Project 11

CengBall: Yapay Zekâ Futbol Algoritmaları Oyunu

Proje Sonuç Raporu

Project 11
14.05.2014

Project 11

İçindekiler

Şekiller Listesi

Özet

Programlama oyunu, oyuncuların oyuna doğrudan bir etki gösteremediği, bunun yerine kendi yazdıkları bilgisayar programı veya programcığı aracılığıyla oyun içi karakterlerin hareketlerini kontrol edebildiği bir bilgisayar oyunu çeşididir. CengBall bir programlama oyunu projesidir. CengBall projesi, yazılımla veya yapay zekâ tasarımıyla ilgilenen insanlara, kendi yapay zekâ kodlarını yazma ve başkalarıyla yarıştırma imkânı vermektedir. Proje, kuralları basitleştirilmiş bir futbol oyunudur. CengBall programlama oyunu, gerçek hayat spor stratejisini, yazılım mühendisliğini ve yapay zekâ algoritma tasarımını bir araya getirmektedir. Kullanıcıların yapay zekâ algoritmaları tasarlayıp bu algoritmalar için yazılım geliştirebileceği ve diğer kullanıcılarla yarışabileceği bir platform olmasının yanında; futbol oyununun bileşenlerini nesnel tabanlı bir şekilde dinamik bir altyapıda kontrol edilebilir ve değiştirilebilir tutarak yapay zekâ geliştirmeye olanak sağlamaktadır.

Proje, geniş şablondan bakmak gerekirse, eşit sayıda oyuncusu bulunan iki futbol takımının yazılmış kodlar aracılığıyla yarıştırılması temeline dayanmaktadır. Takımları kontrol eden kullanıcılar, oyuncular bazında pozisyon değiştirme, şut çekme, pas atma, araya kaçma gibi tercihleri yaparak, her an dinamik değişen top durumuna göre taktik geliştireceklerdir. Simülatör tarafından kullanıcılara düzenli veri akışı sağlanacaktır. Bu veri akışı, kullanıcının takım ajanını öğrenebilir kılmasını sağlayacak dinamik etmenleri de barındırmaktadır. Örnek vermek gerekirse, ajan karşı takımdan şut çeken oyuncuların yüzdesini çıkartabilecek ve dinamik bir şekilde adam adama baskı gibi futbol taktikleriyle kontrol sağlayabilecektir.

Oyun kapsamında; futbol oyununun 2 boyuta indirgenmiş ve ofsayt, köşe vuruşu gibi kuralları çıkartılmış bir halinin benzetimi yapılmaktadır. Fiziksel değişkenler gerçeklikle örtüşür şekilde, bilimsel normlarla hesaplanmaktadır.

Kullanıcı kodunun sisteme dâhil edilmesinin tamamen güvenli hale getirilmesi için sistemin güvenlik protokollerinin tasarımı proje süresince geliştirilmektedir. Ayrıca projede; gerçekleme, yapay zekâ, görselleştirme parçaları ve sınıf tanımları arasındaki veri alışverişi güvenliği, gerçekleme katmanında veri yedekleme mantığıyla sağlanmaktadır. Bu sayede onay alamayacağı çağrılarda bulunan yapay zekâ ajanları uyarılabilecek ve asla gerçeklenen oyunu bozamayacaktır.

Kullanıcı, oyun ara yüzünden sisteme dâhil ettiği kodunu, oyun başı ayarlarını yaptıktan ve sistemde yer alan yerel rakiplerden birini seçtikten sonra gerçekleyebilecektir. Simülasyon sonrası oluşturulan disk dosyası, gerçeklenmiş maçın 2 boyutlu grafik ar yüzleriyle, gerçek zamanlı olarak sergilenmesini sağlayacaktır.

Anahtar kelimeler: yapay zekâ, programlama oyunu, spor algoritmaları, algoritma tasarımı, programlama yarışması

1. Giriş

Bu doküman CengBall: Yapay Zeka Futbol Algoritmaları oyunu projesinin sonuç raporudur. Rapor dahilinde ilişkili çalışmalar incelenecek, tasarım, gerçekleştirim ve konfigürasyon hakkında bilgiler verilecektir. Proje takvimi sunulacak ve proje sonuçları, özgünlük kriterleri bazında değerlendirilecektir.

2. İlişkili Çalışmalar

Programlama oyunu kategorisinde çeşitli oyunlar yer almaktadır. Bu oyunlar genel olarak robotik savaş simülasyonları ve yapboz oyunlarıdır. Spor alanında, yapay zekâ geliştirilerek oynanabilen tek oyun çeşidi ise TORCS isimli araba yarışıdır[1]. Bu araba yarışında kullanıcı arayüzü üzerinden yarış ve araba seçimi yapılmasına karşın, 3 boyutlu yarış anı yalnızca bir görselleştirmeden ibarettir. Oyun, kullanıcının C++ programlama dilinde yazılmış, drive fonksiyonu dâhilinde ilerlemektedir.

Benzer nitelikte şu ana dek yapılan çalışmalar içerisinde en dikkat çekici olan ise uluslararası düzeyde her sene düzenlenen RoboCup Futbol Simülasyon Ligi'dir[2]. Yarışma, fiziksel bir robot olmaksızın, 2 boyutlu gerçekleme ile bilgisayar programlarının futbol oyunu temelinde yarıştırılmasına olanak sağlamaktadır.

CengBall oyun projesinin benzer programlama oyunlarından ayıran özelliklerden 9. Projenin Özgünlüğü bölümünde bahsedilecektir.

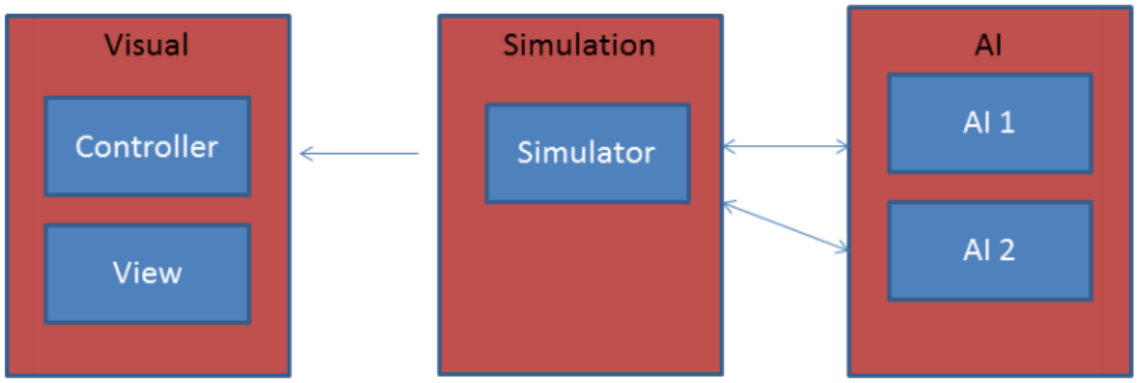
3. Tasarım

Sistem ve veri tasarımıyla ilgili detaylı sınıf diyagramları Yazılım Tasarım Raporu'nda sunulmaktadır.

a. Sistem Tasarımı

Sistem 3 alt sistemden oluşmaktadır. Bunlar oyuncuların sistemle etkileştiği yapay zekâ kodları, bu kodların çalıştırıldığı ve yarıştırıldığı gerçekleme kısmı ve sonuçların görselleştirildiği görselleştirme kısmıdır.

Bileşen şeması şekil 1'de görülmektedir.



