

## **RboT: İsel 2-Boyutlu Dairesel Robot Benzetim Paketi**

Mehmet Akgöl, EE 2005

Elektrik-Elektronik Mühendisliđi Bölümü  
Bođazii Üniversitesi  
Bebek 34342 İstanbul

Proje Danışmanı: Prof.Dr. H. Işıl Bozma

Haziran 2005

## İçindekiler

1	GİRİŞ .....	3
2	PROBLEM TANIMI.....	4
3	KURAMSAL YAKLAŞIM .....	4
4	YAZILIM TASARIMI.....	6
4.1	Amaç .....	6
4.2	Tanımlar & Kısaltmalar .....	6
4.3	Tasarım Elemanları .....	7
4.3.1	Sınıflar .....	7
4.3.2	Sınıf İşlevleri .....	7
4.3.3	Bağımlılıklar .....	10
4.4	“component” Nesneleri .....	10
4.5	RboT.....	12
5	BENZETİM SONUÇLARI.....	13
6	SONUÇ .....	13
7	KAYNAKÇA .....	14
A	Kurulum Aşamaları .....	16
B	RboT KULLANICI KILAVUZU .....	17
B.1	Bilgi Paneli (IP).....	18
B.2	Benzetim Ortamı (WSP) .....	19
B.2.1	Add Component (Bileşen Ekle) .....	20
B.2.2	Run All (Tümünü Çalıştır) .....	24
B.2.3	Pause All (Tümünü Durdur).....	26
B.2.4	Stop All (Tümünü Deaktive Et) .....	26
B.3	Bileşenler (Components).....	27
B.3.1	Run (Çalıştır).....	27
B.3.2	Pause (Durdur) .....	27
B.3.3	Stop (Deaktive Et).....	27
B.3.4	Remove (Benzetim Ortamından Kaldır) .....	27
B.3.5	Component Properties (Bileşen Özellikleri) .....	27
B.4	Liste Paneli(LP).....	29
B.5	Kontrol Paneli (CP).....	29
B.5.1	Preferences Dialog Box (PDB-Seçenekler Ara Yüzü).....	30
C	Runge-Kutta Metodları .....	33
C.1	Klasik Dördüncü Derece Runge-Kutta Metodu .....	33

## Şekiller

Şekil 1:	Robotun görsel alan özellikleri. Robot sadece kırmızı renkli alanı görebilir. ....	11
Şekil 2:	Bir bileşenin çalışma statüleri arasındaki geçişleri gösteren durum diagramı.....	11
Şekil 3:	Taşıyıcı robotların hareket stratejisini gösteren durum diagramı. ....	11
Şekil 4:	RboT’u oluşturan nesnelere .....	12
Şekil 5:	RboT’un bileşenleri .....	17
Şekil 6:	Bilgi Paneli.....	18
Şekil 7:	WSP de farenin sağ tuşuna tıklanıldığında gösterilen açılır menü .....	19
Şekil 8:	New Component (Yeni Bileşen) Ara Yüzü .....	20
Şekil 9:	Robotun görsel alan özellikleri. Robot sadece kırmızı renkli alanı görebilir. ....	21
Şekil 10:	Kullanıcının yeni bileşen için girmiş olduğu değerler .....	23

Şekil 11: “Navigator 1- Yöngüder 1” robotunun ortama eklenmesinden sonra <i>RboT</i> un durumu. Dikkat edilirse yeni bileşenin eklenişi hem <i>bilgi</i> hem de <i>liste paneleri</i> tarafından algılanmıştır. Ayrıca <i>kontrol paneli</i> de aktif hale geçmiştir.....	23
Şekil 12: Bileşenin özellikleri. Kullanıcı bileşenin tipini belirtmemiş ve <i>radius</i> kısmına da tamsayı yerine harf dizisi girmiştir.....	24
Şekil 13: Oluşturulan hata mesajı.....	24
Şekil 14: Benzetim ortamına üç adet <i>yöngüder</i> robot eklendikten sonra <i>RboT</i> un durumu. Şekilden de görüldüğü gibi hem <i>liste</i> hem de <i>bilgi panelleri</i> güncellenmiştir.....	25
Şekil 15: WSP’deki tüm bileşenler çalışırken alınan anlık düzenleşim. Hem <i>bilgi</i> hem de <i>liste</i> panellerinde bileşenlere eşlik eden, bileşenin çalışma durumunu gösteren renklerin sarıdan yeşile geçtiğine/dönüştüğüne dikkat ediniz.....	25
Şekil 16: Kullanıcının “Stop All” komutunu çalıştırmışından sonra <i>RboT</i> un durumu. Hem <i>bilgi</i> hem de <i>liste panellerinde</i> bileşenlere eşlik eden, bileşenin çalışma durumunu gösteren renklerin yeşilden kırmızıya geçtiğine/dönüştüğüne dikkat ediniz.....	26
Şekil 17: Yöngüder robotu için ortaya çıkan “Bileşen Bilgi Ara Yüzü”. Bileşenin ismi CIDB’nin başlık kısmında görülmekte.....	28
Şekil 18 : Kontrol Paneli .....	29
Şekil 19: PDB-Seçenekler Ara Yüzü .....	31

## 1 GİRİŞ

Bu projenin amacı bir düzlem üzerinde hareket eden dairesel robotların geribeslemeye dayalı olarak programlanmaları sonucu oluşan davranışlarını benzetim yolu ile incelemektir. Şu ana kadar bu konu hakkında bazı çalışmalar yapılmış olsa da, bu projede izlenen yöntem ve uygulanan metotlar bir çok yeni husus içermektedir.

Robotların hareketlerini kontrol etmek amacıyla yapılan çalışmalar robotların ortaya çıkmasıyla başlamıştır. Ama burada robot hareketinden kastedilen, robotun yolunu dışarıdan bir yardım olmadan robotun kendisinin bulmasıdır. Bu amaca ulaşabilmek için yapay zeka, bilgisayarlı görme ve robotbilim alanları gelişmiştir.

Robot hareketi aslında klasik bir robotbilim problemidir. Klasik yöntemler genelde açık-döngü yaklaşımlarına dayanmaktadır. Bu yaklaşımlarda, herhangi bir robotun hareket sistemi, robotun izleyeceği yolu önceden hesaplamaya ve daha sonra robotun bu hesaplanan noktalar üzerinden hareket etmesine dayanmaktadır. Ancak, ortamın dinamik olduğu ve dolayısı ile sürekli olarak değiştiği durumlarda, bu yöntemin robotun izleyeceği yolun adaptasyonuna olanak sağlamaması bir dezavantaj olarak ortaya çıkmaktadır. Robotun bulunduğu ortamda herhangi bir değişiklik olursa, robotun izleyeceği yolun güncellenmesi gerekir. Aksi takdirde robotun hedef noktasına ulaşması mümkün olmamaktadır.

Alternatif olarak, kapalı döngü yaklaşımlarda robotların gidecekleri yörüngeler ansal olarak belirlenir. Bu projede de böyle bir yaklaşım kullanılmaktadır. Bu yaklaşımda, robotun hareket planı önceden hesaplanmamakta, bunun yerine anlık kontrol hareketleri tamamen ortamdan gelen geribildirim ile oluşturulmaktadır. Bu yöntem yapay gizilgüç işlevlerine dayanmaktadır. Eğer bir yapay gizilgüç işlevin bir yöngüdümlü işlev olduğu gösterilirse, robotların hedef noktalarına ulaşacakları veya imkansız koşullarda robotların hareketlerini durduracakları garanti edilmiştir.

Bu projenin amacı iki-boyutlu daire şeklindeki robotların geribildirim stratejisi kullanılarak çalıştığı bir benzetim ortamı geliştirmektir. Geribildirim metodunun robot hareketleri üstündeki etkilerini gözlemlemek amacıyla *RboT* geliştirilmiştir. Bu yöntemde robotların yörüngeleri yapay gizilgüç işlevler kullanılarak tamamen dinamik biçimde hesaplanmıştır. Burada uygulanan yöntem açık-döngü metodundan çok kapalı-döngü metodudur. Robotun her hareketinde, robotun bir sonra hareket edeceği nokta ortamda bulunan diğer robotlar ve engeller göz önüne alınarak hesaplanmıştır. Robotların hareketleri birbirinden bağımsız olduğu için her robotun yörüngesi değişmiş ve yeniden hesaplanmıştır.

Şimdiye kadar bu konu hakkında benzer ama basit projeler yapılmış olsa da bu proje yapılmış olan projelerden bazı farklılıklar göstermektedir.

- Şu ana kadar yapılan tüm çalışmalarda ortamda tek tip robot kullanılmıştır. Bu projede ise iki çeşit robot birden kullanılmıştır.
- Ayrıca ortamda aynı tipten birden fazla robot olduğu durumlar göz önüne alınmıştır.

İki tip robot vardır:

- Yöngüder Robotlar: Bu robotların amacı kendilerine belirtilen hedef noktalarına gitmektir.
- Taşıyıcı Robotlar: Bu robotların amacı ise sabit parçaları hedef noktalarına taşımaktır.

Daha önceden ortamda sadece *yöngüder* robotlar bulunduğu zaman, bu robotların kendi hedef noktalarına ,diğer robotlarla hiç bir çarpışma olmadan, ulaşacakları ispatlanmıştır [1]. Ayrıca sadece bir tane *taşıyıcı* robotla yapılan başka bir benzetimde de *yöngüder* robotlar için uygulanan yöntemin *taşıyıcı* robotlar için de uygulanabileceği gözlemlenmiştir [4].

## 2 PROBLEM TANIMI

Farz edelim ki ortamda  $p(=p1+p2)$  { $p1$  *yöngüder* robot sayısı,  $p2$  *taşıyıcı* robot sayısı} tane daire şeklinde robot olsun. Her robot birbirinden bağımsız olarak hareket etmektedir. *Yöngüder* robotlar hedef noktalarına ulaşmaktan, *taşıyıcı* robotlar ise sabit parçaları buldukları yerden hedef noktalarına götürmekten sorumludurlar. Burada yapılan varsayımlar:

- Her robotun ideal, sınırlı tork verme kabiliyeti vardır
- Her robot kendi pozisyonunu her zaman bilmektedir.
- Her robotun bir görüş alanı vardır. Görüş alanı iki değişken tarafından kontrol edilmektedir:
  - $r$ : robotun görebileceği alanın yarıçapı
  - $\theta$ : robotun görebileceği alanın merkez açısı. Bu açı pozitif-x eksenine göre ölçülmektedir.
- Her robot kendi görüş alanı içerisindeki diğer robotların yerlerini ve boyutlarını bilmektedir.

Ayrıca  $b$  vektörünü tüm robotların durumlarını gösterecek vektör olarak tanımlayalım. Burada durum vektörünü kontrol edecek denklem  $b' = u$  olarak kabul edilmiştir. Şimdi biz *yöngüder* robotları herhangi bir ilk pozisyondan kendi hedef noktalarına ulaştırabilecek ve *taşıyıcı* robotların da tüm sabit parçaları hedef noktalarına taşıyabilecek bir  $u$  giriş değeri arıyoruz . Bunun için  $Q(b(t)) : F \rightarrow [0,1]$  bir eşleme tanımlanmıştır. Buradan  $u$ ,  $u = -\nabla Q$  ( $\nabla Q$ ,  $Q$ 'nun  $b$ 'ye göre gradyanıdır) eşitliği kullanılarak bulunabilir.

## 3 KURAMSAL YAKLAŞIM

Bu projede temel amaç sisteme verilecek olan  $u$  giriş değerinin bulunmasını sağlayacak olan  $Q$ , yöngüdümlü işlevini bulmaktır [1].

Varsayalım ki  $Q$

$$Q(b) = \sigma_d \circ \sigma \circ \psi^b$$

olsun. Bu eşitlikte  $\psi^b$  hem hedef noktaya olan uzaklığı hem de serbest uzay sınırını belirlemektedir.  $\psi^b$   $\partial F$ 'e bağlı olduğundan dolayı (serbest uzay sınırı  $\partial F$ 'e bağımlıdır)  $\psi^b$  kabul edilebilir bir değer değildir.  $\psi^b$ 'yi kabul edilebilir bir değer yapmak için,  $\psi^b$ 'ya  $\sigma(x) = x/(x+1)$  işlevi uygulanmaktadır. Bu durumda elde edilen işlev kabul edilebilir bir işlev olduğu halde, hedef nokta dejenere kritik bir nokta olmaktadır. Hedef noktayı dejenere olmaktan çıkarmak için daha önce elde edilmiş olan işleve  $\sigma_d(x) = x^{1/k}$  işlevi uygulanmaktadır. Bu son durumda elde edilen  $Q$  işlevi hem kabul edilebilir bir değer almakta hem de dejenere olmayan bir minimum hedef noktaya sahip olmaktadır [1]. Bu projede kullanılan bazı aday işlevler vardır.

