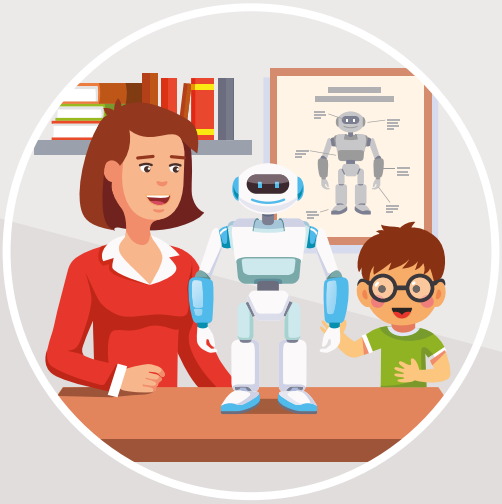


5. ve 6. Sınıf
Bilişim Teknolojileri ve Yazılım Dersi

KODLAMA KILAVUZU

Yenilik ve Eğitim Teknolojileri Genel Müdürlüğü

5. ve 6. Sınıf
Bilişim Teknolojileri ve Yazılım Dersi
Kodlama Kılavuzu

Yenilik ve Eğitim Teknolojileri Genel Müdürlüğü

5. ve 6. Sınıf Bilişim Teknolojileri ve Yazılım Dersi
Kodlama Kılavuzu

Yazar:
Erdal Delebe
Bilişim ve Teknoloji Öğretmeni

Kitap ve Kapak Tasarım:
Tanzer Özder

İnceleme Ekibi:
Ahmet Şamil Demircan
Erdem Uygun
Gökhan Gök
Kafiye Semiz
Zehra Sayın

2018

SUNUŞ

21. yüzyıl içerisinde gelişmiş ülkeler arasında üretim, buluş yapma ve teknolojik gelişme alanlarındaki yarış iyice hızlanmıştır. Bu yarış ortamı bütün ülkeleri bilime, mühendisliğe ve yenilikçi teknolojilere yatırım yapmaya yönlendirmektedir.

Bu sebeple, birçok gelişmiş ve gelişmekte olan ülke analitik düşünme, eleştirel düşünme, bilgi işlemsel düşünme gibi önemli becerilerin önemi kabul edip bu yolda eğitim sistemlerdin değişiklikler yapmaktadırlar. Günümüz eğitim sistemleri sorgulamaya, araştırmaya, üretime ve buluş yapmaya yönelik proje tabanlı disiplinler arası yaklaşımlara öncelik vermektedir. STEM (Fen, Teknoloji, Mühendislik, Matematik) eğitimi de bunlardan biri olarak dikkat çekicidir.

Scientix; Avrupa'da STEM (Fen, Teknoloji, Mühendislik, Matematik) eğitimini Scientix portalı aracılığıyla yaygınlaştırmayı amaçlayan bir projedir. STEM eğitiminin amacı, okullarda öğrencilerin sorgulama, araştırma, üretme ve buluş yapma becerilerini ve ilgilerini ortaya çıkarmaya yönelik öğrenci merkezli eğitimi yaygınlaştırmayı hedeflemektedir. Avrupa Okul Ağı - European Schoolnet tarafından yürütülen Scientix Projesinde Bakanlığımızı temsilen Yenilik ve Eğitim Teknolojileri Genel Müdürlüğümüz, 30 Avrupa ülkesi ile birlikte 2014 yılından itibaren Scientix Projesi Ulusal Destek Noktası olarak çalışmaktadır.

Bu bağlamda, kodlama eğitimi sorgulayan, üreten ve günümüz ihtiyaçları doğrultusunda beceriler edinmesi gereken bireylerin yetiştirilmesi için erken yaşlardan itibaren kodlama eğitim sistemleri içinde kendisine geniş bir yer bulmaktadır. Kodlama eğitimi sadece bilgisayar bilimleri ile sınırlı olmayıp, disiplinler arası etkileşimin sağlanması açısından önemli bir görevi bulunmaktadır. Böylece, çocukların erken yaşlardan itibaren bilgi işlemsel düşünme becerisi kazanarak, farklı alanlardaki problemlere çözüm üretebilme becerisi kazandırılmak istenmektedir.

Elinizdeki bu kılavuz, Yenilik ve Eğitim Teknolojileri Genel Müdürlüğümüz tarafından Bakanlığımıza bağlı okullarda yürütülmekte olan kodlama eğitimlerini geliştirilmesi desteklemeye yönelik öneri niteliğinde hazırlanmıştır. Kılavuz da öğrencilerin kodlamaya kullanarak problem çözme becerilerin geliştirmeye odaklanmaktadır. Algoritma oluşturma, blok tabanlı ortamlar ile kodlama ve programlanabilir elektronik devre kartlarının kullanımı gibi yükselen seviyede örnekler içermektedir.