Şekil 1: Sistem tasarımı

Sınıf diyagramı ana hatlarıyla Yazılım Tasarım raporunda sunulmuştur.

i. Simülastör modülü

Bu parça, periyodik olarak aldığı kullanıcı girdisini, yapay futbol kurallarına uygunluğunu kontrol ederek, gerçekleyecektir. Yapay zekâ kodları arasındaki iletişimi sağlayacaktır. Simülasyon sonlandığı zaman sonuçları kaydedecektir. Sonuçlar daha sonra görselleştirme parçası tarafından görselleştirilecektir. Simülatör parçası içinde barınan Takım sınıfı objelerinden, kendi metotları aracılığıyla aldığı verileri gerçekleyecektir. Takım sınıfından türettiği sınıfı kodlayacak olan kullanıcı, takım oyuncularının her biri için niyet ayarlaması yapacaktır. Örnek vermek gerekirse kullanıcı şut çekmesini istediği oyuncu üzerinden ilgili metodu, kuvvet ve doğrultu verileriyle çağıracak, bu işlemde ilgili oyuncuda bir niyet ayarlaması yapacaktır. Simülatör parçası, takım verilerindeki tutarlılık ve bu verilerin oyunun fiziksel özellikleriyle uyumluluğunu kontrol edecek, bir sonraki sıra için, saha içi nesnelerinin dinamik özelliklerini güncelleyecektir.

Takım sınıfı, kullanıcıların kapsamını belirleyeceği ana parça olma özelliği göstermektedir. İçerisinde her bir zaman diliminde gerçekleme parçası tarafından güncellenen Oyuncu sınıfından iki ayrı nesne listesi yer almaktadır. Bu nesneler, yapay zekâ ortamının tamamen gözlemlenebilir olması amacıyla oyunculara ait pozisyon ve hız bilgilerini barındırmaktadır. Fakat kullanıcı; rakip Takım sınıfına ait Oyuncu nesnelerinin özelliklerini ancak bu özellik sergilendiğinde görebileceklerdir. Örnek vermek ve somutlaştırmak gerekirse, şut çekmemiş bir oyuncunun şut değeri bilinemeyecektir.

Oyuncu sınıfından oluşturulacak nesneler, nesnel tabana dökülmüş bir futbol oyununda takımların birer alt bileşenleridirler. Bu sistematik yapı korunmaktadır. Oyuncu sınıfı nesneleri, bazı değişmez özelliklerin yanı sıra yorulma gibi zaman dilimleriyle doğru orantılı bir şekilde, oyunculara özgü katsayı kadar hızla düşüşe geçen değişken özellikler de barındırmaktadır. Bir Oyuncu nesnesi, şut, pas, top kapma, hareket etme gibi hareketlere olan niyetini, gerçekleme parçasına bildirmek için metotlar içermektedir.

Tekilliği saklanarak gerçekleme parçası içinde korunan Top sınıfı nesnesi, Takım sınıfının da birer bileşeni olma özelliği taşır. Bu sayede yapay zekâ ajanı, kontrol etmenin oldukça önemli olduğu Top nesnesinin pozisyon ve hız bilgilerine istediği zaman ulaşabilir.

Nesnelerinin özellikleri ve matematiksel fonksiyon hesaplamalarında kullanılacak olan katsayıların tanımlarını içeren Saha sınıfı, bazı özelliklerini oyun başı ayarlarından alacaktır.

ii. Yapay zeka modülü

Bu alt sistem, kullanıcının sistemle etkileşime geçtiği parçadır. Bu kodlar sistemden düzenli olarak veri alacak ve bunlara dayalı kararlar vererek, girdileri sisteme iletecektir.

Kullanıcılar kendilerine sağlanan API aracılığıyla kodlarını geliştirebileceklerdir. Bu API kullanıcıya; takımı yönetme ve çevre hakkında bilgi alma imkânı sağlayacaktır. Kullanıcının daha rahat bir şekilde geliştirme sağlayabilmesi için API içerisinde yardımcı yapay zekâ metotları yer alacaktır.

Kullanıcı, Takım ara yüzünü geliştirerek, kendi stratejisini oluşturabilecektir. Bu süreçte dinamik olarak değişen veriler, oyuncu pozisyonları, top pozisyonu ve oyuncu yetenekleri ve değişken durumlarıdır.

Bu veriler, karar verme sırası gelen oyuncuya gerçekleme parçası tarafından sağlanacaktır.

