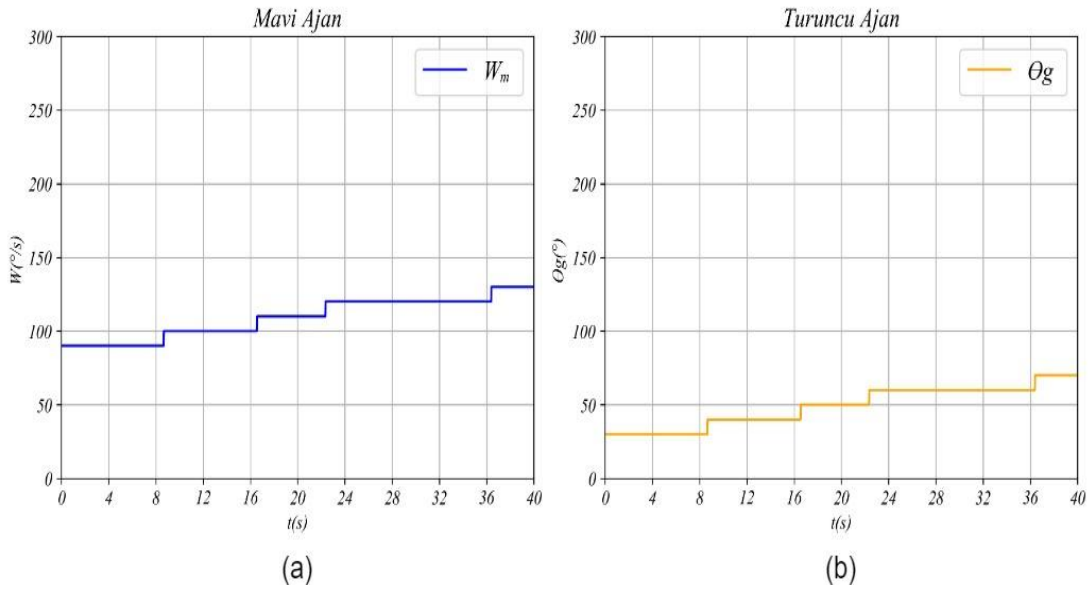
Oyunun amacı doğrultusunda kaçan ajanın oyunu tamamlanması için 10 adet kâğıdı toplaması gerekiyordu. Toplanan her kâğıt kırmızı ajanın çizgisel hızında, mavi ajanın açısal hızında ve turuncu ajanın görüş hattı açı değerinde artış meydana getirmektedir. Kaçan ajan tarafından toplanan kâğıtların kırmızı ajanın çizgisel hızında meydana getirdiği değişiklikler Şekil 4.6'da gösterilmiştir.



Şekil 4.6: Toplanan kâğıtların V_k 'ya etkisi.

Şekil 4.6'da kırmızı ajan oyun alanı içerisinde $V_k = 60\ m/s$ hızla hareket etmektedir. Kaçan ajan her kâğıt topladığında kırmızı ajanın V_k hızı denklem 4.1'de gösterildiği gibi artmaktadır. Kâğıt topladıkça değişim gösteren V_k hızına göre kırmızı ajan oyun alanında otonom hareket etmektedir. Kaçan ajan görüş hattına girdiğinde PD kontrolör yeni V_k hız değerini kontrol ederek kaçan ajanı yakalamaya çalışır. Kaçan ajan görüş hattından çıktığında ise kırmızı ajan yeni V_k hızıyla otonom hareketine devam eder.

Oyunda kâğıt toplandığında mavi ajanın açısal hızındaki artış denklem 4.2'de ve turuncu ajanın görüş hattı açı değeri ise denklem 4.3'te gösterildiği gibi artmaktadır. Toplanan kâğıtların mavi ve turuncu ajana olan etkileri Şekil 4.7'de gösterilmiştir.



Şekil 4.7: Toplanan kâğıtların (a) W_m'ye etkisi (b) θ_g'ye etkisi.