Bu işlevler robotların hızlarını belirlemektedirler. Robotların çeşitlerine bağlı olarak bu işlevler:

$$Q=\alpha^k/\beta$$

▪ Yöngüder robotlar için

$$\circ \alpha = \sum_{i=1}^N (x_{ri}-x_{rit})^2 + (y_{ri}-y_{rit})^2$$

$$\circ \beta = \prod_{i=1}^N \prod_{j=i+1}^N \beta_{ij} ;$$

$$\beta_{ij} = (x_{ri}-x_{rj})^2 + (y_{ri}-y_{rj})^2 - (r_{ri}+r_{rj})^2$$

▪ Taşıyıcı robotların parça ile eşleşmesi için

$$\circ \alpha = (x_r - x_{rt})^2 + (y_r - y_{rt})^2$$

$$\circ \beta = \prod_{i=1}^N (x_r - x_i)^2 + (y_r - y_i)^2 - (r_r + r_i)^2$$

▪ Taşıyıcı robotların parçayı hareket ettirmesi için (burada robotun m<sup>inci</sup> parçayı hareket ettirdiği farz edilmiştir.)

$$\circ \alpha = (x_{pm} - x_{rt})^2 + (y_{pm} - y_{rt})^2 + \sum_{i \neq m}^N (x_i - x_{it})^2 + (y_i - y_{it})^2$$

$$\circ \beta = \prod_{i=1, i \neq m}^N (x_r - x_i)^2 + (y_r - y_i)^2 - (r_r + r_i)^2 * \prod_{i=1, i \neq m}^N (x_{pm} - x_i)^2 + (y_{pm} - y_i)^2 - (r_m + r_i)^2$$

Bu eşitliklerde kullanılan kısaltmalar:

$x_{ri}$  ve  $y_{ri}$  i<sup>inci</sup> yöngüder robotun şimdiki pozisyonu

$x_{rit}$  ve  $y_{rit}$  i<sup>inci</sup> yöngüder robotun hedef noktası

$r_{ri}$  i<sup>inci</sup> yöngüder robotun yarıçapı

$x_r$  ve  $y_r$  taşıyıcı robotun şimdiki pozisyonu

$x_{rt}$  ve  $y_{rt}$  taşıyıcı robotun hedef noktası

$r_r$  taşıyıcı robotun yarıçapı

$x_i$  ve  $y_i$  i<sup>inci</sup> parçanın şimdiki pozisyonu

$x_{it}$  ve  $y_{it}$  i<sup>inci</sup> parçanın hedef noktası

$r_i$  i<sup>inci</sup> parçanın yarıçapı

$$x_{pm} = x_r + l * \cos\theta \text{ ve } y_{pm} = y_r + l * \sin\theta$$

Burada  $l = r_r + r_m$  ve  $\theta$  taşıyıcı robot ile m<sup>inci</sup> parçanın merkezleri arasındaki açı

Tüm robotların hareket planı:

$$r' = - \partial Q / \partial r, \text{ r yol olmak üzere}$$

kullanılarak hesaplanmaktadır. Robot  $r'$  sıfır olunca durmaktadır [1][2].

## 4 YAZILIM TASARIMI

### 4.1 Amaç

Bu projede oluşturulan yazılımın, *RboT*, amacı kullanıcıya değişik robot düzenlemelerini çalışabilecek sanal bir ortam sağlamaktır. Burada kullanıcının robot çeşidine göre bir robotun nasıl çalıştığını bildiği varsayılmıştır. *RboT* kullanıcının istediği zaman sanal ortamda kolaylıkla değişiklik yapmasına olanak vermektedir. Benzetim programını çalıştırabilmek için robotların çeşitleri hakkında herhangi bir kısıtlama yoktur. Kullanıcı istediği zaman, istediği noktaya *Parça*, *Yöngüder Robot*, veya *Taşıyıcı Robot* ekleyebilir. Bunun yanında GUI(Grafik Ara Yüzü) kullanıcın kolaylıkla anlayabileceği ve kullanabileceği bir biçimde tasarlanmıştır. Ayrıca hatalar çok katı bir biçimde kontrol edilmiştir. Bu durumda kullanıcının hata yapma olasılığı sıfıra yakın bir değerde tutulmuştur.

*RboT*'un çalışması için herhangi özel bir ortama ihtiyacı yoktur. Fakat *RboT* JAVA programlama dilinde yazıldığı için, *RboT*'un çalışacağı makinede java yazılım geliştirme ortamı(*software development kit - sdk*)'nın ve java sanal makinesi (*java virtual machine - jvm*)'nin kurulu olması gerekmektedir.

### 4.2 Tanımlar & Kısaltmalar

Bu dökümantasyonda kullanılacak bazı tanımlar ve kısaltmalar aşağıda belirtilmiştir.

- **RboT** : Yazılım Adı
- **component (bileşen)**: Benzetim ortamındaki herhangi bir nesne: Parça, Yöngüder Robot, veya Taşıyıcı Robot
- **WSP** : Benzetim ortamı. Tüm bileşenler burada işlev görürler.
- **LP** : Liste paneli. Benzetim ortamındaki tüm bileşenler burada listelenir.
- **CP** : Kontrol paneli. Benzetimle ilgili kontrol tuşlarını kapsar.
- **IP** : Bilgi paneli. Bileşenlerin durumları hakkında bilgileri gösterir.
- **EL** : İşlev nesnesi. *Rbot*'un tüm elemanlarından gelen istemler bu nesne tarafından ele alınır.
- **NCDB** : Yeni bileşen ara yüzü. Kullanıcıya yeni bir bileşen eklemesi için bir menü gösterir.
- **CIDB** : Bileşen bilgi ara yüzü. Kullanıcın daha önce oluşturmuş olduğu bir bileşenin özelliklerini gösterir. Ayrıca kullanıcının bileşenin özelliklerini değiştirebilmesine olanak sağlar.
- **PDB** : Seçenekler ara yüzü. Kullanıcının benzetim ortamının özelliklerini görebilmesine ve değiştirebilmesine olanak sağlar.
- **NP** :Yeni bileşen nesnesi. *Taşıyıcı* robotların hareket ettirmek istediği bir sonraki bileşenin seçiminden sorumludur.
- **LC** :Pozisyon hesaplayıcı nesnesi. Ortamdaki robotların bir sonraki hareket noktalarını hesaplar.

### 4.3 Tasarım Elemanları

Bu bölümde *RboT*'u oluşturmak için kullanılan tüm elemanlar ve bu elemanların özellikleri verilecektir. JAVA nesne tabanlı bir programlama dili olduğundan dolayı oluşturulan tüm elemanlar nesnedir.

#### 4.3.1 Sınıflar

Tasarımda kullanılan sınıflar aşağıda belirtilmiştir.

- icon
- component
- workSpacePanel
- listPanel
- render
- controlPanel
- eventListener
- dataBank
- componentInfoDialogBox
- color
- colorRender
- newComponentDialogBox
- preferencesDialogBox
- nextPart
- point
- goTargetGradient
- matePartGradient
- movePartGadiant
- locationCalculater
- RboT
- infoPanel

#### 4.3.2 Sınıf İşlevleri

Bu bölümde 4.3.1'de verilen tüm sınıfların işlevleri açıklanacaktır.

- ✚ **icon:** Bu sınıf benzetim ortamında (*WSP*) bulunan bileşenlerin şekillerini, ve eğer varsa hedef noktalarını, *LP* ve *IP* tarafından kullanılan daire şekillerini oluşturmak için kullanılır. Unutulmamalıdır ki tüm bileşenler daire şeklindedir. Eğer bir bileşenin hedef noktası varsa bu nokta da daire şeklinde gösterilmiştir.
- ✚ **component:** Bu nesne *WSP*'de eleman oluşturmak için kullanılır. Oluşturulan elemanlar parça, yöngüder robot veya taşıyıcı robot olabilir. *component* sınıfının özellikleri bir elemanın türünü ve çalışma statüsünü belirtir. Bu sınıfın bazı özellikleri statik olaak tanımlanmıştır. Statik olarak tanımlanan bu özellikler tüm bileşen nesnelere için ortak olup, dizayn esnasında sıkça kullanılmışlardır. Statik özellikler ve anlamları aşağıda belirtilmiştir.

- **PART** : bileşenin parça olduğunu belirtir.
- **NAVIGATOR** : bileşenin yöngüder robot olduğunu gösterir
- **MOVER** : bileşenin taşıyıcı robot olduğunu gösterir
- **RUNNING** : bileşen çalışıyor
- **PAUSED** : bileşen durdurulmuş
- **STOPPED** : bileşen deaktive edilmiş
- **STATIONARY** : bileşen sabit



- ✚ **workSpacePanel:** Bu sınıf oluşturulan bileşenleri taşımaktan sorumludur. Tüm bileşenler hareketlerini bu nesnenin belirlemiş olduğu alan içinde yaparlar. Sınıf bileşenlerin hareketi için 500x500'lük bir alan tanımlamıştır

Bu sınıftan oluşturulan nesne *WSP'dir*. Herhangi bir anda, herhangi bir pozisyoda, *WSP*'ye bileşen eklenebilir. Ayrıca *WSP* tüm bileşenleri aynı esnada çalıştırmaya, durdurmaya, deaktive etmeye ve benzetim ortamından kaldırmaya yarayan işlevlere sahiptir.

- ✚ **listPanel:** Bu sınıf *WSP*'ye eklenen bileşenleri liste halinde göstermekten sorumludur.

Bu sınıf kullanılarak oluşturulan nesne *LP'dir*. *LP* birden fazla bileşen seçmeye olanak sağlayarak, seçilen bu bileşenlerin çalıştırılmasına, durdurulmasına, deaktive edilmesine ve benzetim ortamından kaldırılmasına *CP*'yle birlikte imkan verir. Ayrıca *LP*'de listelenen herhangi bir bileşenin üstüne fare ile çift tıklanması o bileşenin özelliklerini gösteren *CIDB*'nin ortaya çıkmasını sağlar.

- ✚ **render:** Bu sınıfın herhangi özel bir işlevinin olmamasına rağmen, *LP* nesnesinin düzgün çalışması için gereklidir.

- ✚ **controlPanel:** Bu sınıf benzetim ortamının kontrolü için gerekli olan tuşları içermektedir. Bu sınıf kullanılarak oluşturulan nesne *CP'dir*. *CP* yardımıyla *LP*'den seçilen bileşenler çalıştırılabilir, durdurulabilir, deaktive edilebilir veya benzetim ortamından, *WSP*'den, çıkarılabilir. Ayrıca kullanıcının herhangi bir düzenleşimi *WSP*'ye yüklemesine veya *WSP*'den kaydetmesine olanak sağlayan tuş takımına da sahiptir.

- ✚ **infoPanel:** Bu sınıf benzetim ortamı içerisinde meydana gelen çeşitli olaylar hakkında kullanıcıya bilgi vermek amacıyla tasarlanmıştır.

Bu sınıftan türetilen nesne *IP'dir*. *IP* tüm bileşenler tarafından meydana getirilen olaylar hakkında kullanıcıya bilgi verir. Bunun yanında *IP*, *WSP*'de bulunan parça ve robot sayısını ve robotların anlık çalışma durumlarını( çalışan, duran, deaktive edilen) gösteren bir tabloya da sahiptir.

- ✚ **eventListener:** Bu sınıf *RboT*'un çeşitli elemanlarından gelen olayların değerlendirmesinden ve bu olaylar karşısında yapılması gereken işlerin kontrolünden sorumludur. Bu sınıf *RboT*'ta bulunan hemen hemen tüm elemanlarla iletişim içindedir.

Bu sınıf kullanılarak oluşturulan nesne *EL'dir*. *EL* *RboT*'un tüm elemanlarından gelen istekleri değerlendirir ve eğer mümkünse bu isteklerin yapılmasını sağlar. *RboT* elemanları arasındaki iletişim de *EL* aracılığıyla yapılır. Kullanıcının *EL* ile doğrudan bir etkileşimi olmamaktadır.

- ✚ **datbank:** Bu sınıf *WSP*'de bulunan tüm bileşenlerin depolanmasından sorumludur. Bu sınıftan türetilen nesne *DB'dir*. *DB* tüm bileşenleri kendi bünyesinde barındırır. Bir bileşenin kendi komşu bileşenlerini öğrenmesi de *DB* aracılığıyla olmaktadır.

- ✚ **newComponentDialogBox:** Bu sınıf *WSP*'ye yeni bileşen eklemek için kullanıcıya ara yüz sunmaktadır.

Bu sınıftan türetilen nesne *NCDB*'dir. *NCDB* kullanıcının benzetim ortamına yeni bir bileşen eklemesini sağlar. Eğer yeni bir bileşen benzetim ortamına eklenirse *WSP*, *LP* ve *IP* güncellenir.

- ✚ **componentInfoDialogBox:** Bu sınıf daha önce benzetim ortamında bulunan bir bileşenin özelliklerini kullanıcıya, ara yüz kullanarak, göstermekten sorumludur.

Bu sınıftan türetilen nesne *CIDB*'dir. *CIDB* seçilen bileşenin özelliklerini gösterir ve kullanıcının bu bileşene ait özellikleri değiştirmesine olanak sağlar. Eğer bileşende herhangi bir değişiklik yapılırsa *WSP*, *LP* ve *IP* güncellenir.