Ayrıca oyun başında belirlenecek bazı değişmez özellikler bu parçada kullanıcılar tarafından değiştirilemeyecek şekilde ayarlanmaktadır.

```
import java.awt.Color;
import java.awt.geom.Rectangle2D;
import java.util.ArrayList;
import java.util.Arrays;

import rare.Pitch;
import rare.PlayerAction;
import rare.vector2D;
import utils.CengBallException;
import utils.Enums.ActionType;
import common.MatchAssets;
import common.MovingObject;
import common.Player;
import common.Team;

public class Agent2 extends Team {
	private static String teamName = "Besiktas";
	private static Color teamColor1 = new Color(0, 0, 0), teamColor2 = new Color(255, 255, 255);
	private static String[] names = {"Sergen", "Metin", "Ahmet", "Feyyaz", "Pascal"};
	private Rectangle2D.Double myGoalPost, opponentGoalPost;
	private Player[] myPlayers, opponentPlayers;
	private int[] defenderIDs, midfielderIDs, attackerIDs;
	private int opponentID;
	private int PREDICTION_DISTANCE = 30, MARKING_DISTANCE = 6, SHOOT_DISTANCE = 30;
	private double SAFE_RANGE = 10, SAFE_ANGLE = Math.PI / 6; // 30 degrees
	private PlayerAction lastBallAction;

	public Agent2() {
		super(teamName, teamColor1, teamColor2);
	}

	@Override
	public void move() {
		myGoalPost = assets.getPitch().getGoalPost(this.getId());
		opponentGoalPost = assets.getPitch().getOtherGoalPost(this.getId());
		opponentID = assets.getOtherTeamID(this.getId());
		MatchAssets nextPercept = assets.advance(PREDICTION_DISTANCE);
		myPlayers = assets.getPlayerList(this.getId());
		opponentPlayers = assets.getPlayerList(this.getId());
		Player owner = assets.getBall().getOwner();
		if ( owner != null ) {
			if ( owner.getTeamID() == this.getId() ) {	// I have the ball
				vector2D shootPoint = assets.getClosestPoint(opponentGoalPost, owner.getPosition());
				Player pToPass = null;
				if ( (shootPoint.measureDistance(owner.getPosition()) < SHOOT_DISTANCE) ) {
```

```
                                                owner.shoot(shootPoint);
                                                lastBallAction = owner.getIntent();
                                        }
                                        else {
                                                this.moveWithBall(owner);
                                                lastBallAction = owner.getIntent();
                                        }
                                        for(int i = 0; i < myPlayers.length; i++) {
                                                if ( myPlayers[i].getId() != assets.getBall().getOwner().getId() || (pToPass != null && pToPass.getId() != myPlayers[i].getId())) {
                                                        vector2D ownerMovement = owner.getPosition().subtract(owner.getStartupPosition());
                                                        myPlayers[i].move(myPlayers[i].getStartupPosition().add(ownerMovement));
                                                }
                                        }
                                }
                                else {          // Opponent has the ball
                                        manToManMarking(nextPercept);
                                }
                        }
                        else {          // Noone has the ball
                                Player closestPlayer = assets.getPlayerClosestToBall(myPlayers);
                                if ( closestPlayer.getDistanceFrom(assets.getBall()) < assets.getBall().getTrapDistance() ) {
                                        closestPlayer.trap();
                                        lastBallAction = closestPlayer.getIntent();
                                }
                                else {
                                        closestPlayer.move(nextPercept.getBall().getPosition());
                                }

                                for(int i = 0; i < myPlayers.length; i++) {
                                        if ( myPlayers[i].getId() != closestPlayer.getId() ) {
                                                mark(myPlayers[i], nextPercept.getPlayerClosestToObject(nextPercept.getPlayerList(opponentID), myPlayers[i]));
                                        }
                                }
                        }
                }

                private Player getStopper(MovingObject target, MatchAssets newAssets) {
                        if ( target != null && newAssets != null ) {
                                Rectangle2D.Double betweenArea;
                                if ( target.getPosition().getX() > myGoalPost.getX() ) {
                                        betweenArea = new Rectangle2D.Double(0, 0, Math.abs(target.getPosition().getX() - MARKING_DISTANCE), assets.getPitch().getPitchHeight());
                                }
                                else {
                                        betweenArea = new Rectangle2D.Double(target.getPosition().getX() + MARKING_DISTANCE, 0, Math.abs(myGoalPost.getX()-target.getPosition().getX()- MARKING_DISTANCE) , assets.getPitch().getPitchHeight());
                                }
                                Player[] betweenPlayers = newAssets.getPlayersInArea(betweenArea, this.getId());
                                if ( betweenPlayers != null ) {
                                        return newAssets.getPlayerClosestToObject(betweenPlayers, target);
                                }
                                else {
                                        return newAssets.getPlayerClosestToObject(newAssets.getPlayerList(this.getId()), target);
                                }
                        }
                        return null;
                }

                private void moveWithBall(Player owner) {
                        Player[] closestOpponents = assets.getPlayersInRange(owner.getPosition(), SAFE_RANGE, opponentID);
                        double goalAngle = assets.getClosestPoint(opponentGoalPost, owner.getPosition()).subtract(owner.getPosition()).getAngle();
                        double moveAngle = goalAngle;
                        if ( closestOpponents!= null && closestOpponents.length > 0 ) {
                                double[] anglesWithOwner = new double[closestOpponents.length];
                                for(int i = 0; i < closestOpponents.length; i++) {
                                        anglesWithOwner[i] = vector2D.calculateAngle(closestOpponents[i].getPosition(), owner.getPosition());
                                }
                                Arrays.sort(anglesWithOwner);
                                moveAngle = goalAngle;
                                for(int i = 0; i < anglesWithOwner.length; i++) {
                                        if ( (anglesWithOwner[i] - SAFE_ANGLE < goalAngle) && (anglesWithOwner[i] + SAFE_ANGLE > goalAngle)  ) {

                                                if ( Math.abs(goalAngle - (anglesWithOwner[i] - SAFE_ANGLE)) < Math.abs(goalAngle - (anglesWithOwner[i] + SAFE_ANGLE)) ) {
                                                        moveAngle = anglesWithOwner[i] - SAFE_ANGLE;
                                                }
                                                else {
                                                        moveAngle = anglesWithOwner[i] + SAFE_ANGLE;
                                                }
                                        }
                                }
                        }
                        vector2D direction = new vector2D(Math.cos(moveAngle), Math.sin(moveAngle));
                        owner.move( owner.getPosition().add(direction.multiply(owner.getSpeedAbility())) ) ;
                }


                private void mark(Player myPlayer, Player opponent) {
                        Rectangle2D.Double myGoalPost = assets.getPitch().getGoalPost(this.getId());
                        vector2D mgpCenter = new vector2D(myGoalPost.getCenterX(), myGoalPost.getCenterY());
                        double goalAngle = opponent.getPosition().subtract(mgpCenter).getAngle();
                        vector2D movePoint = new vector2D(opponent.getPosition().getX() - MARKING_DISTANCE * Math.cos(goalAngle), opponent.getPosition().getY() - 10 * Math.sin(goalAngle));
                        myPlayer.move(movePoint);
                }

                private void manToManMarking(MatchAssets nextPercept) {
                        ArrayList<Player> opponents = new ArrayList<Player>(Arrays.asList(opponentPlayers));
                        Player closestPlayer = getStopper(assets.getBall(), nextPercept);
                        if ( closestPlayer != null && closestPlayer.getDistanceFrom(assets.getBall()) < assets.getBall().getTrapDistance()) {
                                closestPlayer.trap();
                                lastBallAction = closestPlayer.getIntent();
                        }
                        else {
                                closestPlayer.move(assets.getBall().getPosition());
                        }
```

```
                        opponents.remove(assets.getBall().getOwner());

                        for(int i = 0; i < myPlayers.length; i++) {
                                if ( myPlayers[i].getId() != closestPlayer.getId() ) {
                                        Player closestOpponent = nextPercept.getPlayerClosestToObject(opponents.toArray(new Player[0]), myPlayers[i]);
                                        if ( closestOpponent != null && assets.getPitch().getHalfCourt(this.getId()).contains(closestOpponent.getPosition().toPoint()) ) {
                                                mark(myPlayers[i], closestOpponent);
                                                opponents.remove(closestOpponent);
                                        }
                                        else {
                                                myPlayers[i].move(assets.getBall().getPosition());
                                        }
                                }
                        }
                }

                @Override
                public Player[] assembleTeam(int numberOfPlayers, Rectangle2D.Double halfCourt) throws CengBallException {
                        Player[] myPlayers = new Player[numberOfPlayers];
                        double pDistance1 = (halfCourt.getHeight() - (2*numberOfPlayers/5) * Pitch.DEFAULT_PLAYER_RADIUS * 2) / ((2*numberOfPlayers/5) + 1 );
                        int remainingPlayers = numberOfPlayers - (4*numberOfPlayers/5);
                        double pDistance2 = (halfCourt.getHeight() - (remainingPlayers) * Pitch.DEFAULT_PLAYER_RADIUS * 2) / (remainingPlayers + 1 );
                        double pDistance3 = (halfCourt.getWidth() - numberOfPlayers*Pitch.DEFAULT_PLAYER_RADIUS * 2) / (numberOfPlayers+1);
                        int playerIndex = 0;
                        if ( halfCourt.getX() == 0 ) {
                                for (int i = 0; i < (2*numberOfPlayers/5); i++) {
                                        myPlayers[playerIndex] = new Player(names[playerIndex%5], playerIndex+1, this.getId(), new vector2D(halfCourt.getWidth() -
(pDistance3),pDistance1*(i+1) + Pitch.DEFAULT_PLAYER_RADIUS),
                                                                                this.generateSkillPoint(15),    // speed
                                                                                this.generateSkillPoint(20),    // shoot
                                                                                this.generateSkillPoint(5),                     // pass
                                                                                this.generateSkillPoint(10),    // dribbling
                                                                                this.generateSkillPoint(9),     // stamina
                                                                                this.generateSkillPoint(1));    // tackle
                                        playerIndex++;
                                }
                                for (int i = 0; i < (2*numberOfPlayers/5); i++) {
                                        myPlayers[playerIndex] = new Player(names[playerIndex%5], playerIndex+1, this.getId(), new vector2D(halfCourt.getWidth() -
(pDistance3*3),pDistance1*(i+1) + Pitch.DEFAULT_PLAYER_RADIUS),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10));
                                        playerIndex++;
                                }
                                for (int i = 0; i < remainingPlayers; i++) {
                                        myPlayers[playerIndex] = new Player(names[playerIndex%5], playerIndex+1, this.getId(), new vector2D(halfCourt.getWidth() -
(pDistance3*2),pDistance2*(i+1) + Pitch.DEFAULT_PLAYER_RADIUS),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10));
                                        playerIndex++;
                                }
                        }
                        else {
                                for (int i = 0; i < (2*numberOfPlayers/5); i++) {
                                        myPlayers[playerIndex] = new Player(names[playerIndex%5], playerIndex+1, this.getId(), new vector2D(halfCourt.getX() +
(pDistance3),pDistance1*(i+1) + Pitch.DEFAULT_PLAYER_RADIUS),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10));
                                        playerIndex++;
                                }
                                for (int i = 0; i < (2*numberOfPlayers/5); i++) {
                                        myPlayers[playerIndex] = new Player(names[playerIndex%5], playerIndex+1, this.getId(), new vector2D(halfCourt.getX() +
(pDistance3*3),pDistance1*(i+1) + Pitch.DEFAULT_PLAYER_RADIUS),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10));
                                        playerIndex++;
                                }
                                for (int i = 0; i < remainingPlayers; i++) {
                                        myPlayers[playerIndex] = new Player(names[playerIndex%5], playerIndex+1, this.getId(), new vector2D(halfCourt.getX() +
(pDistance3*2),pDistance2*(i+1) + Pitch.DEFAULT_PLAYER_RADIUS),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10),
                                                                                this.generateSkillPoint(10));
                                        playerIndex++;
                                }
                        }

                        return myPlayers;
                }
}
```