GİRİŞ

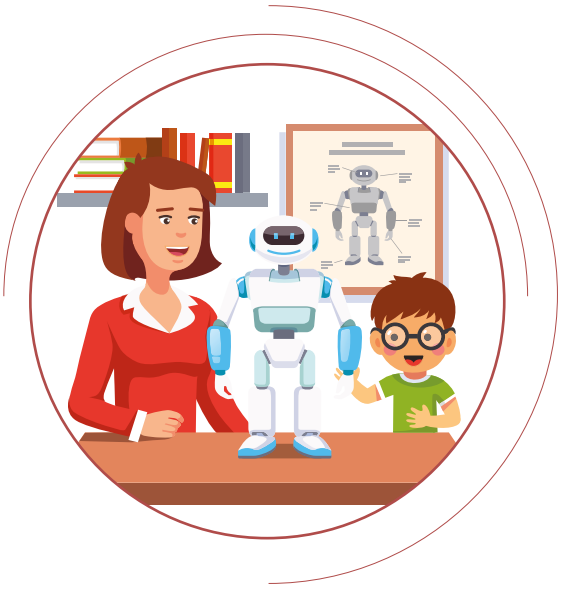
Hayatımızda ilk defa karşılaştığımız problemleri nadiren ilk seferinde çözebiliriz. Problemler daha karmaşık hale geldikçe daha fazla deneme ve daha fazla hata yaparız. Programlama; çocuklara problem çözmenin bir hedef değil bir süreç olduğunu öğretir. Bu, gelecekte mesleğine taşıyabileceği ya da hayatın zorluklarına direnç gösterebileceği bir yaşam becerisidir.

Bilgisayar programcılarının öğrendiği değerli bir süreç de hata ayıklamadır. Yaptığınız şeyleri öğrenmek ve sorunların nereden kaynaklanabileceğini bulmak, program yazma sürecinin bir parçasıdır. Çocuklara erken dönemde nasıl hata ayıklanacağının öğretilmesi, hem bilgisayar programcılığının hem de yaşamın karmaşık zorluklarına onları hazırlar. Kodlar üzerinde hata ayıklamayı öğrenmek programcıların ilk denemesinde veya 30. denemesinde başarılı olmayı öğretmez. Önemli olan hata ayıklama sürecini her tekrarladığımızda hedefe yaklaşacağımızdır. Hatalardan korkmamayı öğretir. Bu sayede karşılaştığı problemlere çözüm üretebilen, hata yapmaktan korkmayan, kendi yeteneklerini keşfedebilen bireyler yetiştirebiliriz.

Günümüz ve geleceğin global dili olan programlama dillerini öğrenmek, yeni projeler geliştirmek, çocuklarımızın hayallerini gerçekleştirebilmesine imkan sağlamak ve ülkemizin kalkınmasına fırsat tanımak adına Kodlama Kılavuzu Milli Eğitim Bakanlığı tarafından tüm öğrenci ve öğretmenlerimizin kullanımına sunulmuştur.

YEĞİTEK tarafından hazırlanan bu kılavuz, Bilişim teknolojileri ve yazılım dersi müfredatının güncellenen içeriğine yönelik destekleyici eğitim materyali ve konu anlatımı içermektedir. 5. ve 6. Sınıf müfredatı göz önüne alınarak hazırlanan bu kılavuzda; öğrencilerimizin problem çözme becerilerinin ve programlama yeteneklerinin geliştirilmesi amaçlanmaktadır. “Programlamaya nereden/nasıl başlarım” sorularına cevap olması açısından hazırlanan kılavuzun tüm öğrenci ve öğretmenlerimize yardımcı olması dileklerimizle.

BÖLÜM



PROBLEM ÇÖZME = KODLAMA

BÖLÜM 1: PROBLEM ÇÖZME = KODLAMA

Davranış Araştırma ve Terapisi, problem çözme becerilerinden yoksun çocukların depresyon ve intihar eğilimi riskinin yüksek olabileceğini ortaya koymuştur. Ayrıca araştırmacılar, çocuğun problem çözme yeteneğini geliştirmenin zihinsel sağlığı da geliştirebileceğini ifade etmektedir.

Çocuklar her gün akademik zorluklardan spor alanındaki sorunlara kadar çeşitli sorunlarla karşılaşıyorlar. Problem çözme becerisine sahip olmayan çocuklar, bir problemle karşılaştıklarında aksiyona girmekten çekinebilmektedirler. Problemi çözüme kavuşturmak yerine, meseleden kaçınmak için enerjilerini ve zamanlarını harcayabilirler. Bu yüzden birçok çocuk, okulda geride kalabilir ya da arkadaşlıklarını sürdürmek için çırpınabilir.