- ✚ **color & colorRender:** Bu iki sınıfın özel işlevleri yoktur. Bu sınıflar *NCDB* ve *CIDB*'nin düzgün çalışması için gereklidir.

- ✚ **point:** Bu sınıf *WSP*'de bir noktayı temsil etmek için kullanılmıştır. Sınıf elemanları *ondalık* olarak belirtilmiştir. *JAVA* bir noktayı *ondalık* olarak belirtmediği için, bu sınıf matematiksel hesaplamalar için kullanılmıştır.

- ✚ **nextPart:** Bu sınıf *taşıyıcı* robotların taşıyacağı bir sonraki parçayı seçmekten sorumludur.

Bu sınıftan türetilen nesne *NP*'dir. *NP taşıyıcı* robota komşu olan parçalar arasından robota en yakın olan parçayı seçmekte ve bunu robota bildirmektedir.

- ✚ **goTargetGradient:** Bu sınıf *yöngüder* robotların hareketlerini belirlemek amacıyla oluşturulan fonksiyonun gradyanını bulmaktan sorumludur.

- ✚ **matePartGradient:** Bu sınıf *taşıyıcı* robotların parçaya yaklaşma hareketlerini belirlemek amacıyla oluşturulan fonksiyonun gradyanını bulmaktan sorumludur.

- ✚ **movePartGradient:** Bu sınıf *taşıyıcı* robotların parçayı taşıma hareketlerini belirlemek amacıyla oluşturulan fonksiyonun gradyanını bulmaktan sorumludur.

- ✚ **locationCalclater:** Bu sınıf robotun bir sonraki hareket noktasını belirlemekten sorumludur.

Bu sınıftan türetilen nesne *LC*'dir. Robotun türüne göre *LC* gradyan sınıflarından birini kullanarak, robotun hareket edeceği bir sonraki noktayı hesaplar. Robotun hareket planı oluşturulurken hesaplanan gradyan, uyarlamalı Cash-Karp Runge-Kutta metodu kullanılarak hesaplanmıştır. Cash-Karp Runge-Kutta metodu için ayrıntılı bilgi Ek-C'de verilmektedir.

- ✚ **RboT:** Bu sınıf benzetim ortamını oluşturan ana sınıftır. Gerekli olan tüm nesnelere bu sınıf içerisinde oluşturulmakta ve programa eklenmektedir.

### 4.3.3 Bağımlılıklar

Dokümantasyonun bu kısmında *RboT* elemalarının oluşturulma sırası verilecektir. Nesnelere oluşturulma sırası:

1. *DB*'nin oluşturulması. Tüm bileşenlerin komşu bileşenleri bilmesi gerekmektedir. Bu da sadece *DB* aracılığıyla olmaktadır.
2. *EL*'nin oluşturulması. *RboT*'un tüm elemanlarının birbiriyle olan iletişimi *EL* tarafından sağlanmaktadır. Bundan dolayı tüm *RboT* elemanlarının *EL* nesnesine ihtiyacı vardır.
3. *WSP*'nin oluşturulması. *WSP* bileşenlerin hareket edebilecekleri ortamı sağlamaktadır. *WSP* ile *LP* etkileşim içinde olduğundan dolayı *WSP*'nin *EL* nesnesine referansı vardır.
4. *LP*'nin oluşturulması. *LP*, *WSP*'de bulunan tüm bileşenleri listeler. *LP* hem *WSP*'den hem de *CP*'den gelecek olan istemleri yapar hem de *IP*'ye bilgi gönderir. Bundan dolayı *LP*'nin *EL* nesnesine referansı vardır.
5. *CP*'nin oluşturulması. *CP* benzetim ortamını kontrol edecek olan tuşları içermektedir. *CP* hem *WSP*'yle hem de *LP*'yle iletişim içindedir. Bu yüzden *CP*'nin *EL* nesnesine referansı vardır.
6. *IP*'nin oluşturulması. *IP* benzetim ortamında meydana gelen olaylar hakkında kullanıcıya bilgi verir. *IP* *LP*'yle etkileşim içindedir. Bu yüzden *IP*'nin *EL* nesnesine referansı vardır.

Burada dikkat edilmesi gereken nokta *WSP*, *LP*, *CP* ve *IP*'nin aynı *EL* nesnesine referanslarının olmasıdır. Aksi takdirde nesnelere arasındaki iletişim mümkün olmayacaktır. Oluşturulan bu altı nesne *RboT*'u meydana getirmektedir.

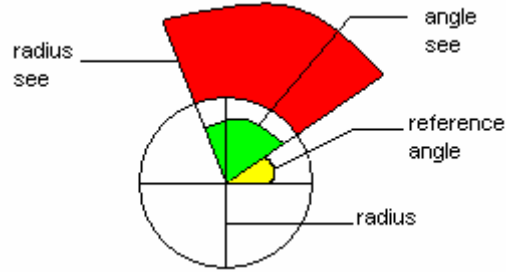
### 4.4 “component” Nesnelere

“component” nesnelere bu projenin temel nesnelereidir. “component” nesnelereinin özellikleri aşağıda belirtilmiştir.

- name : bileşenin ismi
- type : bileşenin tipi
- radius : bileşenin yarıçapı
- currentLoc : bileşenin şu anda bulunduğu konum
- targetLoc : bileşenin hedef noktası
- radiusSee : bileşenin görebileceği maximum radyal uzaklık
- angleSee : bileşenin görebileceği maximum açısal uzaklık
- referenceAngle: bileşenin görmeye başlayacağı açı değeri
- workStatus :bileşenin anlık çalışma durumu(hareket halinde, durdurulmuş, deaktive edilmiş veya sabit)
- neighbourComponents: bileşenin komşu bileşenleri. Komşu bileşenler, bu bileşenin görüş alanı içerisindeki bileşenlerdir.

- **neighbourParts** : bileşenin komşu parçaları. Komşu parçalar, bu bileşenin görüş alanı içerisindeki parçalardır.

Yukarıda belirtilen değerler Şekil 1’de gösterilmiştir.

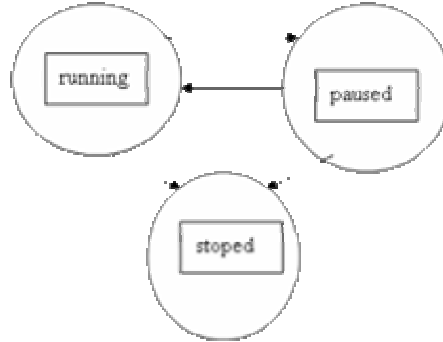


Şekil 1: Robotun görsel alan özellikleri. Robot sadece kırmızı renkli alanı görebilir.

*Yöngüder* veya *taşıyıcı* robotlar herhangi bir anda aşağıdaki çalışma durumlarından sadece bir tanesinde bulunabilirler:

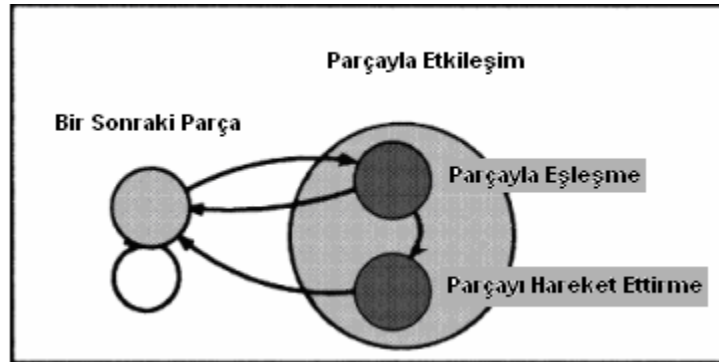
*RUNNING*(hareket halinde) *PAUSED*(durdurulmuş) *STOPPED*(deaktive edilmiş)

Bu durumlar arasında meydana gelen değişiklikleri gösteren durum diagramı Şekil 2’de gösterilmiştir.



Şekil 2: Bir bileşenin çalışma statüleri arasındaki geçişleri gösteren durum diagramı

*Taşıyıcı* robotun hareket planı da bir durum diagramında gösterilebilir. Bu durum Şekil 3’de belirtilmiştir.

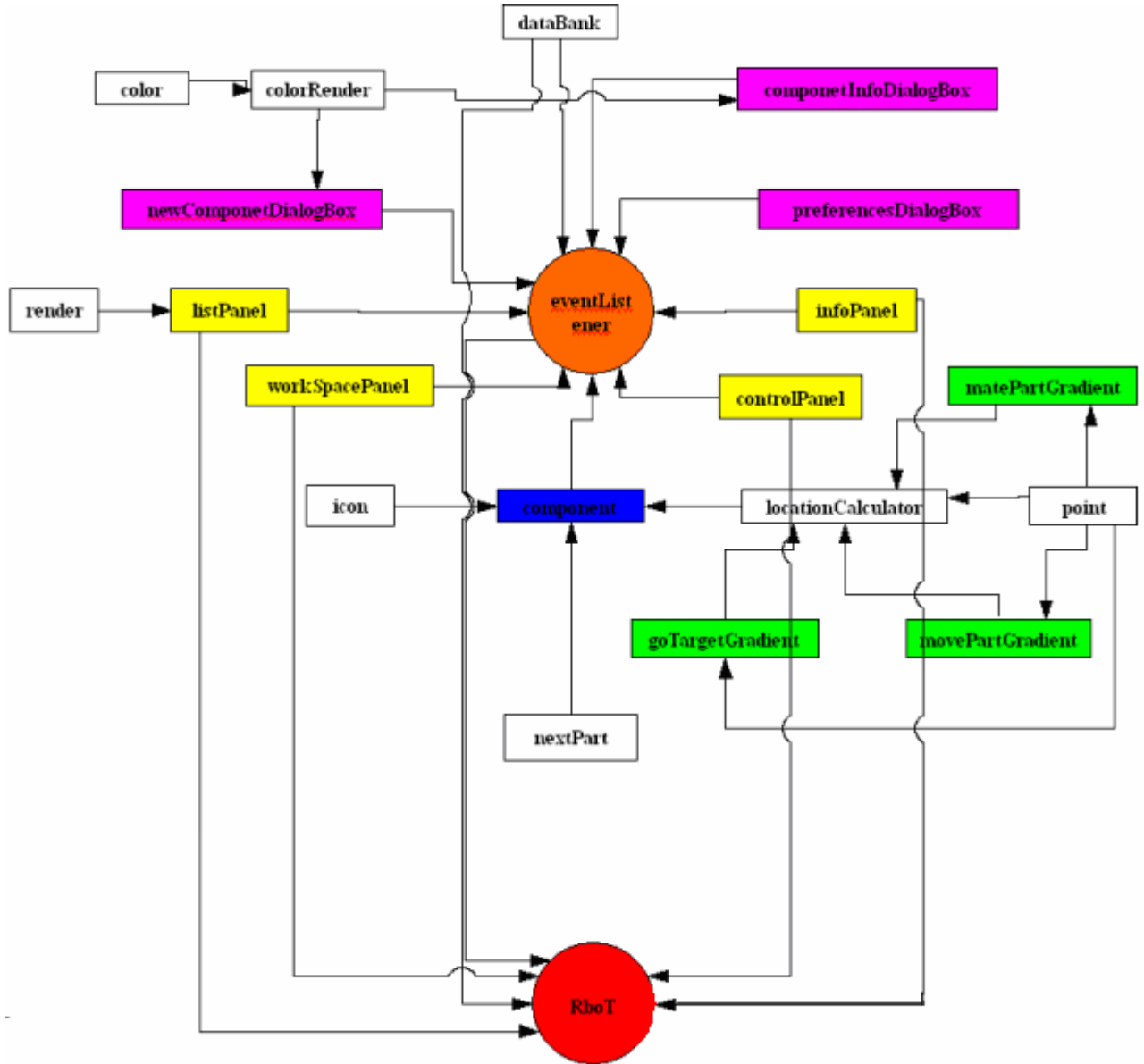


Şekil 3: Taşıyıcı robotların hareket stratejisini gösteren durum diagramı.

#### 4.5 RboT

*RboT* hem applet hem de normal uygulamalar için tasarlanmıştır. Fakat applet uygulamasında kullanıcı herhangi bir düzenleşimi, JAVA sanal makinesinin gerektirdiği zorunluluktan dolayı, bilgisayarından yükleyememekte ve bilgisayarına kayıt edememektedir.

*RboT*'u oluşturan nesnelere Şekil 4'de gösterilmiştir.



Şekil 4: *RboT*'u oluşturan nesnelere

## 5 BENZETİM SONUÇLARI

RboT oluşturulduktan sonra, bazı benzetim düzenleřimleri test edilmiřtir. Bu proje kendi alanında ilklerden biri olduėundan dolayı elde edilen sonular gelecek alıřmalar iin byk nem tařımaktadır.

İlk deneyde ortamda sadece *yngder* robotlar olduėu durum gzlemlenmiřtir. Dzenleřimlerin byk oėunluėunda robotlar arasında iletiřim olmadıėında, robotların hedef noktalarına ulařtıkları grlmřtir. Buna karřılık, robotlar arasında iletiřim saėlandıėında belirlenemeyen sebeplerden dolayı robotlar hedef noktalarına ulařamamıřtır.