5. SONUÇ VE TARTIŞMA

Bu tez çalışmasında, Unity 3D oyun motorunda kaçma-kovalama tabanlı bilgisayar oyunu tasarlandı. Bilgisayar oyunu tasarımı için kullanılan oyun motorunun yapısı ve sahip olduğu elemanlar hakkında araştırma yapıldı. Oyun motoru içerisinde çeşitli sayıda elemanlar bulunmaktadır. Tasarlanan bilgisayar oyunu için gerekli olan elemanlar tez kapsamında kullanıldı. Bilgisayar oyununda alan tasarımına karar verilmesi için literatür araştırması yapıldı. Kaçma-kovalama oyun tasarımı için araştırmacılar tarafından çeşitli geometrilerde alanlar tercih edilmiştir. Oyunda kovalayan ajan sayısının artırılması, oyun alanına engeller ve ölüm bölgelerinin yerleştirilmesi ile dikdörtgen boyutlarda oyun alanı tasarlanmıştır. Oyunun amacı doğrultusunda kaçan ajan tarafından toplanması gereken kâğıt oyun objeleri yerleştirildi.

Oyun alanı tasarımı ve amacı tamamlandıktan sonra, oyun içerisine farklı renklerde dört adet kovalayan ajan ve bir adet kaçan ajan yerleştirildi. Kaçan ajan insan kullanıcı tarafından kontrol edilmekte olup, kovalayan ajanlar oyun içerisinde otonom hareket etmektedir. Otonom hareket etmesi için gerekli oyun motoru elemanları ve yol nokta yapısı kovalayan ajanlara tanımlandı. Kovalayan ajanlara belirli bir mesafede kaçan ajanı görmelerini sağlamak için görüş hattı yapısı tasarlandı. Tez kapsamında kırmızı, mavi ve turuncu ajana yukarıda bahsedilenler ışığında kovalama stratejisi geliştirildi. Kaçan ajan kovalayan ajanların görüş hattına girdiğinde kovalanacak, çıktığında ise kovalayan ajanlar otonom hareket edecektir. Ayrıca mavi ve turuncu ajanlara kırmızı ajan ile konum haberleşmesi sağlanarak kaçan ajanı sağdan ve soldan takip etmesi sağlandı. Yapılan kovalama deneyleri sonucunda bu üç ajanın kaçan ajanı yakalayamadığı görülmüştür.

Oyunda bulunan pembe ajan için diğer ajanlarda ortaya çıkan problemi çözmek amacıyla "Takagi-Sugeno-Kang" bulanık modeli kullanılarak strateji geliştirilmiştir. Bu model ile iki adet bulanık sistem tasarlanmış olup, pembe ajanın görüş hattı yapısında kullanılmıştır. Tasarlanan birinci bulanık sistem ile kaçan ajan pembe ajanın görüş hattına girdiğinde insanın düşünme ve karar verme özelliğini taklit edilerek

aralarındaki mesafe değişimine göre çizgisel hızı kontrol edilmiştir. Pembe ajanın değişim gösteren mesafede sahip olması gereken çizgisel hız için oyun motorunda bulunan fizik motoru bulanık sistem ile kontrol edildi. Mesafe değişimine göre gerekli kuvvet pembe ajana uygulanarak kovalama için gerekli çizgisel hız elde edildi. Diğer bir deyişle aralarındaki mesafe fazla ise büyük kuvvet uygulanarak yüksek bir hız, aralarındaki mesafe az ise küçük kuvvet uygulanarak kaçan ajanı yakalaması gereken düşük hız sağlandı. İkinci tasarlanan bulanık sistem ise kaçan ajanın pembe ajandan kaçacağı konumu hesaplamıştır, birinci bulanık sistem ile pembe ajanın ilgili konuma gitmesi sağlanmıştır. Böylelikle pembe ajan için tasarlanan iki adet bulanık sistem ile akıllı bir kovalama stratejisi geliştirilmiştir.

Sonuç olarak, kaçma-kovalama temalı bilgisayar oyununda kırmızı, mavi ve turuncu ajanlarda yakalama problemi ortaya çıkmıştır. Bu probleme çözüm getirmek için oyun alanına pembe ajan eklenip, bu ajan için bir bulanık model ile strateji geliştirilmiştir. Yapılan deneyler sonucunda tasarlanan iki adet bulanık sistem ile pembe ajanın kaçan ajanı yakalama konusunda diğer ajanlardan daha başarılı ve performans açısından daha tahmin edici olduğu görülmüştür.

KAYNAKLAR

[1] **Url-1** <https://www.tsf.org.tr/>**,**erişim tarihi, 13.05.2020

[2] **Url-2** < https://en.wikipedia.org/wiki/Pong >,erişim tarihi, 13.05.2020

[3] **Url-3** <https://www.bandainamcoent.com/games/pac-man>**,** erişim tarihi, 13.05.2020

[4] **D. Robles and S. M. Lucas**.(2009)."A simple tree search method for playing Ms. Pac-Man," *2009 IEEE Symposium on Computational Intelligence and Games*, Milano, pp. 249-255, doi: 10.1109/CIG.2009.5286469.