Günlük hayatta olduğu gibi programlama derslerinin de başarılı bir şekilde sürdürülebilmesi için, öğrencilerimizin sahip olması gereken özelliklerin başında ***"problem çözme yeteneği"*** gelir. Bu beceri, günlük hayatın yanısıra mesleki ve eğitimsel açıdan en önemli bilişsel etkinlik olarak görülmektedir. Geleneksel programlama öğretiminde, problem çözme üzerine pek durulmadığı görülmüştür.

Programlamayı öğrenmek birçok öğrenci için zor bir süreçtir. Soyut kavramları anlama, genelleme yapma, öğrendiklerini sonraki konulara aktarabilme ve eleştirel düşünme gibi bir hiyerarşi gerektirir. Programlamanın öğretilmesinde öğretmenin doğru materyallere sahip olması ve öğrencinin de doğru çalışma yöntemini takip etmesi gerekmektedir.

Programlama; pratik ve yoğun bir çalışma yaklaşımı gerektirir. Mevcut durumlar için öğrenilen bilgilerin, test edilerek uygulama dönüştürülmesi gerekir. Çoğu öğrencimizin karşılaştığı temel zorluk; bir programlama dilinde kod yazmak değil, bir problem için çözüm tasarlamak ve bunu formüle etmektir.

1.1: Problemi Tanımlayın

Günlük hayatta problem ifadesini genellikle can sıkıcı durumlarda kullanırız. İstenmeyen bir durumla karşılaştığımızda bunun hayatımızı tehdit eden bir hale bürünmeden çözülmesi gerektiği kanaatine vararak çözüm aramaya başlarız. Programlama dünyasında ise problem ifadesi; çözülmesi, geliştirilmesi veya tamamlanması gereken durumları ifade etmektedir. Herhangi bir konuda olmasını istediğimiz durum ile mevcut durum arasındaki her türlü eksiklik, problem olarak görülür ve bunları çözmek bir programcı/kod yazan için en önemli görevdir.

Her gün farkında olmadan birçok problemle karşılaşır ve bunları çözeriz. Bir öğrenme ortamında genellikle ihtiyacınız olan bilgilerin çoğu size verilir: sorunun açık bir şekilde ifade edilmesi, gerekli ihtiyaçlar ve elde edilecek sonuçlar. Örneğin matematik dersinde; eni 4 metre boyu ise 5 metre olan bir dikdörtgen bahçe için kaç metrelik bir çit teli kullanılacağı sorulsun. Verilenler bahçenin eni ve boyu istenilen ise bu bilgiyi kullanarak dikdörtgenin çevre hesabının yapılmasıdır. Elde edeceğimiz sonuç ise 18 metredir. Sınırları belli bir probleme çözüm bulmak oldukça kolaydır.

Gerçek hayatta ise işler bu kadar kolay yürümez. Tatildeyken saksıdaki çiçekleri sulamak için nasıl bir yol izleyebiliriz? Bir otomatik sulama sistemi tasarlanabilir mi? Çiçekler komşuya mı bırakılır? Tatilden döndüğünüzde kurumuş çiçeklerle karşılaşmamak için bir çözüm bulmak gerekecektir. Karşılaştığımız problemi tanımlı hale getirmemiz, ardından bu problemi çözebilmek için hangi bilgilere ihtiyacımız olduğuna karar vermemiz ve sonuçların ne olacağını netleştirmemiz gerekir.

1.2. Problem Çözme Öğretimi

Karşılaştığımız her problem için sabit bir reçete olsa hayat belki de çok daha kolay olurdu. İnsan doğası gereği her yeni problem için yeni çözümler aramaya çalışmak zorundadır. Hayatta kalma dürtüsü, bizleri, problemlerin çözümünde önce elimizdeki mevcut bilgiyi ve tecrübeyi kullanmaya iter. Mevcut bilgi ve tecrübemiz çözümü bulmak için yeterli gelmezse ne yapabiliriz?