Yapay zekâ modülünden orta seviyede örnek Takım sınıfı kodu.

iii. Görselleştirme modülü

Gerçekleme parçasının oluşturup, sabit diske kaydettiği kayıt dosyalarını okuyarak, gerçekleştirilen müsabakanın iki boyutlu görselleştirilmesini sağlayan parçadır. Bu parça aracılığıyla, oyun gerçek zamandan çok daha hızlı bir gerçekleme aşamasından sonra, gerçek zamanlı olarak veya istenilen hızlarda kullanıcıya izlettirilebilecektir. Kaydedilmiş oyun dondurulup, tekrar başlatılabilecek, zaman dilimleri şeklinde ilerleyerek analiz edilebilecektir. Görselleştirme parçasının ihtiyacı olan veri, bir KayıtDosyası sınıfı nesnesinden oluşan fiziksel bir veri dosyasıdır. Her bir zaman dilimi sırasında gerçekleme parçası tarafından Algı sınıfı nesnesi oluşturulacaktır. Oyunun gerçeklenmesi aşaması tamamlandığında, gerçekleme parçası içerisinde, tüm zaman dilimlerine ait Algı sınıfı nesneleri bulunacaktır. Bunlar KayıtDosyası sınıfının bir nesnesi altında tutulmaktadırlar. Bu nesne ise google-gson kütüphanesinin yardımıyla JSON dosyası şeklinde oluşturulup, sıkıştırılarak saklanacaktır. Diske yazılacak bu dosya 2 dakikalık bir görselleştirme için yaklaşık 2MB büyüklüğünde olacaktır. Simülasyon tarafından kullanılmak üzere, diske yazılan verinin boyutunu azaltmak için kullanılan sınıf tanımları mevcuttur. Bunlardan MetaVeri sınıfı bir sonuç dosyasına en başta ve bir kere yazılmaktadır. Görselleştirme parçası Swing (Java) kütüphanesinden nesneler kullanan alt sınıflar barındırmaktadır. Görselleştirme bu sınıf nesneleri aracılığıyla yapılmaktadır.

b. Veri Tasarımı

Sistem, parçalar arası iletişim için 2 adet veri tipine sahiptir. Bunlar gerçekleme ve görselleştirme parçalarının bilgi alışverişinde kullandığı KayıtDosyası ve gerçekleme ve yapay zekâ kodlarının iletişimi için kullandığı Algı nesneleridir.

i. Algı verisi

Bu veri tipinden oluşturulan nesneler, gerçekleme parçası tarafından yapay zekâ algoritmalarına bilgi gönderirken kullanılmaktadır. Algı, genel olarak oyun içerisindeki bütün nesnelerin anlık pozisyon ve hız bilgisidir. Oyun, sırası gelen kullanıcıdan aldığı girdiye göre pozisyonları ve hızları değiştirir, nesnelerin o anda sahip oldukları hız ve pozisyon verilerini bir Algı objesi haline getirir ve kayıt eder. Bu sebeple her gerçekleme sonucunda içerisinde bütün nesnelerin pozisyonlarının bulunduğu bir Algı listesi oluşur. Algı içerisindeki bölümler şöyledir: oyuncuların pozisyon ve hız bilgileri, topun pozisyon ve hız bilgileri, maçın anlık skoru.

ii. Simülasyon sonuç dosyası

Simülatör parçası gerçekleme sürecinde oyun içerisindeki nesnelerin durumlarını belirli bir formata göre kaydetmektedir. Bu şekilde gerçekleme içerisinde gerçekleşen bütün olaylar görselleştirme parçasına bir dosya olarak aktarılır. Modüller arasındaki bağımsızlığı sağlayan temel veri tipidir. Görselleştirme parçasının bir simülasyonu görselleştirmesi için bilmesi gereken tek veri, dosyanın disk üzerinde depolandığı yer bilgisidir. Simülasyon sonuç dosyası, KayıtDosyası sınıfından google-gson kütüphanesi kullanılarak oluşturulan bir JSON formatında fiziksel dosyadır.

KayıtDosyası içerisindeki bölümler şu şekildedir: kullanıcıların kendi takımları için belirledikleri takım ismi ve takım renkleri, takım içerisindeki oyuncuların isimleri ve forma numaraları, gerçeklemenin oluşturduğu Algı listesi.

c. Arayüz Tasarımı

i. Ana ekran

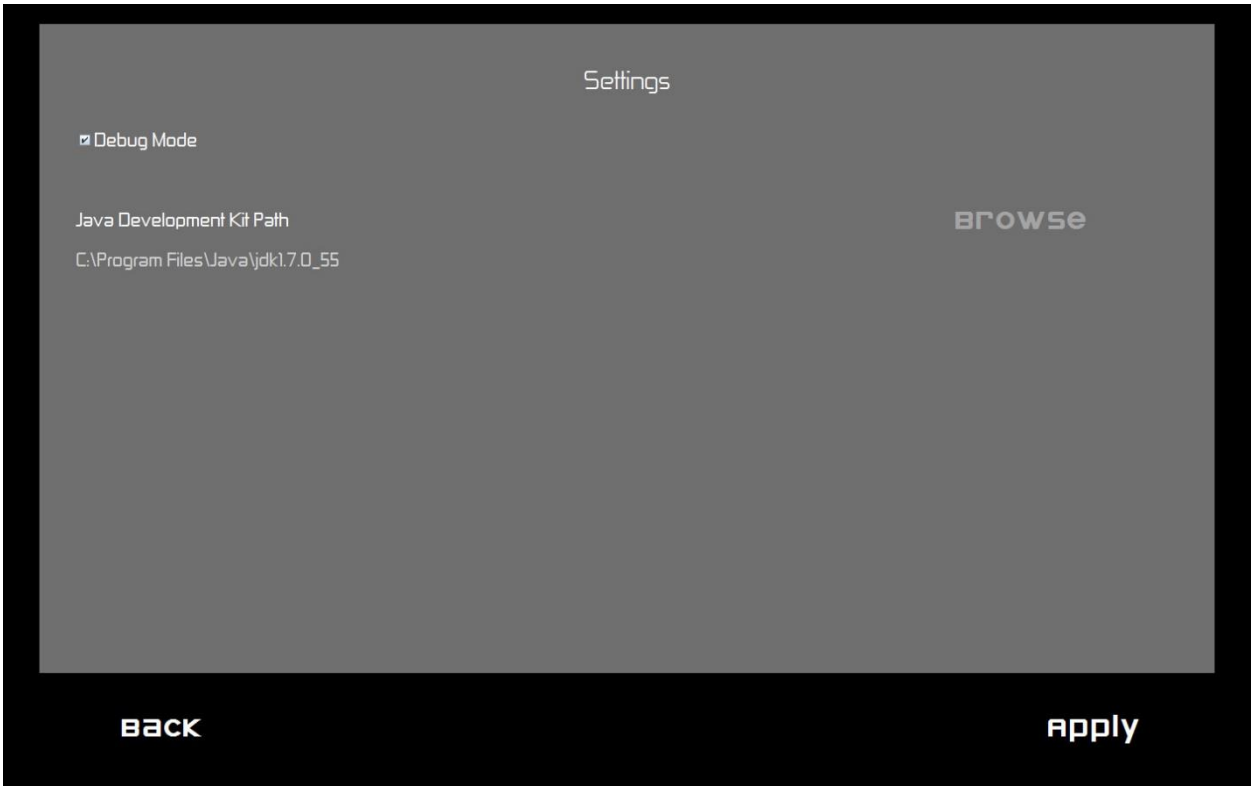
Oynun giriş ekranıdır. 5 seçenekli bir menü şeklinde tasarlanmıştır. Bu seçenekler sırasıyla Maç Başlatma, Maç İzleme, Kod Yükleme, Ayarlar ve Çıkış'tır.