Ařaėıda belirtilen tm deneyler robotlar arasında iletiřimin olmadıėı durumlarda yapılmıřtır. Ayrıca tařıyıcı robotlar iin yapılan alıřmalar devam etmektedir.

Bir bařka deney ise ortamda sadece *tařıyıcı* robotların olduėu durum iin gerekleřtirilmiřtir. *Tařıyıcı* robotlar arasındaki iletiřim matematiksel hesaplamalarda herhangi bir etki oluřturmadıėından dolayı, tm *tařıyıcı* robotların hedef paralarıyla buluřtukları gzlenmiřtir.

Bařka bir deney dzenleřiminde ise ortama para, *yngder* ve *tařıyıcı* robot beraber konulmuřtur. Yapılan bu deneyde robotların ilk konumlarının, robotlarının hareketlerini byk lde etkilediėi grlmřtir. Genellikle robotların hedef noktalarına ulařmalarına raėmen, bazı durumlarda, ortamdaki diėer bileřenler yznden, bir noktada takılı kaldıėı gzlenmiřtir.

řu ana kadar belirtilen tm deneyler oklu alıřma prensibine dayanmaktadır. Bazı deneylerde ise robotlar sırayla alıřtırılmıřtır. Bu deneyler de yukarıda olduėu gibi sadece *yngder*, sadece *tařıyıcı* robotların olduėu durumlar ile *yngder* ve *tařıyıcı* robotların aynı anda bulunduėu ortamlarda gerekleřtirilmiřtir. ok kompleks durumlar test edilmemesine raėmen, robotların oėunlukla hedef noktalarına ulařtıkları grlmřtir.

Ayrıca robotların hareket planının dinamik olarak hesaplandıėını gstermek amacıyla da deneyler yapılmıřtır. Bu deneylerde sadece *yngder* ya da *tařıyıcı* robotlar kullanılmıřtır. Her iki dzenleřimde de robotların hedef noktaları, robotlar hareket halindeyken deėiřtirilmiřtir. Bu durumlarda da robotlar hedef noktalarına ynelmiřlerdir.

Bir bařka deney ise *yngder* robotlar hedeflerine ulařacakları esnada yollarına sabit paralar konularak yapılmıřtır. Bu durumda ise robotlar bařka yollar bularak hedef noktalarına eriřmiřlerdir.

## 6 SONU

Robotların ilk olarak kullanılmaya bařladıėı zamandan bu yana robot hareketleri nemli arařtıma konularında biri olmuřtur. Ancak bu konu deėiřik disiplinler altında yapılan btn alıřmalara raėmen henz tam olarak zme ulařmıř deėildir.

Robot hareket kontrol iin gnmzde benimsenen temel yaklařım aık-dng stratejilerine dayanmaktadır. Bu yntemde robotun izleyeceėi yol, robot harekete bařlamadan nce hesaplanmaktadır. Bu durumda problem basit bir yrnge takip problemine dnřmektedir. Fakat bu yntem dinamik deėildir. Eėer robotun bulunduėu ortamda bir deėiřiklik yapılırsa, rneėin robotun nne bir engel konulursa, robotun hareket planının yeniden

hesaplanması gerekmektedir. Bu yüzden açık-döngü metodları kullanışlı değildir. Bu yöntem alternatif bir yaklaşım ise kapalı-döngü metodudur. Bu metotta robotun hareketi anlık olarak hesaplanmaktadır. Bu projede kapalı-döngü metodları kullanılmıştır. Bundan dolayı robot etrafındaki değişikliklere dinamik olarak, anında reaksiyon göstermektedir. Kullanılan bu yöntemde ortam ile ilgili tüm bilgileri barındıran yapay zihin işlevleri oluşturulmuştur. Bu dökümantasyonda kullanılan tüm navigasyon işlevleri ortamda başka bileşenlerin olduğunu hesaba katmıştır. Bu da robotun ortamdaki herhangi bir değişikliğe reaksiyon vermesine olanak sağlamıştır.

Navigasyon işlevlerinin getirdiği en büyük kısıtlama ise robotların şeklidir. Bu yöntemde ortamda bulunan tüm bileşenler, parçalar, *yöngüder* ve *taşıyıcı* robotlar, daire şeklinde olmalıdır. Fakat şu anda bileşenlerin şekilleri önem taşımamaktadır.

Bu projenin asıl amacı araştırmacılara değişik robot düzenlemelerini deneyecek bir benzetim ortamı sunmaktır. *RboT* bu amaçla tasarlanmıştır. *RboT* kullanıcıya deneylerini yapması için kullanımı kolay bir ortam sağlamaktadır.

*RboT* kullanılarak elde edilen benzetim sonuçları göstermiştir ki, çoğu düzenleme için robotlar hedef noktalarına ulaşmıştır. Burada en önemli husus deneylerin değişik tipte robotlarla ve dinamik bir ortamda yapılmasıdır. Bu robot hareketi alanında yapılan büyük bir gelişmedir.

Bu projede elde edilen sonuçlar ışığında, yakın bir zamanda robotların daha otonom bir şekilde hareket edebileceğini söylemek yanlış olmaz.

## 7 KAYNAKÇA

1. CS Karagöz, HI Bozma, and DE Koditschek, Coordinated motion of disk-shaped independent robots in 2D workspaces Ann Arbor: Univ. Michigan, Tech. Rep. CSE-TR-486-04, Feb. 2004.
2. C. Serkan Karagöz, H. Işıl Bozma t & Daniel E. Koditschek, Event-Driven Parts' Moving in 2D Endogenous Environments In *Proceedings of IEEE Int. Conference on Robotics & Automation*, San Francisco, CA, 2000
3. C. S. Karagöz, H. I. Bozma, and D. E. Koditschek, Edar - a mobile robot for parts' moving based on a game-theoretic approach *IEE Electronics Letters*, 38(3):147-149, 2002.
4. C. S. Karagöz, H. I. Bozma ,and D. E. Koditschek, Feedback-based event-driven parts' mover robot, *IEEE Transactions on Robotics*, 2004.
5. W. Press, S. Teukolsky, W. Vetterling, B. Flannery. Numerical Recipes in C, Cambridge University Press, 1994. NUMERICAL RECIPES IN C: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43108-5) Copyright (C) 1988-1992 by Cambridge University Press.
6. <http://en.wikipedia.org/wiki/Runge-Kutta>

EKLER



## A Kurulum Aşamaları

*RboT*'u kullanmak için CD'de verilen *RboTApplication* klasörünün içindeki *RboTApplication* dosyasının üstüne fare ile çift tıklamanız yeterlidir.

*RboTApplication* kendi kendine çalışabilen bir dosyadır. Kullanıcı bu dosyayı ya dosyanın üstüne çift tıklama yaparak yada komut satırına aşağıdaki komudu yazarak çalıştırabilir.

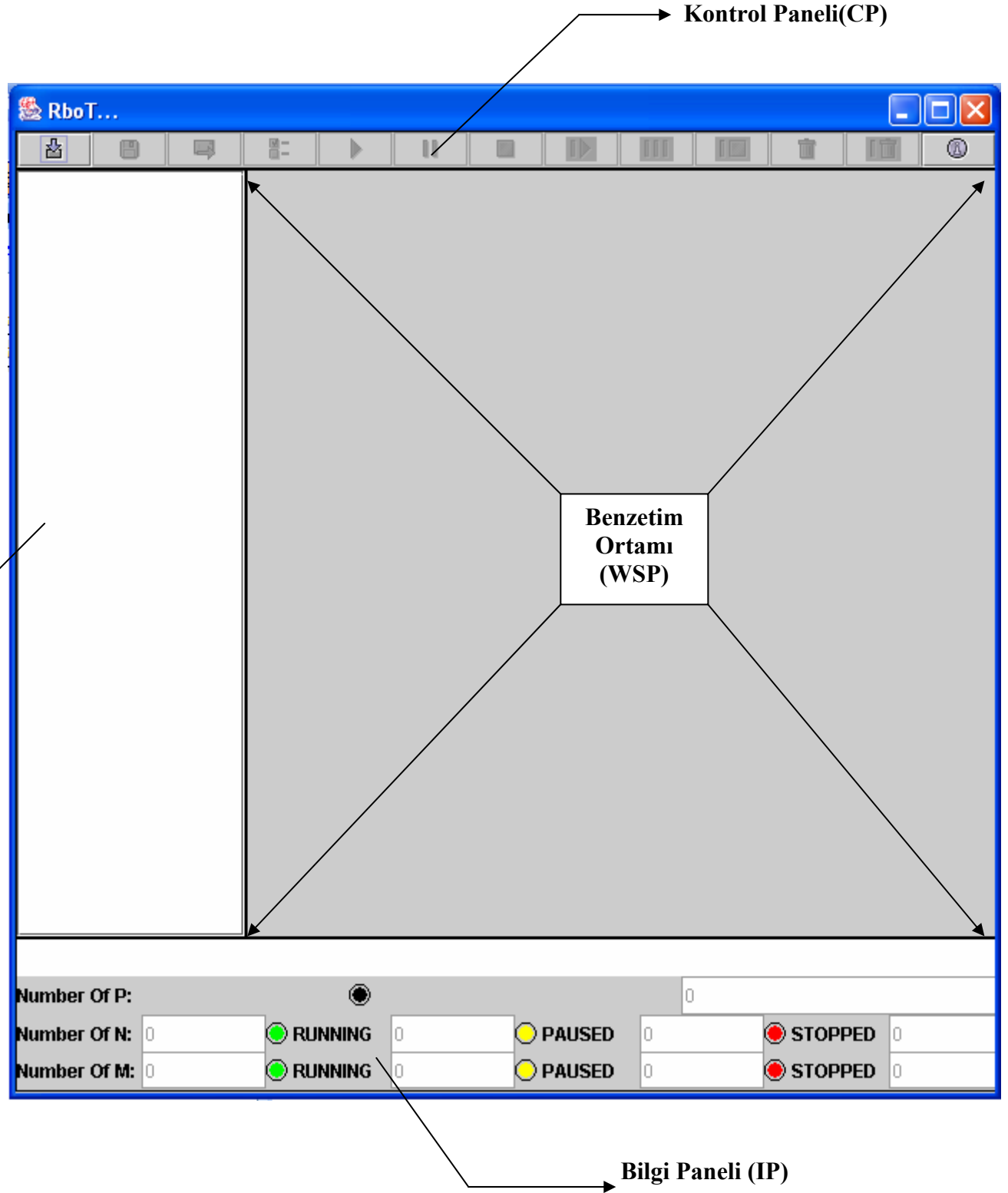
```
java -jar RboTApplication.jar
```

*RboT*'un çalışması için gerekli olan dosyalar *RboTApplication* klasörünün içinde verilmiştir. Bu dosyalar genelde .gif uzantılı dosyalar olup *RboT*'un arayüzünü oluşturmak için kullanılmışlardır. Ayrıca yine aynı klasörde *snapshots* adlı başka bir klasör bulunmaktadır. Bu klasör kullanıcının anlık düzenlemelerinin kayıt edildiği klasördür.

*RboT* yazılımının kullanabilmesi için, kullanıldığı bilgisayarda Java yazılım geliştirme ortamı(sdk) ve java sanal makinesi(jvm)'nin kurulu olduğu, ayrıca java 1.4 ve jar komutlarının kullanıcının işlev yolunda olduğu kabul edilmiştir. Java sdk ve jvm'nin olmadığı bir makinede *RboT*'un çalışması mümkün değildir. Ayrıca eğer kullanıcı java sdk'sını yüklemiş ama yukarıda belirtilen komut, kullanıcının işlev yolunda değilse, kullanıcının *RboTApplication* dosyasını ../j2sdk1.4.1/bin klasörünün içine kopyalaması ve yukarıda belirtilen komutu bu klasör altında çalıştırması gerekmektedir.

## B RboT KULLANICI KILAVUZU

RboT'u oluşturan bileşenler Şekil 5'de gösterilmiştir.