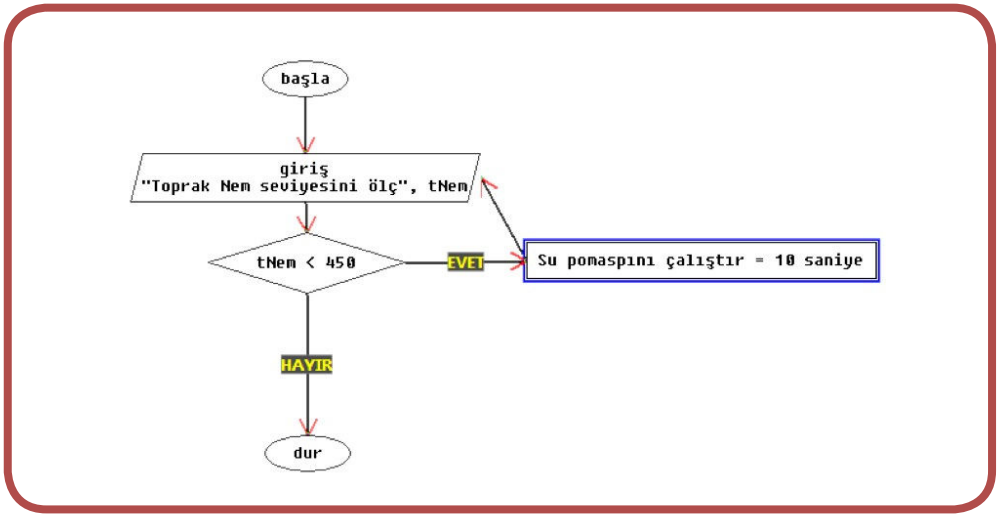
Problem çözme konusunda çocuklarımızın, ezberlenmiş kurallar yerine sistematik bir şekilde çözüm için çalışmaları gerekir. Öğretmenlerimizin temel görevi öğrencilerimizin bir problemle karşılaştığında takip edebileceği stratejileri kullanabilmeyi öğretmektir. Şimdi bu stratejileri inceleyelim.

1.3. Problem Çözme Stratejileri

Bir problemi anladıktan ve analiz ettikten sonra, bir algoritma ile çözüm bulmalısınız. Algoritma; belirli bir sürede, belirli bir veri ile problemi çözmek için takip edilen adımlardır. Genellikle algoritmalar basmakalıp izlenmesi gereken adımlar gibi düşünülür. Sıralama, ortalama hesaplama, faktöriyel hesaplama algoritması gibi... Sabit algoritmaları takip eder ve sonuca ulaşırız. Bilgisayar programlamanın problem çözme aşamasında, her probleme uygun algoritmaları tasarlamalısınız. Daha açık ifade etmek gerekirse problem çözme stratejilerinin farkında olarak bunları programlama problemlerine adım adım uygulayabilmeniz gerekir. Örnek olarak topraktaki nem seviyesine göre bir saksıdaki çiçeği sulamak istediğimizi varsayalım. Aşağıdaki örnekte problem ve problemi çözecek algoritma gösterilmektedir.

Problem: Toprak nem seviyesine göre saksının sulanması
Algoritma:
İşlem1: Başla
İşlem2: Toprağın nem seviyesini ölç
İşlem3: Eğer nem seviyesi 450 değerinin altında ise su pompasını 10 saniye boyunca çalıştır işlem2'ye git.
İşlem4: Bitir.

Şekil 1'de algoritmanın akış şeması gösterilmektedir.



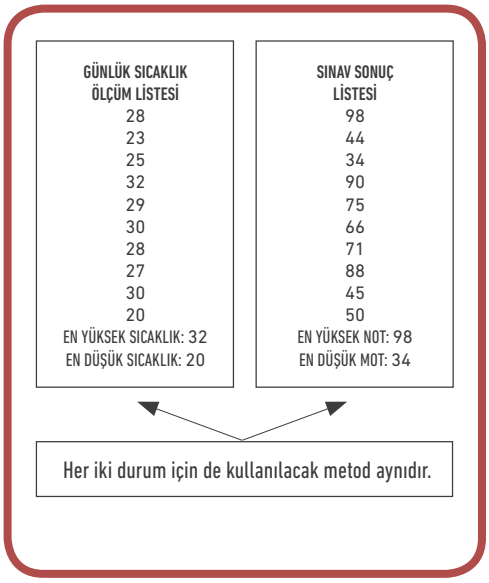
Şekil 1: Saksının nem seviyesini ölçerek toprağı sulayan sistemin akış şeması

1.3.1: Soru Sorun

Sözlü olarak bir emir aldığınızda, yapılması gereken işi tam olarak anlayana kadar; Ne zaman? Nerede? ve Nasıl? sorularını sorarız. Programlama üzerinde çalışırken de önceliğimiz, problemi ve mevcut durumu anlaya kadar sorular sormaktır:

· Çalışmak için neye ihtiyacım var, yani elimdeki bilgiler neler?
· Mevcut bilgiler, daha önceden karşılaştığım bilgilere benziyor mu?
· Elimdeki bilgiler üzerinde nasıl bir çalışma yapmalıyım?
· Sonuç olarak istenen nedir?
· Hangi hatalar ortaya çıkabilir?