[5] **M. Gallagher and A. Ryan**.(2003) ."Learning to play Pac-Man: an evolutionary, rule-based approach," *The 2003 Congress on Evolutionary Computation, CEC '03.*, Canberra, ACT, Australia, 2003, pp. 2462-2469 Vol.4, doi: 10.1109/CEC.2003.1299397.

[6] **H. Handa and M. Isozaki**.(2008). "Evolutionary fuzzy systems for generating better Ms. PacMan players," *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, Hong Kong, pp. 2182-2185, doi: 10.1109/FUZZY.2008.4630672.

[7] **Paul, Partha & Goon, Surajit & Bhattacharya, Abhishek**.(2012). "History and comparative study of modern game engine". International Journal of Advanced Computer and Mathematical Sciences.3.2230-9624

[8] **Xiao Cui and Hao Shi**.(2011). "Evolutionary fuzzy systems for generating better A*-based Pathfinding in Modern Computer Games, IJCSNS International Journal of Computer Science and Network Security,48 VOL.11 No.12,

[9] **Zikky, M**. (2016). Review of A* (A Star) Navigation Mesh Pathfinding as the Alternative of Artificial Intelligent for Ghosts Agent on the Pacman Game. *EMITTER International Journal of Engineering Technology*, *4*(1), 141-149.

[10] **S. Wang and X. Zhao**.(2011). "The improved Dijkstra's shortest path algorithm," *2011 Seventh International Conference on Natural Computation*, Shanghai, 2011, pp. 2313-2316, doi: 10.1109/ICNC.2011.6022589.

[11] **Syahputra, Mohammad & Arippa, A & Rahmat, Romi & Andayani, Ulfi**. (2019). Historical Theme Game Using Finite State Machine for Actor Behaviour. Journal of Physics: Conference Series. 1235. 012122. 10.1088/1742-6596/1235/1/012122.

[12] **Sekhavat, Yoones**. (2017). Behavior Trees for Computer Games. International Journal on Artificial Intelligence Tools. 26. 10.1142/S0218213017300010.

[13] **Orkin, J.** (2006). Three States and a Plan: The A.I. of F.E.A.R.

[14] **Orkin, Jeff.** (2004). "Symbolic Representation of Game World State: Toward Real-Time Planning in Games." .

[15] **D. A. Suyikno and A. Setiawan**(2019). "Feasible NPC Hiding Behaviour using Goal Oriented Action Planning in case of Hide-and-Seek 3D Game Simulation," 2019 Fourth International Conference on Informatics and Computing (ICIC), Semarang, Indonesia, pp. 1-6, doi: 10.1109/ICIC47613.2019.8985962.

[16] **Chung, T. H., Hollinger, G. A., Isler, V.** (2011). Search and pursuit-evasion in mobile robotics, *Autonomous robots*, ***31***, 299.

[17] **Desouky, S. F., Schwartz, H. M.** (2011). Q (λ) learning adaptive fuzzy logic controllers for pursuit–evasion differential games, *International Journal of Adaptive Control and Signal Processing*, ***25***, 910–927.

[18] **Bopardikar, S. D., Bullo, F., and Hespanha, J. P.** (2009). A cooperative homicidal chauffeur game, *Automatica*, ***45***, 1771–1777.

[19] **Camci, E. , and Kayacan, E.** (2016). Game of drones: UAV pursuit-evasion game with type-2 fuzzy logic controllers tuned by reinforcement learning, *IEEE International Conference on Fuzzy Systems*, 618–625.

[20] **Kachroo, P., Shedied, S. A., Bay, J. S., and Vanlandingham, H.** (2001). Dynamic programming solution for a class of pursuit evasion problems: the herding problem, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, ***31***, 35–41.

[21] **Bhattacharya, S., and Hutchinson, S.** (2009). On the existence of nash equilibrium for a two player pursuit-evasion game with visibility constraints. In Algorithmic Foundation of Robotics VIII. Springer, Berlin, Heidelberg, 251–265.

[22] **Choi, Han-Lim & Tahk, Min-Jea & Bang, Hyochoong**. (2006). Neural network guidance based on pursuit-evasion games with enhanced performance. Control Engineering Practice. 14. 735-742. 10.1016/j.conengprac.2005.03.001.

[23] **Cui, Xiao & Shi, Hao.** (2010). A*-based Pathfinding in Modern Computer Games. 11.