Şekil 2:Ana ekran görüntüsü

ii. Ayarlar ekranı

Ayarlar ekranında kullanıcı tarafından seçilebilecek bir Hata ayıklama modu bulunmaktadır. Bu seçenek işaretlendiği takdirde oyun içerisinde karşılaşılan bilgilendirme ve hata mesajları fiziksel bir kayıt dosyası aracılığıyla saklanmaktadır.

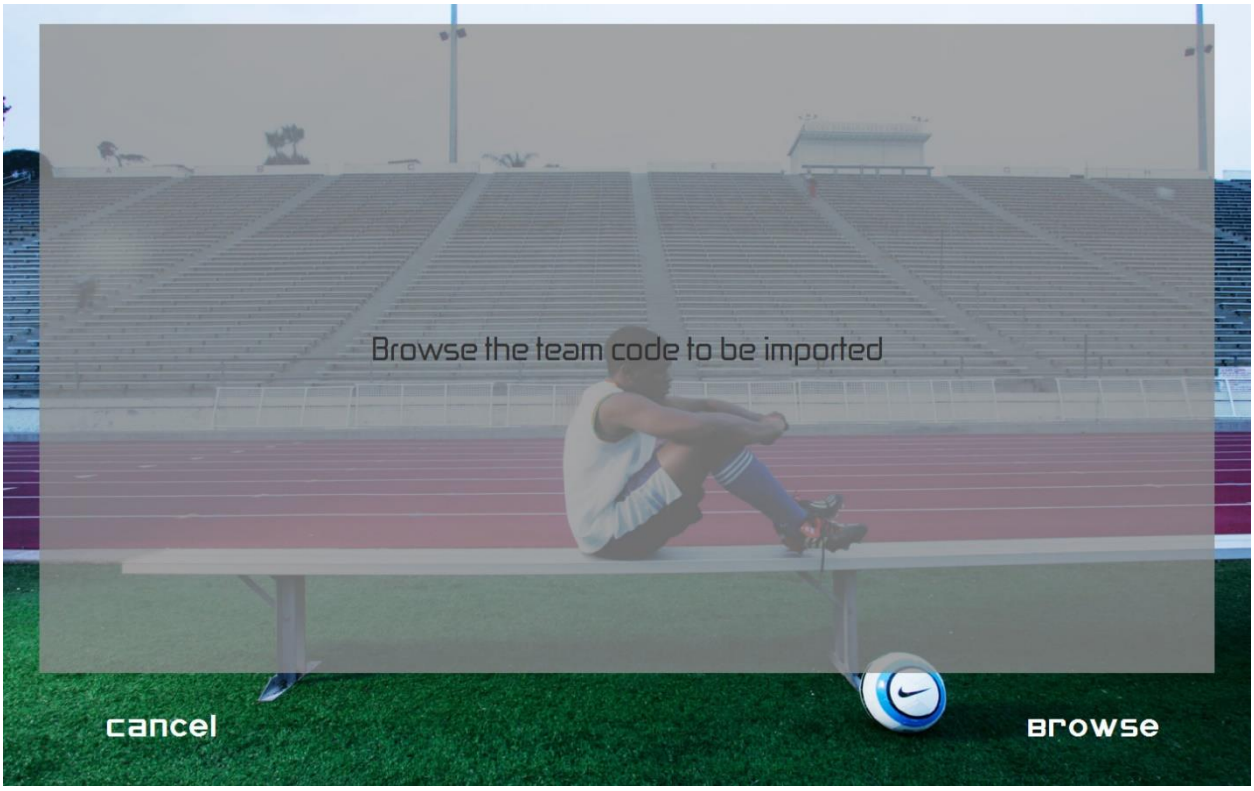
Ayrıca, kullanıcı tarafından belirtilmesi gereken Java Development Kit uygulamasının dizininin ayarlaması da burada yapılmaktadır.



Şekil 3: Ayarlar ekranı

iii. Kod yükleme ekranı

Kod yükleme aracılığıyla kullanıcı yapay zeka kodu sisteme dahil edilmektedir. Burada kod kontrolleri sonrasında, derleme işlemi yapılmaktadır. Ayarlar menüsünden Java Development Kit dizinin seçilmiş olması gerekmektedir.



Şekil 4: Kod yükleme ekranı

iv. Maç izleme ekranı

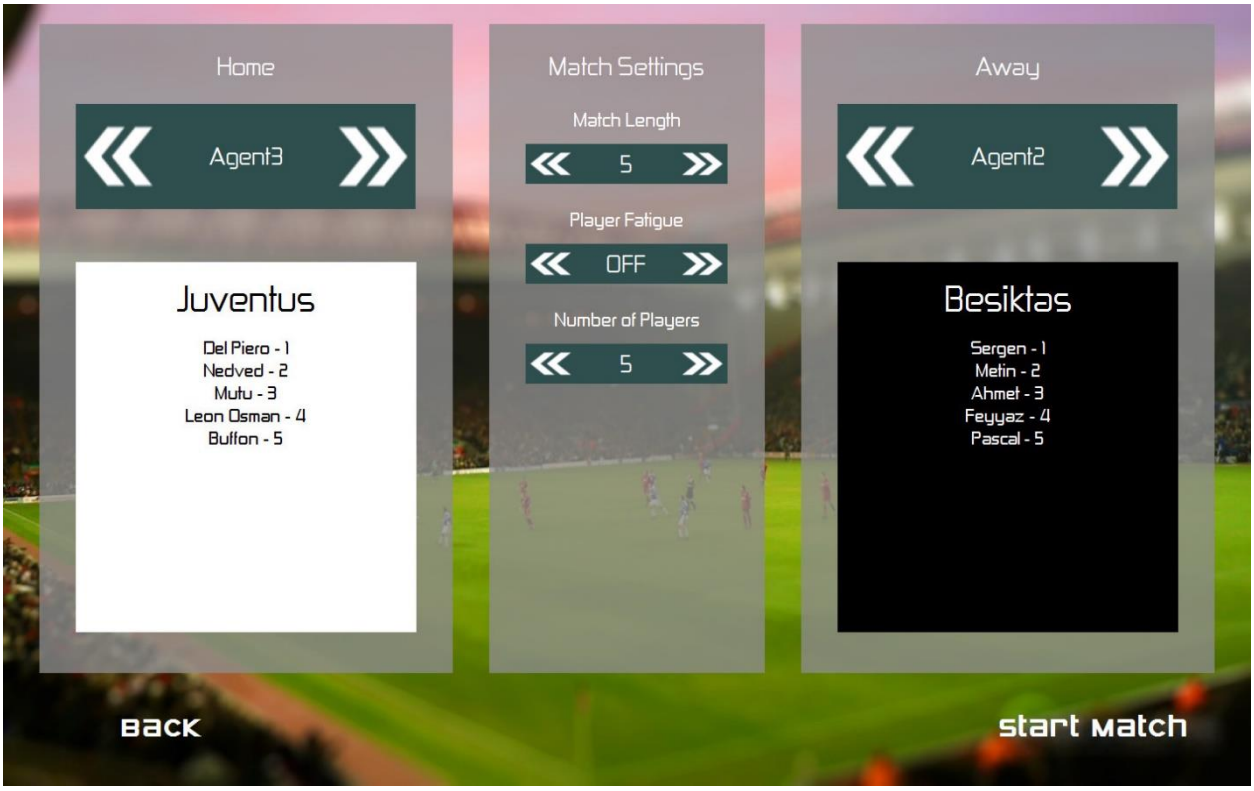
Burada daha önceden gerçeklenmiş oyunların oyun sonu kayıt dosyaları listelenmektedir. Kullanıcı buradan ilgili karşılaşmayı seçerek maç izleme ekranına geçiş yapabilmektedir.