Bir problemin anlaşılıp anlaşılmadığını test etmek için öğrencinin şu iki soruyu cevaplaması önemlidir:
Neler verilmiştir? İstenen nedir?



Şekil 2: Tanıdık Bilgi Var Mı?

1.3.2: Tanıdık Bilgi Var mı?

Tekerleği yeniden icat etmeye gerek yok. Probleminiz için mevcut bir çözüm varsa, onu kullanın. Eğer daha önceden benzer bir problemi çözmüş iseniz aynı çözümü tekrarlayın. İnsan bünyesi, benzer durumları tanımakta iyidir. Bakkala süt almak için nasıl gideceğimizi, yumurta almak için nasıl gideceğimizi, ekmek almak için nasıl gideceğimizi ayrı ayrı öğrenmemiz gerekmiyor. Bakkala nasıl gideceğimizi biliyoruz, sadece aldığımız ürün değişiyor.

Programlamada da belirli problemler farklı biçimlerde tekrar tekrar karşımıza çıkar. İyi bir programcı daha önce çözmüş olduğu problemi hemen tanır ve çözüme odaklanır. Örneğin günlük hava sıcaklığı ölçümünü yapan bir cihazı programlarken günlük en yüksek sıcaklık ile en düşük sıcaklığı bulmak, çözülmesi gereken bir görevdir. Benzer şekilde bir sınıftaki en yüksek not alan öğrenci ile en düşük not alan öğrenciyi bulmak da çözülmesi gereken bir görevdir, problemdir. Her iki problemi benzer metotlar kullanarak çözebiliriz.

1.3.3: Çıkarım Yapın

Problem çözümüyle belirli bir süre uğraştıktan sonra, genellikle bir problem daha önce gördüğünüz bir problemi hatırlatır. Diğer problemi nasıl çözdüğünüzü hatırlarsanız, daha kolay çözebilirsiniz. Başka bir ifadeyle iki problem arasında benzeşim kurabiliriz.

Bir sorunu çözmek için bir algoritma bulmaya çalışırken, kendinizi bilgisayar odaklı çözümlerle sınırlandırmayın. Soruna daha geniş bir açıdan bakın. Eğer yaptığınız çıkarımlar mükemmel değilse bunun için endişelenmeye gerek yok, elinizdeki sorunu önceden çözdüğünüz bir probleme benzetmeye çalışmak çözüm için iyi bir başlangıç noktasıdır.

1.3.4: Araç-Amaç Analizi

Başlangıç ve bitiş durumları verilen bir problemin aşamaları arasında birinden diğerine nasıl geçileceği başlı başına bir problemdir. Ankara'dan yola çıkıp (başlangıç noktası), İstanbul'a ulaşmak istiyorsunuz (bitiş noktası). Örneğin güzergah olarak Ankara, Bolu, Sakarya, Kocaeli, İstanbul arasında bir yolculuk planlıyorsunuz. Problemimiz bir noktadan diğerine nasıl geçeceğimizdir. Bunun için birçok seçeneğiniz vardır; yürüyerek, bisiklet kullanarak, otomobil, hızlı tren, helikopter veya uçak. Eğer çok aceleniz varsa muhtemelen uçağı seçeceksiniz.