Şekil 5: RboT'un bileşenleri

*RboT*'un dört temel bileşeni vardır:

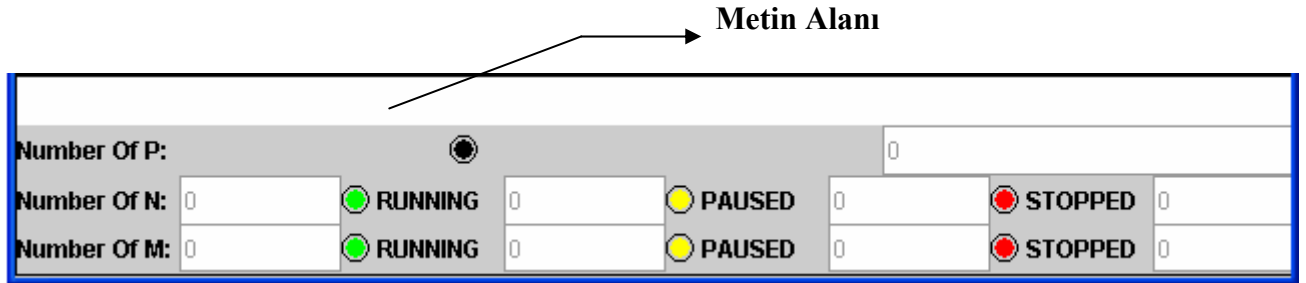
- Kontrol Paneli
- Liste Paneli
- Benzetim Ortamı
- Bilgi Paneli

*Kontrol Paneli* benzetim ortamını kontrol eden “çalıştır”, “geçici olarak durdur” veya “sona erdir” gibi, tuşları, barındırmaktadır. *Liste Paneli* benzetim ortamında bulunan tüm bileşenlerin isimleri ve tipleriyle birlikte listelendiği alandır. *Benzetim Ortamı* bileşenlerin eklendiği ve hareket ettiği alandır. *Bilgi Paneli* ise benzetim ortamında gerçekleşen çeşitli olaylar hakkında kullanıcıya bilgi gösteren paneldir.

*RboT*'u oluşturan bileşenleri açıklamadan önce, *RboT*'ta gösterilen bazı şekillerin anlamları verilecektir.

- : bileşenin bir sabit bir parça olduğunu gösterir
- : bileşenin çalışan bir robot olduğunu gösterir
- : bileşenin durdurulmuş bir robot olduğunu belirtir
- : bileşenin deaktive edilmiş bir robot olduğunu belirtir

### B.1 Bilgi Paneli (IP)



Şekil 6: Bilgi Paneli.

BP, benzetim ortamıyla ilgili bilgilerin görüntülediği paneldir. Dört kısımdan oluşur:

- *Metin Alanı*: Benzetim ortamında gerçekleşen çeşitli olaylar hakkında kullanıcıya mesajların gösterildiği alandır.
- *Number of P*: Benzetim ortamındaki parça sayısını gösterir.
- *Number Of N*: Benzetim ortamındaki yöngüder robot sayısını gösterir.
  - RUNNING :Çalışan yöngüder robot sayısını gösterir.
  - PAUSED :Geçici olarak durdurulmuş yöngüder robot sayısını gösterir.
  - STOPPED :Tamamı ile durdurulmuş yöngüder robot sayısını gösterir.

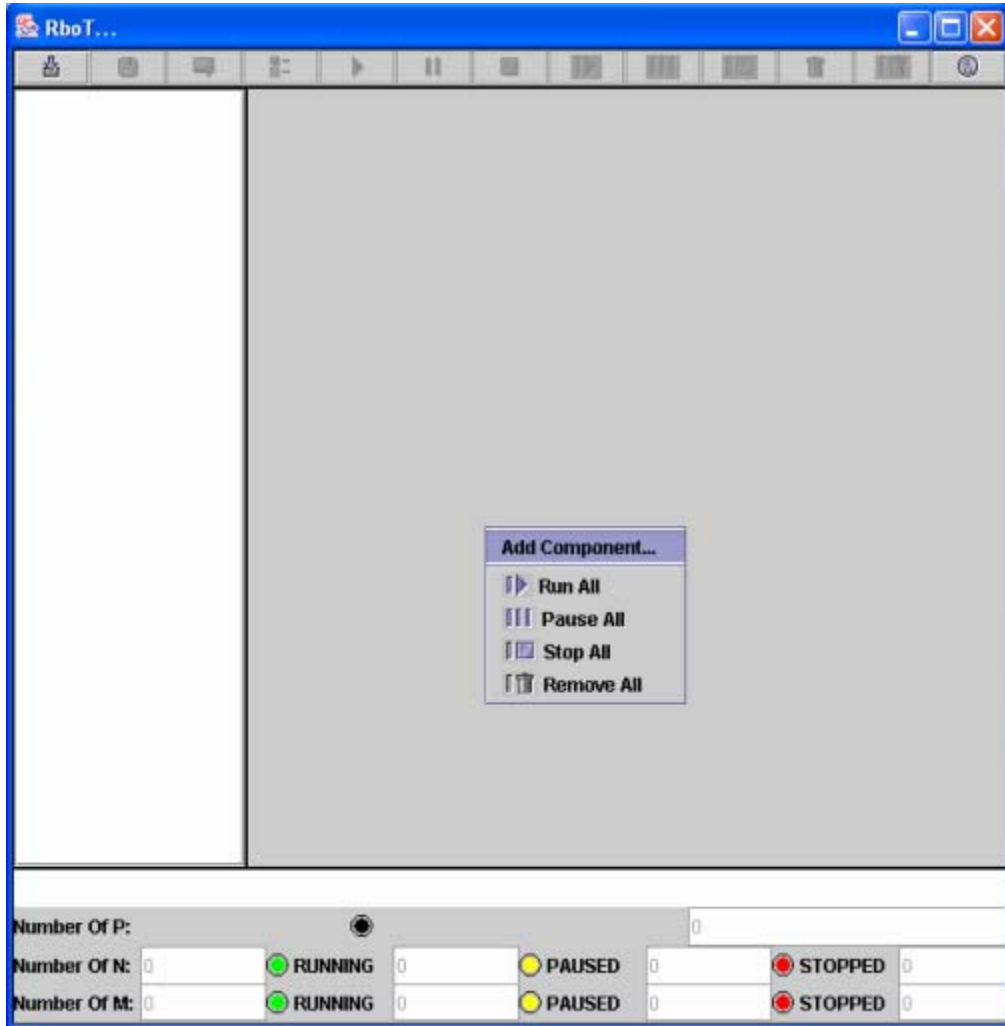
- *Number Of M*: Benzetim ortamındaki taşıyıcı robot sayısını gösterir.
  - RUNNING :Çalışan taşıyıcı robot sayısını gösterir.
  - PAUSED :Geçici olarak durdurulmuş taşıyıcı robot sayısını gösterir.
  - STOPPED :Tamamı ile durdurulmuş taşıyıcı robot sayısını gösterir.

Kullanıcı BP'de herhangi bir modifikasyon yapamaz. Kullanıcıya bildirilmesi gereken tüm olaylar,bir bileşenin benzetim ortamına eklenmesi gibi, IP'nin metin alanında gösterilir.

## B.2 Benzetim Ortamı (WSP)

Benzetim ortamı(*WSP*) kullanıcın bileşen eklediği ve bu bileşenlerin hareketlerini gerçekleştirdikleri ortamdır. *RboT* başlatıldığında *WSP* boştur.

*WSP* üzerinde eğer kullanıcı farenin sağ tuşuna tıklarsa Şekil 7'de gösterildiği gibi bir açılır menü kullanıcıya gösterilecektir.

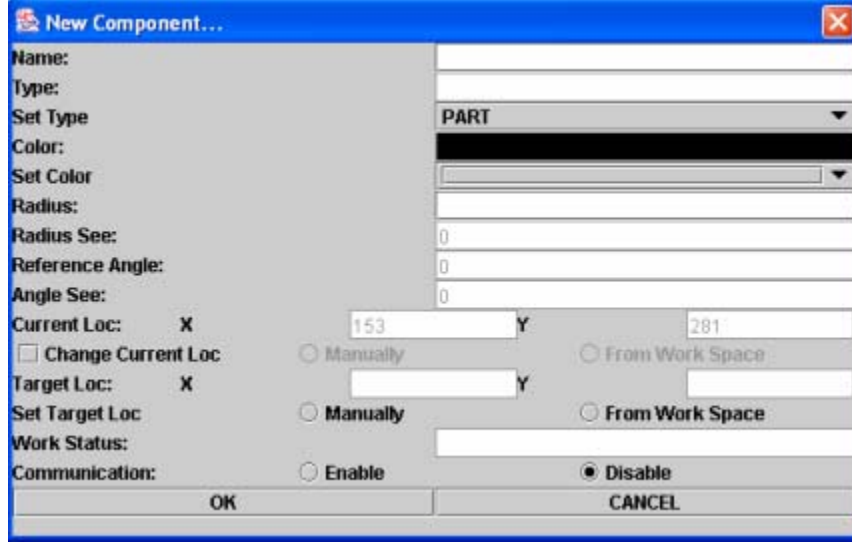


Şekil 7: *WSP* de farenin sağ tuşuna tıklanıldığında gösterilen açılır menü

*WSP* üzerinde yapılabilecek tüm olaylar açılır menüde gösterilen seçeneklerle sınırlıdır.

### B.2.1 Add Component (Bileşen Ekle)

Bu seçenek kullanıcının benzetim ortamına yeni bileşen eklemesini sağlar. Kullanıcı bu seçeneği seçerse Şekil 8’teki “New Component -Yeni Bileşen” ara yüzü kullanıcıya gösterilir.



Şekil 8: New Component (Yeni Bileşen) Ara Yüzü

“New Component Dialog Box (NCDB)- Yeni Bileşen Ara Yüzü” ’nde bir bileşenin sahip olacağı tüm özellikler listelenmiştir.

#### B.2.1.1 New Component Dialog Box (NCDB-Yeni Bileşen Ara Yüzü)

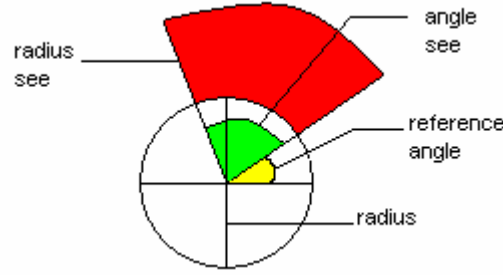
##### NCDB’nin Kısımları:

1. Name : Bileşenin ismi. Bileşenin ismi kullanıcı tarafından girilir.
2. Type : Bileşenin tipi. Kullanıcı bileşenin tipini, *parça*, *yöngüder robot* veya *taşıyıcı robot*, “Set Type-(Tipi Seç)” ilgili menüden seçebilir.
3. Color: Bileşenin rengi. Varsayılan renk siyahtır. Kullanıcı bileşenin rengini “Set Color-(Renk Seç)” ilgili menüden seçebilir.
4. Radius: Bileşenin yarıçapı. Unutulmamalıdır ki bileşenler daireseldir ve WSP’de daire şeklinde gösterilmişlerdir. Bileşenin yarıçapı kullanıcı tarafından girilir.
5. Radius See: Bileşenin görebileceği maximum radyal uzaklık. Değer kullanıcı tarafından girilir. *Radius see* ’nin 0 olarak belirtilmesi, bileşenin tüm benzetim ortamını görmesi olarak kabul edilmiştir.
6. Angle See: Bileşenin görebileceği maximum açısal uzaklık. Değer kullanıcı tarafından girilir. *Angle see* ’nin 0 olarak belirtilmesi, bileşenin 360°’lik alanı tarayabilmesi olarak kabul edilmiştir.

7. Reference Angle: Bileşenin etrafını görmeye başladığı açıdır. Değer kullanıcı tarafından girilir.

“Radius” ve “radius see”nin birimleri pikseldir. Buna karşılık “angle see” ve “reference angle”nın birimleri derecedir. Burada belirtilen tüm değerler ya sıfır ya da pozitif tamsayı olmalıdır. Açı değerleri pozitif-x eksenini referans alınarak ölçülmektedir.

Burada belirtilen değerler Şekil 9 'de gösterilmiştir.



Şekil 9: Robotun görsel alan özellikleri. Robot sadece kırmızı renkli alanı görebilir.

8. Current Loc: Bileşenin merkezinin WSP'deki *anlık konumu*. X ve Y değerlerinin birimleri pikseldir. Varsayılan *anlık konum* kullanıcının farenin sağ tuşuna tıkladığı noktadır. Bileşenin *anlık konumu*'nu değiştirmek için kullanıcının “Change Current Location - Anlık Konumu Değiştir” düğmesini işaretlemesi gerekmektedir.

Kullanıcı bileşenin anlık konumunu iki şekilde belirtebilir:

- Manually: Bu durumda kullanıcının kendisi bileşenin anlık konumunu X ve Y ile belirtilen metin kısmına girer.
- From Work Space: Bu durumda NCDB kaybolur ve kullanıcı WSP'de herhangi bir noktaya fare ile tıklar. Kullanıcının bu seçeneği seçmesi durumunda, kullanıcıya sunulan tüm olanaklar geçici olarak deaktive edilir. Başka bir deyişle, kullanıcı WSP üzerinde herhangi bir noktaya tıklamadan başka bir olay gerçekleştiremez. WSP'de bir noktaya tıkladıktan sonra NCDB tekrar görünür ve bileşenin *anlık konumu* kullanıcının WSP'de tıkladığı nokta olarak alınır.

9. Target Loc: Bileşenin WSP'deki *hedef noktası*. X ve Y değerlerinin birimleri pikseldir. Kullanıcı “Set Target Loc – Hedef Noktayı Seç” tuş grubunu kullanarak bileşenin hedef noktasını belirtebilir.

Taşıyıcı robotların kendi hedeflerini kendilerinin bulacağı farz edilmiştir. Bundan dolayı kullanıcının bileşenin tipini “MOVER - TAŞIYICI” olarak seçmesi durumunda, “Set Target Loc - Hedef Noktayı Seç” tuş grubunu kullanması mümkün değildir.