Şekil 5: Maç izleme ekranı

v. Maç başlatma ekranı

Sisteme dahil edilen yapay zeka algoritmalarının ardından bu ekranda ilgili takımlar listelenmektedir. Kullanıcı bu kodlardan iki tanesini karşılıklı olarak seçerek oynu başlatabilmektedir.

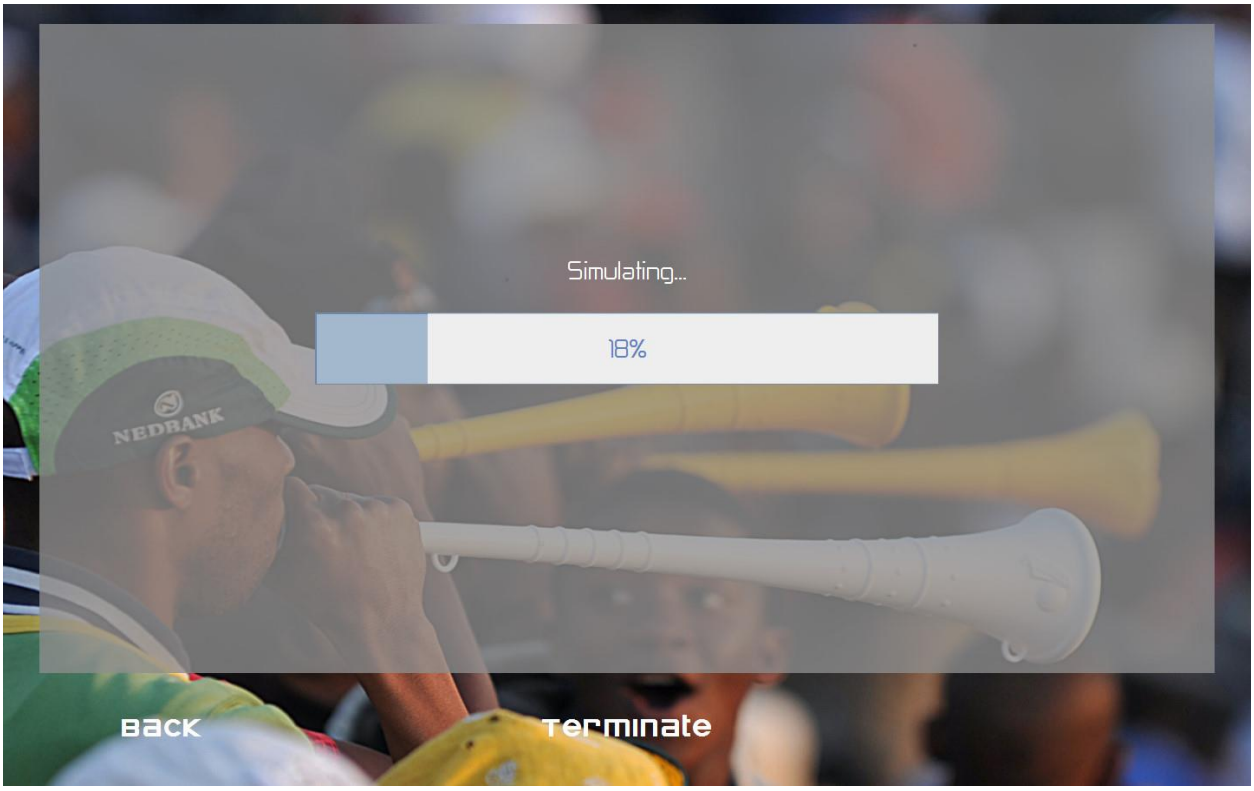
Ayrıca maç süresinin ne kadar olacağını, maç içi yorulma özelliğinin geçerliliğini ve takım başına düşen oyuncu sayısını bu ekrandan ayarlamak mümkündür.



Şekil 6: Maç başlatma ekranı

vi. Simülasyon ekranı

Maç başlatma ekranından yapılan ayarlar ışığında simülasyon bu ekrandan başlatılabilmektedir. Simülasyon süreci yüzde grafiği yardımıyla takip edilebilmektedir. Ardından simülasyon süreci tamamlanan karşılaşmalar maç izleme ekranı aracılığıyla izlenebilmektedir.



Şekil 7: Simülasyon ekranı

vii. Maç ekranı

Maç ekranı görselleştirme modülü kapsamında oluşturulmuş iki boyutlu futbol görsellemesini içermektedir. Bu ekranda simülasyon tarafından oluşturulmuş kayıt dosyası okunarak, mücadele ekrana taşınmaktadır.

Şekil 8: Maç ekranı

4. Gerçekleştirim

Kullanıcı girdisinin doğrudan kod aracılığıyla olmasından dolayı, hem tasarımda hem de kodlama kısmında üstünde durulan en önemli kısım sistemin güvenilirliği olmuştur. Kullanıcı kodunun sisteme dâhil edilmesi, bu kodun, sistemi bozacak komutlar göndermesine fırsat sunmaktadır. Bu noktada, sistemi her şartta ayakta tutabilmek adına çalışmalar yapılmış ve kodlama bu şekilde sürdürülmüştür.

Kullanıcının, oyun kuralları dışına hareket etmesini engellemek amacıyla, oyun içinde kullanılan dinamik değişkenlerin referansları kullanıcıya doğrudan verilmemiş, kopyalar oluşturularak paylaşılmıştır. Aynı şekilde, kullanıcıdan alınan bu değişkenlerdeki kural dışı davranışlar göz ardı edilmiştir.

Kullanıcı kodunun, sistem gerçeklemesi sırasında hata alması veya durması durumu sistemin çalışmasını engellememektedir. Kullanıcı kodu, ana işlem üzerinde değil, arka planda çalışan farklı işlemler ile çalıştırılmaktadır. Kullanıcı kodunda oluşan bir hata gerçekleme işlemini durdurmaz. Gerçekleme işlemi, kullanıcı kodu işlemleri üzerinde söz sahibidir. Bu noktada, efendi – köle tasarımı kullanılmıştır. Kullanıcı kodunun, gerçeklemeyi oluşturmak için yeterli nesneleri oluşturamaması durumunda, sistem bu nesneleri varsayılan değerleriyle oluşturur.