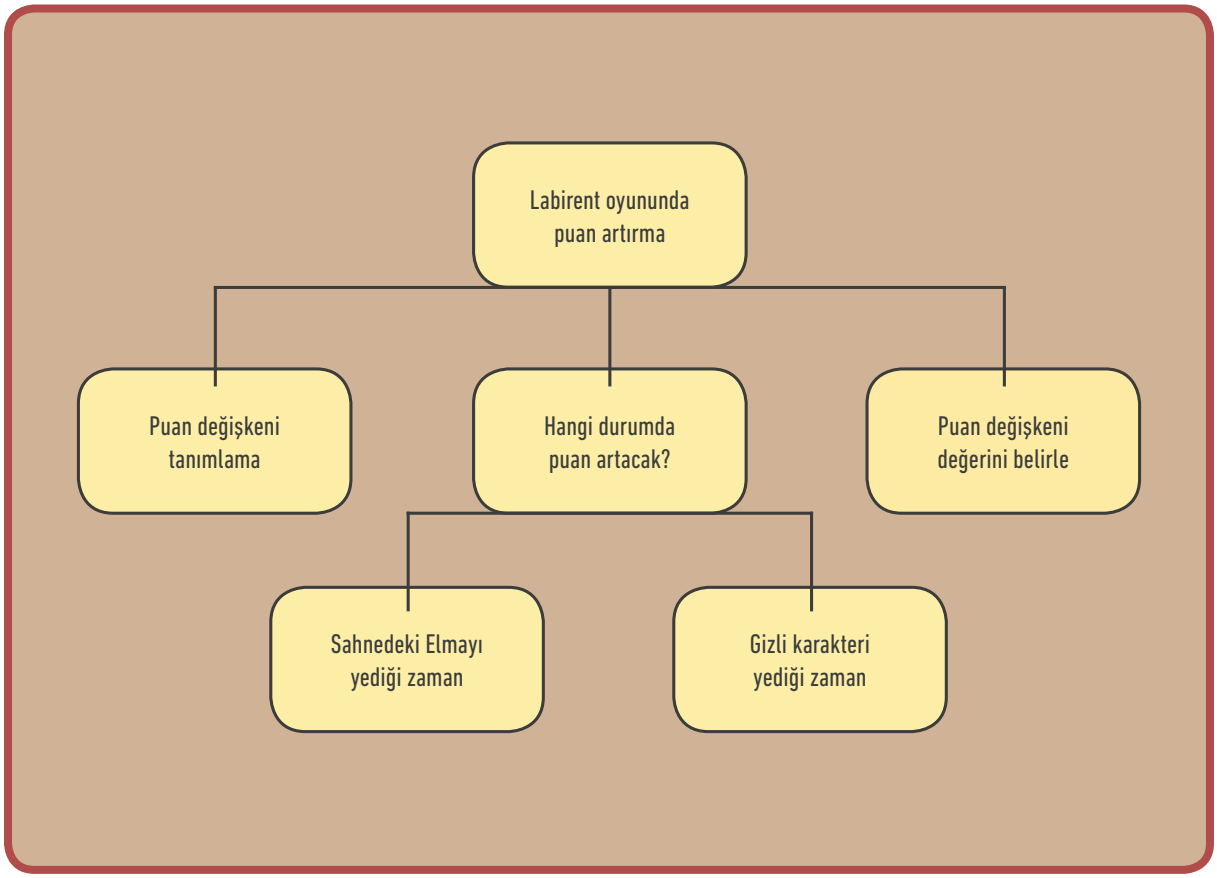
Yapılacak işin detaylarına göre seçenekleri daraltmak durumundasınız. Bu sayede genel hedefe ulaşmak için daha ulaşılabilir ara hedefler oluşturmayı sağlarsınız.

Yolculuk için hava koşullarından dolayı Ankara'daki tüm uçuşların iptal edildiğini ancak hızlı tren ile Eskişehir üzerinden yola çıkılabileceği bilgisini aldığımızda tüm rota ve hareket şartlarımızı değiştirmemiz gerekebilir.

Araç-amaç analizi stratejisinin temel amacı, ulaşmak istediğiniz hedefi tanımlamak ve bu hedefe giderken ki araçlarınızı ve ara hedefleri analiz etmektir. Bu süreç kolaylıkla bilgisayar programlamada kullanılabilir. Girişte verilenleri ve çıkışta ne olacağını yazarak işe başlayabilirsiniz. Ardından bilgisayarın yapabileceklerini göz önüne alarak girdiyi, sonuçlara dönüştürebilecek eylemler dizisi belirlersiniz.

1.3.5: Problemi Küçük Parçalara Bölün

Genellikle büyük problemleri daha kolay idare edilebilen daha küçük birimlere ayırırız. Tüm evi bir anda temizleme fikri yorucu gelebilir ancak odaları birer birer temizlemek daha kolay ve daha az yorucudur. Aynı prensibi programlamada da kullanırız. Büyük bir problemi tek tek çözebileceğimiz küçük problemlere bölebiliriz.



Şekil 3: Problemi küçük parçalara ayırma

1.3.6: Parçadan Bütüne

Büyük bir sorunu çözmeye çalışmadan önce, mevcutta problemin küçük parçaları için herhangi bir çözüm olup olmadığını kontrol ediniz. Küçük problemlerin çözümlerini birleştirerek büyük problemin çözümünü de sağlamış oluruz. Bu strateji bir duvarın inşasında küçük tuğlaların birbirine harçla tutturularak duvarın yükselmesini sağlamak gibi bir çalışmadır.