Kullanıcı bileşenin hedef noktasını iki şekilde belirtebilir:

- **Manually:** Bu durumda kullanıcının kendisi bileşenin hedef noktasını X ve Y ile belirtilen metin kısmına girer.
- **From Work Space:** Bu durumda NCDB kaybolur ve kullanıcı WSP'de herhangi bir noktaya fare ile tıklar. Kullanıcının bu seçeneği seçmesi durumunda, kullanıcıya sunulan tüm olaylar geçici olarak deaktive edilir. Başka bir deyişle, kullanıcı WSP üzerinde herhangi bir noktaya tıklamadan başka bir olay gerçekleştiremez. WSP'de bir noktaya tıkladıktan sonra NCDB tekrar görünür ve bileşenin *hedef noktası* kullanıcının WSP'de tıkladığı nokta olarak alınır.

10. **Work Status:** Bu kısım bileşenin şu anki çalışma durumunu gösterir. Anlık çalışma durumu iki değer alabilir: PAUSED(durdurulmuş) veya STATIONARY(sabit).

*Parçalar*'ın varsayılan anlık çalışma durumu STATIONARY, *yöngüder* ve *taşıyıcı* robotların varsayılan anlık çalışma durumu ise PAUSED olarak alınmıştır.

11. **Communication(iletişim):** Bir bileşen diğer bileşenlerle iletişimde bulunabilir. Bileşenin diğer bileşenlerle iletişimde bulunması, bileşenin bir sonraki hareket noktasını hesaplamada kullanılan matematiksel formüllerinde kendini göstermektedir.

Kullanıcı bileşenin iletişim olanağını aktive(enable) veya deaktive(disable) edebilir. Bileşenin varsayılan iletişim durumu deaktivedir.

12. **OK:** Kullanıcının bu tuşa basması bileşenin tüm özelliklerinin kullanıcı tarafından belirtildiği anlamını taşır. *OK* tuşuna basıldıktan sonra *RboT* bileşenin özelliklerini kontrol eder. Eğer herhangi bir yanlışlık varsa bu kullanıcıya bildirilir.

Hata mesajları *radius*, *radius see*, *angle see*, *reference angle*, ve *target loc* kısımlarına tamsayı olmayan değerlerin girilmesi, bileşenin isminin veya tipinin belirtilmemesi sonucu meydana gelebilir.

Eğer *RboT* herhangi bir hata bulmazsa *NCDB* kapatılır ve yeni bileşen WSP'ye eklenir. Ayrıca liste ve bilgi panelleri de kendi bilgilerini güncellerler.

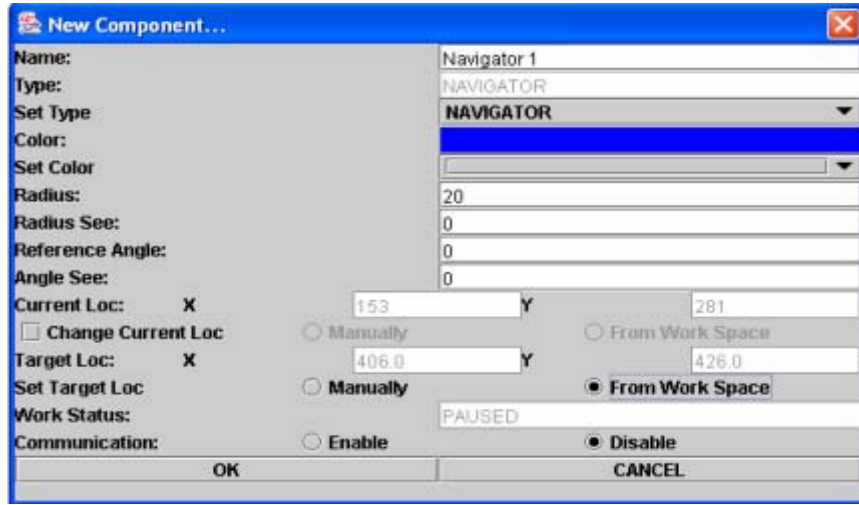
13. **CANCEL:** Kullanıcının bu tuşa basması kullanıcının yeni bileşen eklemekten vazgeçtiğini belirtir. Bu durumda *NCDB* kapatılır ve *RboT* benzetime, yeni bileşen eklemekten, kaldığı yerden devam eder.

*NCDB*'nin arayüz penceresindeki X işareti ile kapatılması da *CANCEL* tuşuna basılması ile aynı anlamı taşır.

Tek bir anda sadece bir tane *NCDB* aktif olabilir.

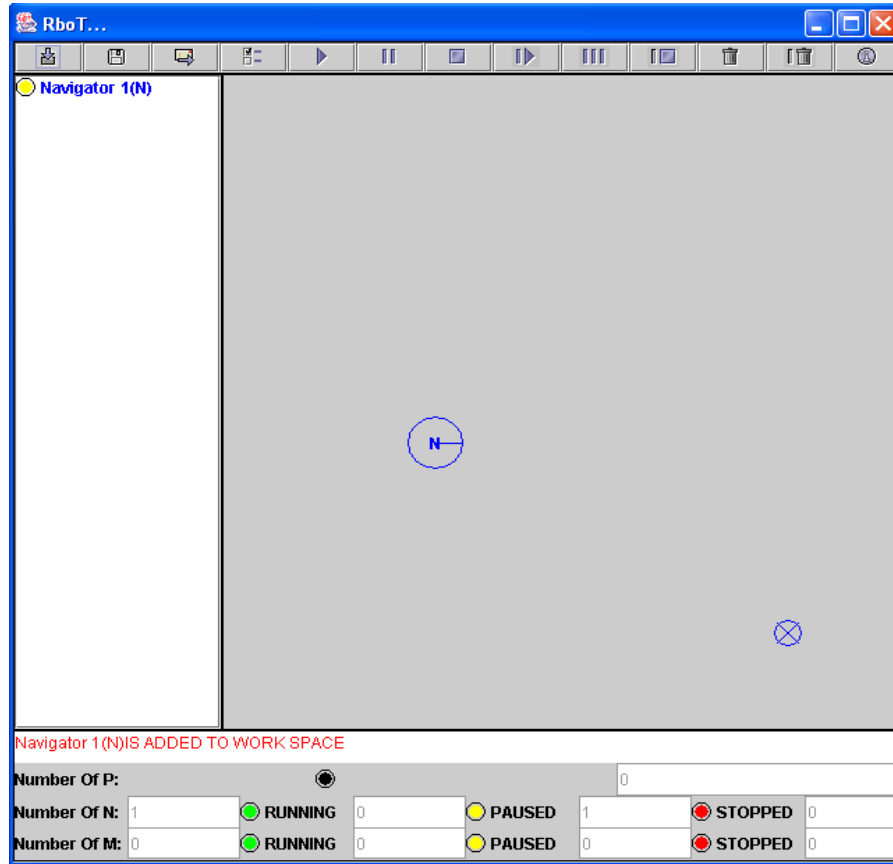
Örnek:

Kullanıcın Şekil 10’da gösterilen bilgileri NCDB’ye girdiğini farzedelim.



Şekil 10: Kullanıcın yeni bileşen için girmiş olduğu değerler

Eğer kullanıcı Şekil 10’da gösterilen NCDB’de *OK* tuşuna basarsa *RboT*’un son hali Şekil 11’deki gibi olur.



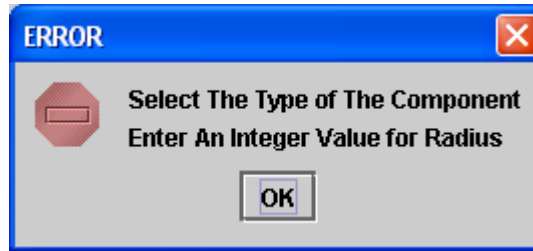
Şekil 11: “Navigator 1- Yöngüder 1” robotunun ortama eklenmesinden sonra *RboT*’un durumu. Dikkat edilirse yeni bileşenin eklenişi hem *bilgi* hem de *liste panelleri* tarafından algılanmıştır. Ayrıca *kontrol paneli* de aktif hale geçmiştir.



Şimdi de kullanıcının NCDB'ye Şekil 12'deki gibi yanlış ve eksik bilgiler girdiğini farz edelim.

Şekil 12: Bileşenin özellikleri. Kullanıcı bileşenin tipini belirtmemiş ve *radius* kısmına da tamsayı yerine harf dizisi girmiştir.

Eğer kullanıcı Şekil 12'de gösterilen NCDB'de *OK* tuşuna basarsa, Şekil 13'de gösterilen hata mesajı ile karşılaşacaktır.

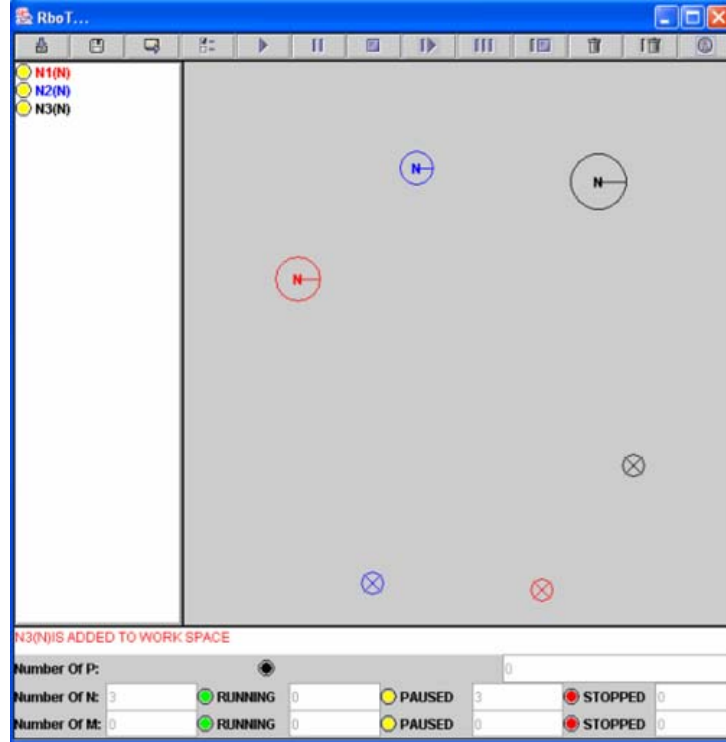


Şekil 13: Oluşturulan hata mesajı

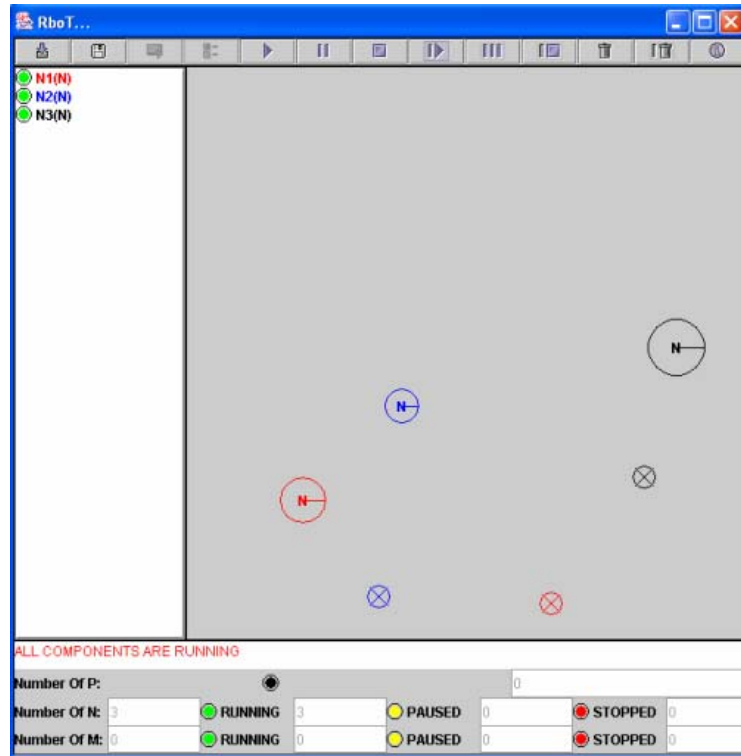
### B.2.2 Run All (Tümünü Çalıştır)

Bu komut tamamı ile durdurulmamış tüm robotların çalışır duruma geçmesini sağlar. Hali hazırda çalışır durumda olan robotlar bu komuttan etkilenmezler.

Benzetim ortamında üç adet yöngüder robot olduğunu farzedelim. Bu durum Şekil 14'de gösterilmiştir. Bu düzenleşim için kullanıcının *WSP*'deki açılır menüden *Run All* komutunu seçtiğini farz edelim. Bu komuttan sonra *RboT*'un durumu Şekil 15'de gösterilmiştir.



Şekil 14: Benzetim ortamına üç adet yöngüder robot eklendikten sonra *RboT*'un durumu. Şekilden de görüldüğü gibi hem *liste* hem de *bilgi panelleri* güncellenmiştir.