Kodlama aşamasında üzerinde durulan bir diğer konu da kullanıcıya yapay zekâ yazma sırasında kolaylık sağlamaktır. Bu noktada, kullanıcının daha çok teoriler üzerinde çalışmasını sağlamak adına kullanıcıya fiziksel hesaplamalar gerektiren fonksiyonlar oyun içi dinamik değişkenleri barındıran bir nesne aracılığıyla verilmiştir. Bu nesne aracılığıyla, kullanıcı gelecek olayları öngörebilir ya da o andaki önemli bilgileri bu fonksiyonlar ile kullanabilir.

Sistemin ara yüzü olabildiğince basit tutulmuş, ara yüzün gerekli görevleri yerine getirmesi üzerinde durulmuştur. Kullanıcı kolay bir şekilde kodunu sisteme dâhil edebilir, önceden gerçekleştirilmiş maçları izleyebilir ya da sisteme dâhil ettiği kodlar arasında gerçekleme oluşturabilir.

Sistemin kolayca değiştirilebilir ve geliştirilebilir olması kodlama aşamasında önemli bir etken olmuştur. Bu hususta, sistem nesne merkezli olup kolayca genişletilebilir. Oyunun değişkenliği kullanıcının isteğine bırakılmıştır. Takım başına düşen oyuncu sayısı, yorulma etkeni ve maç uzunluğu gerçekleme işlemi öncesi değiştirilebilir.

Oyunun adil ve gerçekçi olması adına, oyuna şans faktörü eklenmiştir. Bu durum, aynı takım kodlarıyla farklı sonuçlar elde edilmesine olanak sağlamıştır. Kullanıcının bu durumu düşünerek yapay zekâ kodu yazması gerekmektedir.

Son olarak, sistemde oluşabilecek hataların daha sonra çözülebilmesi için sistem çalışırken hataların bir dosyaya çıkarılması sağlanmıştır. Bu özellik, kullanıcıya ayarlar ekranı üzerinden bir seçenek olarak sunulmuştur. Kullanıcı ayarları da ayrıca bir şekilde saklanmaktadır.

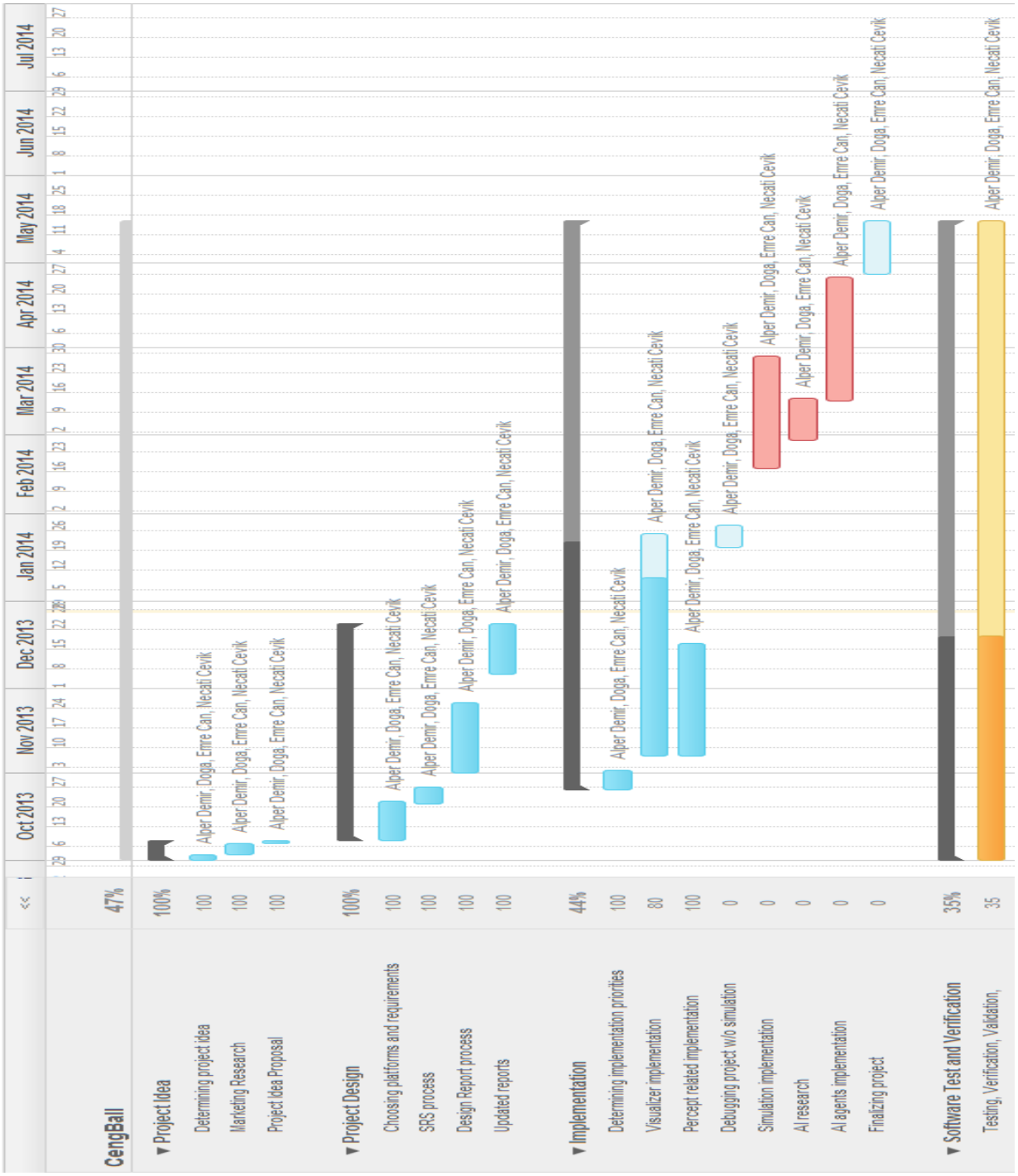
5. Konfigürasyon

Sistem, Java desteği olan herhangi bir platformda sorunsuzca çalışabilmektedir. Oldukça küçük bir dosya boyutuna sahip olup, çalıştırılabilir JAR dosyası olarak kullanıcıya sunulacaktır. Sistem önemli bir grafik gereksinimi barındırmaz. Çalıştırılmasıyla birlikte, gerekli olan dosyaları oluşturur. Kullanıcının takım kodu oluşturması için sağlayacağımız kütüphane, herhangi bir Java kod geliştirme platformuna dâhil edilebilir ve bu sayede takım kodu oluşturulabilir. Oluşturulan takım kodu, Java geliştirme dizininin sisteme tanıtılması şartıyla kolayca derlenip sisteme eklenebilir.

6. Proje Takvimi

Proje takviminde belirlenen tarihlere geliştirme süreci boyunca uyulmaya özen gösterilmiştir. Proje sonucu itibariyle hedeflenen süre tutturulmuştur.

Şekil 8'de proje başlangıç aşamasında belirlenen ve süreç boyunca uygulanan takvim yer almaktadır.