Şekil 4: Parçadan bütüne yolculuk

1.3.7: Çözümleri Bir Araya Getirin

Bir problemin çözümü için gerekli adımların bir araya getirildiği yöntemdir. Örneğin bir sınıfın bir sınavdan aldığı notların ortalamasını hesaplamak istiyoruz. Bunun için hem toplama işlemi hem de sınıf mevcudunu sayma işlemini yerine getirmeliyiz. Bu çözümleri yerine getirirken sınıf listesini hem toplam değeri bulmak için sayacağız hem de sınıf mevcudunu öğrenmek için sayacağız. Bu iki çözümü birleştirirsek, bir öğrencinin notunu okuyun ve ardından onu mevcut toplama ekleyin ve bir sonraki değere geçmeden önce sınıf mevcudu sayımıza 1 ekleyin. Alt problemler de aynı adımlar tekrarlandığından bunları birbirinin ardından çalıştırmak yerine, bir araya getirin. Örnek problemin algoritması şu şekilde olabilir:

Problem: Notları girilen bir sınıfın mevcudunu ve not ortalamasını hesaplamak
Algoritma:
İşlem1: Başla
İşlem2: Not giriniz(N)
İşlem3: Toplam = Toplam+N, Sınıf Mevcudu = Sınıf Mevcudu+1
İşlem4: Ortalama = Toplam/Sınıf Mevcudu
İşlem5: Yeni not varsa işlem2'ye git
İşlem6: Sınıf mevcudunu ve ortalamayı göster
İşlem7: Bitir.

1.3.8: Başlangıç Korkusu

Yazarlar, nereden başlayacağım korkusuyla aklındaki birçok düşünceyi hayata geçiremeden unutmuştur. Programcılar da çözülmesi gereken büyük bir problemle karşılaştığında, bunaltıcı bir görüntüyle karşı kaşıya kalırlar. İnsanı en çok yoran nereden başlayacağına karar verememek daha da kötüsü kararsız kalmaktır.

Bir problemi çözmeye başlamak için en iyi yol kağıt kalemi elinize alıp, kendi sözlerinizle problemin ne olduğunu, ne anladığınızı tarif etmektir. Sorunu açıkladığınızda, tüm sorunu tek tek ele almaya çalışmak yerine alt bölümlerin her birine odaklanabilirsiniz. Bu işlem ana problemin daha net bir resmini verir. Bu sayede daha önceden çözmüş olduğunuz problemlerin çözümle ilgili parçalarını görmenizi sağlar. Konu hakkında bilgi sahibi olmanızı gerektirecek alanları da belirlemenizi sağlar.

Bir problemi tanımlarken bazı bölümlerini küçük, anlaşılabilir veri ve işlem parçaları olarak gruplandırmaya çalışırız. Bu sayede problemi bir bütün olarak görmek yerine kolay aşılabilir hedefler haline getirmiş oluruz.

Çoğu zaman problemi çözmeye başlayamama, problemi anlayamamaktan kaynaklanır. Probleme odaklanmak ve bir çözüm için neyin gerekli olduğuna karar vermek amacıyla problemin ne olduğunu kendi ifadelerinizle yeniden yazmak problemi alt parçalara ayırmak faydalı olacaktır.

1.3.9: Algoritmik Problem Çözümü

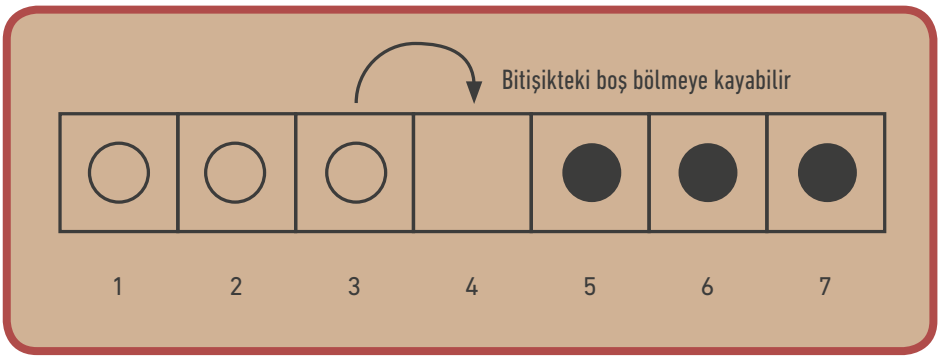
Bir problemi çözmek için algoritma tasarlarız ve bu algoritma tasarımı bir deneme yanılma çalışması şeklinde olur. Sorunu gerçekten çözüp çözmediğimizi görmek için test ederiz. Sorun çözüldüyse algoritma başarılıdır. Sorun çözülmediyse tekrar deneriz. Genellikle, herhangi bir belirgin sorunu çözmek için anlattığımız tekniklerin bir kombinasyonunu kullanırız.

Bilgisayarın belirli şeyler yapabileceğini unutmayın. Öyleyse temel kaygımız, bilgisayarın istenen çıktıyı üretebilmesi için girdi verilerini nasıl dönüştürmesi, kullanması, hesaplaması ve işlemesidir. Programlama dilinizde izin verilen talimatları ve veri türlerini aklınızda tutarsanız, kodlanması zor veya imkansız bir algoritma tasarlamazsınız.