Şekil 15: WSP'deki tüm bileşenler çalışırken alınan anlık düzenleşim. Hem bilgi hem de liste panellerinde bileşenlere eşlik eden, bileşenin çalışma durumunu gösteren renklerin sarıdan yeşile geçtiğine/dönüştüğüne dikkat ediniz.

### B.2.3 Pause All (Tümünü Durdur)

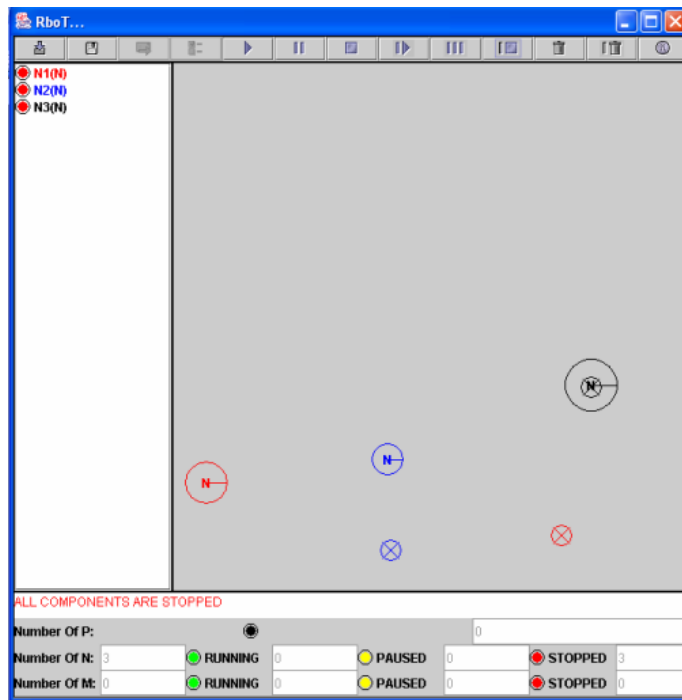
Bu komut *WSP*'de bulunan çalışır durumdaki tüm robotların durmasını sağlar. Daha önceden durdurulmuş olan robotlar bu komuttan etkilenmezler. Bileşenlerin çalışma durumlarında meydana gelen bu değişiklik yine hem bilgi hem de liste panelleri tarafından algılanır.

Benzetim ortamının Şekil 15'de gösterildiği gibi olduğunu farz edelim. Burada tüm robotlar çalışır durumda bulunmaktadır. Bu durumda kullanıcının *WSP*'deki açılır menüden *Pause All* komutunu seçtiğini farz edelim. *RboT*'un durumu Şekil 14'dekine benzer olacaktır. Sadece bileşenlerin konumları (bileşenler durduruldukları noktada kalacaklar) ve bilgi panelindeki mesaj "*N2 (N) IS PAUSED-N2 (N) DURDURULDU*" değişik olacaktır. Tüm bileşenler eş zamanlı çalıştığından dolayı bilgi panelinde sadece en son durdurulan robotun mesajı gösterilecektir.

### B.2.4 Stop All (Tümünü Deaktive Et)

Bu komut *WSP*'de bulunan tüm robotların tamamı ile çalışamaz hale gelmelerini sağlar. Daha önceden deaktive edilmiş robotlar bu komuttan etkilenmezler. Bileşenlerin çalışma durumlarında meydana gelen bu değişiklik *run all* ve *pause all*'da olduğu gibi hem bilgi hem de liste panelleri tarafından algılanır.

Deaktive edilmiş bileşenler ne çalıştırılabilirler ne de durdurulabilirler, sadece benzetim ortamından çıkarılabilirler. Benzetim ortamının Şekil 15'de gösterildiği gibi olduğunu farz edelim. Şekil 15'de tüm robotlar çalışır durumda bulunmaktadır. Bu durumda kullanıcının *WSP*'deki açılır menüden *Stop All* komutunu seçtiğini farz edelim. Elde edilen son durum Şekil 16'da gösterilmiştir.



Şekil 16: Kullanıcının "Stop All" komutunu çalıştırmısından sonra *RboT*'un durumu. Hem *bilgi* hem de *liste panellerinde* bileşenlere eşlik eden, bileşenin çalışma durumunu gösteren renklerin yeşilden kırmızıya geçtiğine/dönüştüğüne dikkat ediniz.

### B.3 Bileşenler (Components)

*RboT*'un yapı taşları bileşenlerdir. Tüm bileşenler *WSP*'de daire şeklinde gösterilmiştir. Gerçek hayatta bu bileşenler parçaları ve robotları temsil etmektedir.

*RboT* bileşenler üstünde bazı komutların uygulanmasına olanak sağlar. Bu bölümde bu komutlar ele alınacaktır.

Bir bileşenin üstünde farenin sağ tuşuna basılması bileşenin tipine ve çalışma durumuna bağlı olarak bir açılır menünün görünmesini sağlar. En genel pop-up menü seçenekleri :

- Run (Çalıştır)
- Pause (Durdur)
- Stop (Deaktive Et)
- Remove (Benzetim Ortamından Kaldır)
- Component Properties (Bileşen Özellikleri)

#### B.3.1 Run (Çalıştır)

Bu komut eğer bileşen durdurulmuş bir robot ise açılır menüde görülür. Komut robotun yeniden çalışmasını sağlar.

#### B.3.2 Pause (Durdur)

Bu komut eğer bileşen çalışan bir robot ise açılır menüde görülür. Komut robotun durmasını sağlar.

#### B.3.3 Stop (Deaktive Et)

Bu komut eğer bileşen bir robot ise açılır menüde görülür. Komut robotun deaktive olmasını sağlar.

#### B.3.4 Remove (Benzetim Ortamından Kaldır)

Bu komut tüm bileşenler için açılır menüde görülür. Komut bileşenin benzetim ortamından çıkarılmasını sağlar.

Bileşenin çalışma durumundaki her değişiklik benzetim ortamı, liste paneli ve bilgi paneli tarafından algılanır.

#### B.3.5 Component Properties (Bileşen Özellikleri)

Bu komut tüm bileşenler için açılır menüde görülür. Komut bileşenin özelliklerinin görüntüleyen “component info dialog box(CIDB) – bileşen bilgi ara yüzü”nün açılmasını sağlar.

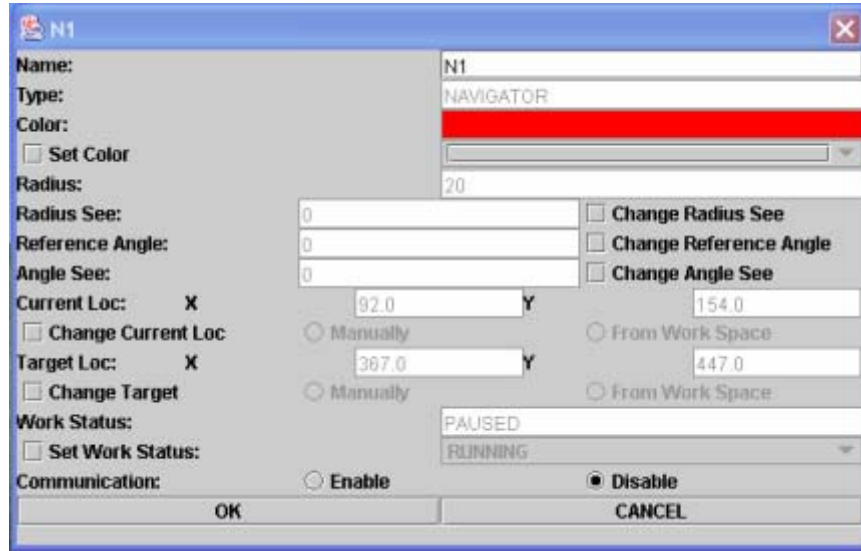
*WSP*'de seçilen bileşen liste panelinde magenta rengeyle belirtilmektedir.

Kullanıcı bileşenin anlık pozisyonunu ve hedef noktasını fareyi kullanarak değiştirebilir. Bunun için kullanıcının bileşenin kendisini veya hedef noktası için gösterilen şekli, farenin

sol tuşuyla tutması ve sürüklemesi gerekmektedir. Farenin sol tuşunun bırakıldığı nokta yeni anlık pozisyon veya hedef nokta olacaktır.

### B.3.5.1 Component Info Dialog Box (CIDB – Bileşen Bilgi Ara Yüzü)

Bileşenin tipine göre CIDB’ler biraz farklılar gösterebilirler. Kullanıcının Şekil 14’da gösterilen “N1” yöngüder robotu için ortaya çıkan açılır menüden “Component Properties...” seçeneğini seçtiğini farz edelim. Ortaya çıkan CIDB Şekil 17’de gösterilmiştir.



Şekil 17: Yöngüder robotu için ortaya çıkan “Bileşen Bilgi Ara Yüzü”. Bileşenin ismi CIDB’nin başlık kısmında görülmekte.

CIDB, NCDB ile aynı özellikleri taşımaktadır. Bir bileşenin iki özelliği hariç tüm özellikleri değiştirilebilir. Bileşenin değiştirilemeyen özellikleri:

Radius (yarıçap) & Type(Tip)

Gerçek hayattaki uygulamalar düşünüldüğünde bileşenlerin bu iki özelliğinin değiştirilememesi anlam taşımaktadır. Kullanıcının yanlışlıkla bileşenin özelliklerini değiştirmemesi için gerekli tüm yerlerde onay kutuları kullanılmıştır.

Kullanıcının bileşen özelliklerini değiştirdikten sonra *OK* tuşuna basması gerekmektedir. Eğer kullanıcı *CANCEL* tuşuna basarsa veya *CIDB*’yi ara yüzde bulunan X işaretine basarak kapatırsa, bileşen üstünde herhangi bir değişiklik yapılmaz.

*CIDB*’de bulunan “*Current Loc - anlık durum*” ve “*Target Loc – Hedef Nokta*” pozisyonlarını değiştirme prensibi *NCDB*’de açıklandığı gibidir.

Kullanıcın *OK* tuşuna basmasından sonra eğer bileşenle ilgili bir hata bulunursa, hata mesajları kullanıcıya *NCDB*’de olduğu gibi kullanıcıya gösterilir.





Bileşenin çalışma durumundaki her değişiklik benzetim ortamı, liste paneli ve bilgi paneli tarafından algılanır.

Tek bir anda sadece bir tane *CIDB* aktif olabilir.

#### B.4 Liste Paneli(LP)

Liste paneli benzetim ortamında bulunan tüm bileşenlerin liste halinde gösterildiği yerdir.

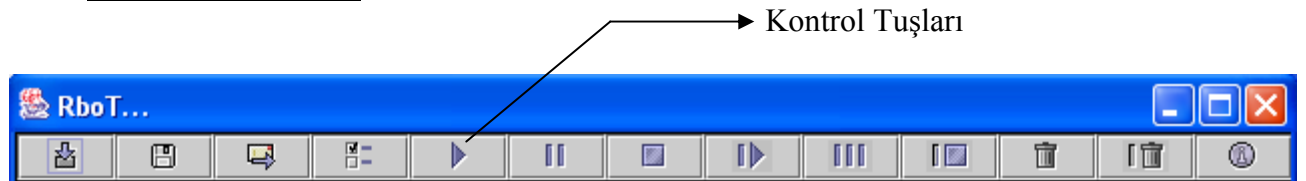
Bir bileşen liste panelinde şu özellikleri ile gösterilir:

- Bileşenin kendi rengiyle yazılmış ismi
- Bileşenin çalışma durumu
  - Parça (Sabit) 
  - Çalışan Robot 
  - Durdurulmuş Robot 
  - Deaktive Edilmiş Robot 

*Liste paneli* kullanıcıya bazı kolaylıklar sağlar:

1. Kullanıcının *liste panelinde* seçtiği bileşen, *WSP*'de bileşenin kendi rengiyle doldurulmuş daire şeklinde gösterilir.
2. *Liste paneli* kullanıcıya aynı anda, *CP* yardımıyla, birden fazla bileşenle iş yapabilmeye özelliği sunmaktadır. Bu özellik kullanıcıya aynı anda birden fazla bileşeni çalıştırma, durdurma, deaktive etme ve benzetim ortamından çıkarma olanağı verir. Bu özelliğin kullanılabilmesi için klavye üzerindeki "Ctrl" tuşunun basılı tutulması gerekmektedir.
3. Liste panelinde bileşenin isminin üzerinde fareyle çift tıklanması bileşene ait "*bileşen bilgi ara yüzü*"nün gösterilmesini sağlar.


#### B.5 Kontrol Paneli (CP)



Şekil 18 : Konrol Paneli

Benzetim ortamının daha kolay kullanılmasına olanak sağlayan kontrol paneli, benzetim ortamını kontrol eden tuşları barındırır.


#### CP'nin Kısımları:

 (Düzenleşim Yükle): Benzetim ortamına bileşenleri dosyadan yükleme seçeneğidir. Yüklenecek istenen dosya anlık düzenleşim veya kullanıcının hazırlamış olduğu düzenleşim dosyası olabilir.