Şekil 9:Proje takvimi

7. Değerlendirme

a. Ölçeklenebilirlik

Uygulama bünyesinde kavram olarak futbol oyununun basitleştirilmiş bir hali seçilmiştir. Fakat nesnel tabanlı tasarım korunduğu için uygulama kavramı, karşılıklı mücadelenin bulunduğu ve kontrolün, rakiplerin nesneleri bazında yapıldığı her türlü alana kaydırılabilir. Rakip durumundaki kontrol edicilerin ve bu kontrol edicilerin özgül nesnelerinin sayısının önemli ölçüde artması durumunda, gerçekleme bu durumdan 2 şekilde etkilenmektedir. Birincil olarak, gerçeklemenin çok izlekli bir yapı barındırması ve her bir izleğin bir takıma değiştirilebilir bir üst zaman sınırı koyması sebebiyle, gerçekleme süresi uzamaktadır. Fakat bu süre, seçilen kavramın durum uzayının büyüklüğüne göre ayarlanabilir olduğu için istenilen sonucun elde edilmesini etkilememektedir. İkincil olarak ise, gerçekleme tarafından serileştirilerek, fiziksel olarak saklanan veri dosyasının boyutunda artış meydana gelmektedir. Fakat bu dosya boyutunu Java programlama dilinin, değişkenleri, nesnelerin kalıcı durumları hakkında bilgi içermediğini ifade etmek için kullandığı transient önekini kullanarak düşürmek mümkündür. Ayrıca oluşturulan JSON tipindeki dosya, sıkıştırma algoritmaları kullanılarak beşte birine kadar küçültülebilmektedir.

b. Güvenlik

Uygulama kapsamında Algı verisinin bütünlüğü gerçekleme parçası tarafından korunmaktadır. Gerçekleme, ömür süresi boyunca, doğruluğunu korumakla yükümlü olduğu verilerin kopyasını kendi bünyesinde, dışarıya kapalı bir şekilde saklamaktadır. Dış etmen olarak nitelendirilebilecek yapay zekâ kodları ile gerçekleme arasındaki veri trafiği; yapay zekâ kodlarının kendilerine sunulan ve gerçekleme tarafından da erişilebilen değişkenleri (niyet) ayarlaması şeklinde olmaktadır. Simülatör bu değişkenleri, sakladığı birincil veriler bazında değerlendirmektedir. Simülatör tarafından yapay zekâ kodlarına periyodik olarak aktarılan verinin bütünlüğü ise, yapay zekâ kodlarına istediği zaman çağırabildiği metotlar sağlanarak korunmaya çalışılmıştır. Kullanıcı girdisi, doğrudan kod aracılığıyla olduğundan dolayı, bu kodun yapabilecekleri konusundaki güvenlik konuları önemlidir. Bu noktada, kullanıcı kodunun sistemi bozamaması için, proje kapmasında oyun çalışır haldeyken kullanıcı kodunu derleyip sisteme dâhil etmek mümkündür. Derleme işlemi, kullanıcıya, takım kodunu yazması için gerekli olan sınıf kodlarını içeren bir kütüphane aracılığıyla yapılmaktadır. Bu kütüphane, kullanıcının çalıştığı kod yazma platformunda, gerekli altyapıya sahip olmasını sağlayacaktır. Ayrıca, kullanıcı kodunu sınırlamak adına, Java güvenlik kütüphaneleri kullanılacak, kullanıcının oyunu durdurması veya kapatması engellenecektir.

c. Değerlendirme sonucu

Oyun, tarafımızca, kullanıcı bakış açısıyla birçok kez denenmiştir. Testler sırasında, kullanıcı kodu oluşturulurken gerekli olan fonksiyonlar üzerine çalışılmış ve bu fonksiyonlar kütüphaneye eklenmiştir. Takım kodu oluşturma konusunda amaçlanan kolaylığın sağlandığı düşünülmektedir.

Takım kodunun sisteme eklenmesi ve sistemde gerçeklenmesi için sistem defalarca denenmiş, sistem açıkları giderilmiş ve kullanıcıya güvenilir bir sistem sunulmuştur. Ara yüzdeki ihtiyaçlar göz önüne alınmış ve bu ihtiyaçlar doğrultusunda değişikliklere gidilmiştir.

Geliştirilen takım kodunun yaptıklarının izlenmesi heyecan vericidir. Bu noktada, isteğimizi gerçekleştirmiş olmamız, geldiğimiz noktada bizi memnun etmektedir.

8. Sonuçlar

CengBall projesi, hedeflendiği noktaya kadar tamamlanmış olup, geliştirilmeye devam etmektedir. Sistemin dayanıklılığı üzerine testler yapılmış ve olumlu sonuçlar alınmıştır. Geliştirme sürecinde planlanan, ara yüzün geliştirilmesi ve yapay zekâ örneklerinin oluşturulmasıdır.

CengBall projesi, kullanıcılara yapay zekâ alanında kendilerini sınamaları için fırsat veren ve programlama yeteneklerini göstermelerini sağlayan bir projedir. Basit futbol tabanı, futbolun popülerliği ve izlemesi zevkli bir oyun olması dolayısıyla seçilmiştir. Bu noktada, programlamaya ilgisi olan ve futbol seven insanların ilgisi hedeflenmiştir. Proje, temelde iki takımı karşılıklı oynatacak bir platform oluşturmuş ve oyun türünü değiştirilebilir kılmıştır. Bu hususta, gelecekte aynı platform kullanılarak farklı oyun türlerinde de yapay zekâ geliştirmek ve oynamak mümkündür.

9. Projenin Özgünlüğü

a. Yenilikçilik

CengBall projesi, yaygın olan oyun kavramına önemli bir alternatif sunmaktadır. Programlama oyunu çalışmaları dünya genelinde az olup ülkemizde yok denecek seviyededir. Programlama öğrenmenin çok önemli olduğu bu zamanlarda, oyun kavramına yeni bir soluk getirmektedir.

Proje sadece bir oyun olmamakla beraber, akademik bir altyapı da barındırmaktadır. Yapay zekâ odaklı olan projemiz, en başından beri ticari bir amaçtan çok akademik bir amaca sahiptir. Yapay zekâ ve programlamaya ilgisi olan insanlara bu bilgilerini sınama şansı vermektedir. Bu sınama süreci, sınavlar aracılığıyla değil, eğlenceli bir oyun üzerinden gerçekleştirilmiştir.

b. Uygulanabilirlik

Futbolun ve Java'nın yaygınlığı, oyunu ön plana çıkarmaktadır. Başlangıçta projenin hedef kitlesinin dar olduğu düşünülse de, Java programlamanın yaygınlaşması ve futbolun sevilen bir spor olması projemizi ilgi çeker hale getirmektedir. Bilgisayar bilimi merkezli üniversite bölümlerinde, yapay zekâya olan ilgiyi karşılamak adına CengBall turnuvaları düzenlenebilir ya da MIT Battlecode [3] örneğindeki gibi öğrencilere ders olarak sunulabilir.

c. Kullanışlılık

Sistemin çok az yer kaplamakta ve kolayca çalıştırılabilmektedir. Java desteği olması dışında platform gereksinimlerinden bağımsızdır. Takım kodu geliştirilmesi aşamasında kullanıcıya yardımcı olması için basit rehberler ve kod nesnelerinin açıklandığı Wiki sayfası hazırlanacaktır. Kullanıcı kolayca örnek takım kodlarına ulaşabilecek, onlar üzerinden kendi takım kodunu geliştirebilecektir.

10. Referanslar

[1]	TORCS programlama oyunu	http://torcs.sourceforge.net/
[2]	The RoboCup Soccer Simulator	http://sourceforge.net/projects/sserver/
[3]	MIT BattleCode yarışması ve dersleri	https://www.battlecode.org/