[24] **Cui, S.-G & Wang, H. & Yang, L.** (2012). A simulation study of A-star algorithm for robot path planning. 506-509.

[25] **Rodríguez-Puente, R., Lazo-Cortés**. (2013)M.S. Algorithm for shortest path search in Geographic Information Systems by using reduced graphs. *SpringerPlus* **2,** 291

[26] **H. Kolivand and M. S. Sunar**. (2013). "A survey of shadow volume algorithms in computer graphics," *IETE Technical Review*, vol. 30, no. 1, pp. 38–46,

[27] **N. R. Sturtevant and R. Geisberger**. (2010)."A comparison of high-level approaches for speeding up pathfinding," in *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE '10)*, pp. 76–82,

[28] **J. van den Berg, R. Shah, A. Huang, and K. Goldberg**. (2011).“ANA∗: anytime nonparametric A,” in Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI ’11), pp. 105–111,.

[29] **Abd Algfoor, Zeyad & Sunar, Mohd Shahrizal & Kolivand, Hoshang**. (2015). A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games. International Journal of Computer Games Technology. 2015. 1-11. 10.1155/2015/736138.

[30] **Zadeh, L. A.** (1965). Fuzzy sets, *Information and* Control, ***8***, 338–353.

[31] **Mizumoto, M.** (1995). Realization of PID controls by fuzzy control methods, *Fuzzy sets and* systems., ***70***, 171–182.

[32] **Guo, Y., and Woo, P. Y.** (2003). An adaptive fuzzy sliding mode controller for robotic manipulators, *IEEE Transactions on Systems, Man, and Cybernertics-Part A: Systems and Humans*, ***33***, 149–159.

[33] **Beke, A. , Kumbasar, T.** (2017). Game of spheros: A real-world pursuit-evasion game with type-2 fuzzy logic, *IEEE International Conference on Fuzzy Systems*.

[34] **Bloch, I.** (2015). Fuzzy sets for image processing and understanding, *Fuzzy Sets and Systems*, ***281***, 280–291.

[35] **Bellman, R. E. and Zadeh, L. A.** (1970). Decision-making in a fuzzy environment, *Management Science*, ***17***, B–141–B–164.

[36] **Pirovano, M.** (2012). The use of Fuzzy Logic for Artificial Intelligence in Games.

[37] **Li, Y., Musílek, P., & Wyard-Scott, L**. (2004). Fuzzy logic in agent-based game design. *IEEE Annual Meeting of the Fuzzy Information, 2004. Processing NAFIPS '04., 2*, 734-739 Vol.2.

[38] **Lara, C.A., Mitre-Hernández, H.A., Flores, J.J., & Perez, H**. (2018). Induction of Emotional States in Educational Video Games through a Fuzzy Control System. *IEEE Transactions on Affective Computing*, 1-1.

[39] **Vorachart, V. & Takagi, H.** (2017). Evolving fuzzy logic rule-based game player model for game development. International Journal of Innovative Computing, Information and Control. 13. 1941-1951.

[40] **Etlik, U. B., Korkmaz, B., Beke, A., & Kumbasar, T.** (2019). A fuzzy logic-based autonomous car control system for the JavaScript Racer game. *Transactions of the Institute of Measurement and Control*.

[41] **Chiu, S. L.** (1994). Fuzzy model identification based on cluster estimation, *Journal of Intelligent & fuzzy systems*, ***2***, 267–278.

[42] **Zadeh, L. A.** (1996). Fuzzy logic= computing with words, *IEEE Transactions on Fuzzy Systems*, ***4***, 103–111.

[43] **Mamdani, E. H.** (1974). Application of fuzzy algorithms for control of simple dynamic plant. *Proceedings of the Institution of Electrical Engineers*, 1585–1588.

[44] **Mamdani, E. H., and Assilian, S.** (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies*, ***7***, 1–13.

[45] **Takagi, T., and Sugeno, M.** (1985). Fuzzy identification of systems and its applications to modelling and control, *IEEE Transactions on Systems, Man, and Cybernertics*, ***15***, 116-132.

ÖZGEÇMİŞ



Ad-Soyad **:** İhsan Şahin
Doğum Tarihi ve Yeri **:** Çorum, 28.12.1991
E-posta **:** sahini17@itu.edu.tr

ÖĞRENİM DURUMU:

- **Lisans** **:** 2015, Erciyes Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2020-…. : Araştırma Görevlisi, Nişantaşı Üniversitesi, Mekatronik Mühendisliği