Eğer kullanıcı düzenleşim dosyasını kendisi oluşturmak istiyorsa, dosyanın formatı şu şekilde olmalıdır:

- Bileşenin ismi
- Bileşenin tipi
  - Parçalar için 0
  - Yöngüder robotlar için 1
  - Taşıyıcı robotlar için 2
- Bileşenin yarıçapı
- Bileşenin anlık pozisyonununun x koordinatı
- Bileşenin anlık pozisyonununun y koordinatı
- Bileşenin hedef noktasının x koordinatı
- Bileşenin hedef noktasının y koordinatı
- Bileşene ait Radius see
- Bileşene ait Reference angle
- Bileşene ait Angle see
- Bileşenin bir önceki çalışma durumu, *-1 olmalıdır*
- Bileşenin anlık çalışma durumu ya *3(durdurulmuş)* ya da *5 (deaktive edilmiş) olmalıdır*
- Bileşenin iletişim durumu (*iletişim özelliği varsa 1, yoksa 0 olmalıdır*)

Burada belirtilen tüm değerler satırlar arasında boşluk bırakılmadan yazılmalıdır. Taşıyıcı robotlar hedef noktalarını kendileri buldukları için dosyaya yazılan hedef noktası önemli değildir. Eğer dosyanın yüklenmesi durumunda herhangi bir hata meydana gelirse, lütfen dosyanın içeriğini kontrol ediniz.


 (Düzenleşimi Kaydet): Benzetim ortamındaki bileşenlerin anlık durumlarını dosyaya kaydeder.


Kayıt edilen dosyanın ismi şu formattadır:

(Gün-Yazı ile) (Ay-Yazı ile) (Gün-Sayı ile) (Saat) (EEST) Yıl---belli bir zamandan şu ana kadar geçen milisaniye.

Örneğin:

Fri Jun 10 02\_04\_32 EEST 2005---1118358272890

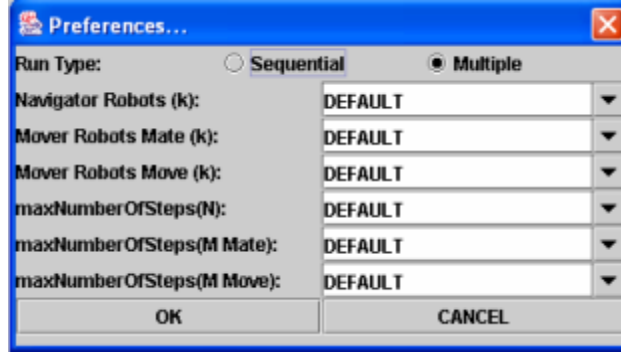
 (İletişimi aktive/deaktive et): Bileşenlerin iletişim özelliklerini aktive/deaktive eder. Butona bir sonraki basış, bileşenlerin iletişim özelliklerini ters çevirir.

 (Seçenekler): Seçenekler ara yüzünün gösterilmesini sağlar.

### **B.5.1 Preferences Dialog Box (PDB-Seçenekler Ara Yüzü)**

*RboT* bir bileşenin bir sonraki konumunu hesaplarırken bazı matematiksel formüller kullanır. Bundan dolayı *RboT* bu matematiksel formüllere ait bazı değişkenleri kullanıcının belirlemesine olanak sağlar. Bu değişkenler *PDB* tarafından kullanıcıya gösterilmekte ve yine

*PDB* aracılığı ile değişkenlerin değerleri değiştirilmektedir. **Şekil 19** örnek bir *PDB* göstermektedir.



Şekil 19: PDB-Seçenekler Ara Yüzü

### PDB'nin kısımları:

1. Run Type (Çalışma Şekli): *RboT* kullanıcıya bileşenleri çoklu(*Multiple*) veya ardışıl(*Sequential*) biçimde çalıştırma imkanı verir.

Bir bileşenin bir sonraki pozisyonunun hesaplanması tipiyle ve dolayısıyla hesaplamada kullanılan  $k$  değeri(lütfen üçüncü bölüme bakınız) ile son derece alakalıdır. Kullanıcı bu  $k$  değerini değiştirerek, bileşenin hareketine olan etkisini gözlemleyebilir.

2. Yöngüder Robots (k): Bu parametre üçüncü bölümde yöngüder robotlar için verilen formüldeki  $k$  değeridir.
3. Taşıyıcı Robots Mate (k): Bu parametre üçüncü bölümde taşıyıcı robotların parçayla eşleşmesi için verilen formüldeki  $k$  değeridir.
4. Taşıyıcı Robots Move (k): Bu parametre üçüncü bölümde taşıyıcı robotların parçayı hareket ettirmesi için verilen formüldeki  $k$  değeridir.

Bir bileşenin bir sonraki konumunun hesaplanması türevsel denklem sistemlerinin çözümünü gerektirmektedir. Çözüm sayısal olarak hesaplandığından dolayı, belli bir anda bileşen bir noktaya takılıp kalabilir. Bunun için *RboT* *maxNumberOfSteps* adlı bir değişken tanımlamıştır. Bu değişken belli bir nokta için yapılabilecek en fazla özyineleme sayısını belirtmektedir. Yani eğer bir noktada *maxNumberOfSteps* kadar özyineleme yapılmışsa *RboT* bunu, bileşenin o noktada takılı kaldığı olarak yorumlayacak ve yapılan özyinelemeyi durduracaktır. Bileşenin durdurulduğu bu nokta bileşenin hedef noktası olabileceği gibi herhangi başka bir nokta da olabilir. Özyineleme durdurulduktan sonra bileşen "*PAUSED-DURDURULMUŞ*" çalışma durumuna geçirilmektedir.

5. *maxNumberOfSteps* (N): Yöngüder robotlar için *maxNumberOfSteps*
6. *maxNumberOfSteps*(M Mate): Taşıyıcı robotların parçayla eşleşmeleri için *maxNumberOfSteps*












7. `maxNumberOfSteps` (M Move): Taşıyıcı robotların parçayı hareket ettirmeleri için `maxNumberOfSteps`

Kullanıcı `maxNumberOfSteps` değerini kendisi girebilir veya varsayılan(DEFAULT) değeri seçebilir. Kullanıcının bu alanlara sayı olmayan değerler yazması veya sıfır veya daha küçük sayılar hata sebebi olup, tesbit edilen hatalar kullanıcıya gösterilir.

`maxNumberOfSteps` için varsayılan değer 100'dür.  $k$  değerleri için varsayılan değerler ise bileşenlerin tipine ve iletişim özelliklerinin olup olmamasına göre değişiklik göstermektedir.

Tek bir anda sadece bir tane PDB aktif olabilir.

-  (Seçili Bileşenleri Çalıştır): Liste panelinde seçilmiş olan bileşenleri çalıştırır.
-  (Seçili Bileşenleri Durdur): Liste panelinde seçilmiş olan bileşenleri durdurur.
-  (Seçili Bileşenleri Deactive Et): Liste panelinde seçilmiş olan bileşenleri deactive eder.
-  (Tüm Bileşenleri Çalıştır): Benzetim ortamındaki tüm bileşenleri çalıştırır.
-  (Tüm Bileşenleri Durdur): Benzetim ortamındaki tüm bileşenleri durdurur.
-  (Tüm Bileşenleri Deactive Et): Benzetim ortamındaki tüm bileşenleri deactive eder.
-  (Seçili Bileşenleri Kaldır): Liste panelinde seçilmiş olan bileşenleri benzetim ortamından çıkarır.
-  (Tüm Bileşenleri Kaldır): Benzetim ortamındaki tüm bileşenleri benzetim ortamından çıkarır.
-  (Bilgi): *RboT* ile ilgili kısa bilgi gösterir.

Kontrol paneli ortamda ancak en az bir tane bileşen varken aktif haldedir. Aksi halde sadece “Düzenleşim Yükle” ve “Bilgi” tuşları aktif konumdadır. Kontrol paneli tarafından gerçekleştirilen tüm olaylar benzetim ortamı, liste paneli ve bilgi paneli tarafından algılanır.

## C Runge-Kutta Metodları

Nümerik analizde Runge-Kutta metodları, özellikle sıradan diferansiyel denklemlerin çözümünde, önemli bir yer teşkil etmektedir. Runge-Kutta metodları 1900'lü yıllarda C. Runge ve M.W. Kutta tarafından bulunmuştur [5].

### C.1 Klasik Dördüncü Derece Runge-Kutta Metodu

Runge-Kutta metodlarının en yaygın kullanılanı ve genelde *RK4* olarak adlandırılan yöntemdir [5]. Bu bölümde verilen tüm bilgiler [5,6]'dan alınmıştır. Bir ilk değer probleminin aşağıdaki gibi verildiğini farzedelim:

$$y' = f(t, y), \quad y(t_0) = y_0$$

Bu durumda RK4 kullanılarak bulunan  $y$ 'nin bir sonraki değeri:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Bu eşitlikte:

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\ k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\ k_4 &= f(t_n + h, y_n + hk_3) \end{aligned}$$

Eşitlikten de anlaşılacağı üzere ardışıl yaklaşımla  $y$  nin her bir sonraki değeri ( $y_{n+1}$ ), o anki değerinin ( $y_n$ ) adım ölçüsü ( $h$ ) ve tahmini eğimin çarpımı sonucuyla toplanmasıyla elde edilir.

$k_1$  başlangıç aralığının eğimini;  
 $k_2$  alınan aralığın orta noktasındaki eğimi  $k_1$  değerine bağlı olarak;  
 $k_3$  yine orta noktadaki eğimi ancak bu kez  $k_2$  değerine bağlı olarak;  
 $k_4$  aralık sonundaki eğimi  $k_3$  değerine bağlı olarak vermektedir.

Bu eğimlerin ortalaması alındığında orta noktadaki eğim değerlerine daha fazla ağırlık verilmektedir:

$$\text{Eğim} = \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

olarak bulunur. RK4 dördüncü dereceden bir metod olup toplam hata  $h^4$  mertebesindedir.

Dördüncü dereceden Runge-Kutta yaklaşımı bazı durumlarda çok fazla iterasyon gerektirebilir. Bu durumda kullanılan adım ölçüsünün yenilenmesi sonuca ulaşmayı önemli ölçüde kolaylaştırmaktadır. Adım ölçülerindeki bu değişiklikler dördüncü ve beşinci dereceden Runge-Kutta Metodlarının sonuçları arasındaki farklardan yararlanarak yapılır.

Beşinci dereceden Runge-Kutta formülünün genel yapısı aşağıda ifade edildiği şekildedir:

$$k_1 = h * f(x_n, y_n)$$

$$k_2 = h * f(x_n + a_2 h, y_n + b_{21} k_1)$$

$$k_6 = h * f(x_n + a_6 h, y_n + b_{61} k_1 + \dots + b_{65} k_5)$$

$$y_{n+1} = y_n + c_1 k_1 + c_2 k_2 + c_3 k_3 + c_4 k_4 + c_5 k_5 + c_6 k_6 + O(h^6)$$

Bileşik dördüncü dereceden Runge-Kutta formülü ise:

$$y_{n+1}^* = y_n + c^*_1 k_1 + c^*_2 k_2 + c^*_3 k_3 + c^*_4 k_4 + c^*_5 k_5 + c^*_6 k_6 + O(h^5)$$

şeklinde ve sonuç olarak tahmini hata;

$$\Delta \equiv y_{n+1} - y_{n+1}^* = \sum_{i=1}^6 (c_i - c_i^*) k_i$$

formülüyle bulunur.

Bileşik Runge-Kutta Metodu için Cash-Karp Parametreleri								
$i$	$a_i$	$b_{ij}$					$c_i$	$c_i^*$
1							$\frac{37}{378}$	$\frac{2825}{27648}$
2	$\frac{1}{5}$	$\frac{1}{5}$					0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				$\frac{250}{621}$	$\frac{18575}{48384}$
4	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			$\frac{125}{594}$	$\frac{13525}{55296}$
5	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		0	$\frac{277}{14336}$
6	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$
$j =$		1	2	3	4	5		

Eğer  $h_1$  adım büyüklüğü  $\Delta_1$  kadar hataya sebep oluyorsa,  $h_0$  olarak ifade edilen ve istenilen hata büyüklüğü  $\Delta_0$ 'a sebep olan adım ölçüsü aşağıda verilen şekilde bulunabilir.

$$h_0 = \begin{cases} Sh_1 \left| \frac{\Delta_0}{\Delta_1} \right|^{0.20} & \Delta_0 \geq \Delta_1 \\ Sh_1 \left| \frac{\Delta_0}{\Delta_1} \right|^{0.25} & \Delta_0 < \Delta_1 \end{cases}$$

Bu denklemde  $\Delta_0 = \varepsilon * h_1 * \frac{dy}{dx}$  ve  $\varepsilon = 10^{-6}$  gibi bir sayı olarak düşünülebilir. S ise güvenlik faktörü olarak ifade edilen birden küçük ve bire yakın bir sayıdır.

Bu hesaplamalar adım ölçüsünün nasıl yenilenebileceğini göstermektedir. Bu projede denklemlerin çözümü için yukarıda kısaca anlatılan Runge-Kutta metodundan faydalanılmıştır.