Sırada çözmeniz için bir puzzle sizleri bekliyor.

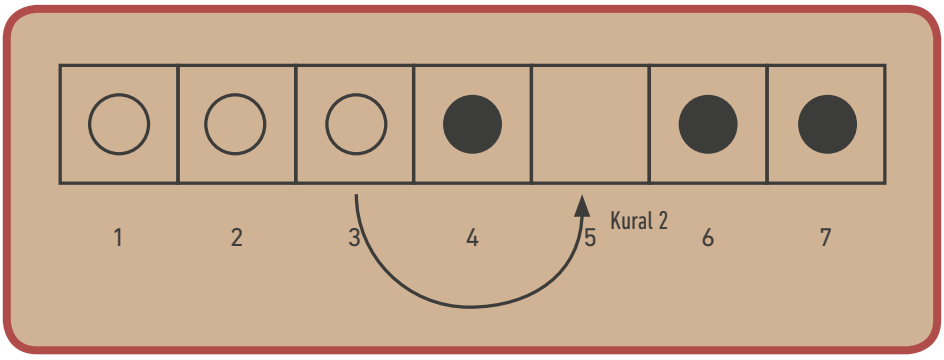
Etkinlik-1: Bulmaca (Puzzle) Çözümü

1975 yılında Kordemsky tarafından uyarlanan bulmacayı çözelim. 7 bölmeli bir çekmeceye sahibiz. 0'dan başlayarak her çekmece 6 ya kadar sıralanmıştır. 0,1 ve 2 numaralı ilk üç bölmede beyaz taş, 4,5 ve 6 numaralı bölmelerde ise siyah taş bulunmaktadır. 3 nolu bölme ise boştur. Görevimiz bu taşların yerlerini değiştirmektir. Siyah taşlar beyaz taşların yerine, beyaz taşlar da siyah taşların yerine geçecektir. Bu görevi yapabilmek için iki kuralı göz önünde tutacağız. Herhangi bir taş bitişik boş bir bölmeye kayabilir.Şekil5'de kural gösterilmektedir.

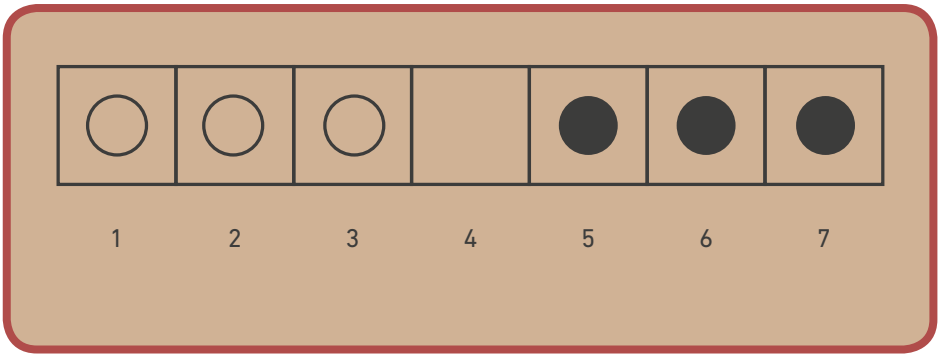


Şekil 5: Kural 1, Taşlar bitişiğinde bulunan boş bölmeye kayabilir.

Diğer kuralımız ise bir taş, boş bölmeye geçmek için diğer taşın üstünden atlayabilir. Şekil 6'da kural gösterilmektedir.



Şekil 6: Kural 2; 2 numaralı bölmedeki beyaz taş 3 numaralı siyah taşın üstünden atlayarak 4 numaraya geçebilir.



Şekil 7: Bulmaca etkinliği

Problemimiz basit; siyah ve beyaz taşların yerini değiştirmek. Bunun için problem çözme adımlarını kullanarak nasıl bir yol izleyeceğinizi belirlemelisiniz. İzlediğiniz her adımı, geri aldığınız değişiklikleri bile yazınız.

Karelerin içindeki nesne olarak bozuk paraların yazı tura kısımlarını kullanabilirsiniz. Olası iki hareket şu şekilde olabilir;

1: 2 nolu karedeki parçayı, 3'e taşıyın.

2: 4 nolu karedeki parçayı, 2'ye taşıyın.

Yazdığınız talimatı okuyan herhangi bir kişi, bu bulmacayı daha önce çözmemiş olsa dahi, sizin yazdıklarınız yardımı ile bu bulmacayı çözebilmelidir.

Şimdi çözümünüzü probleminizle birlikte değerlendirmelisiniz. Talimatlarınız ne kadar uzun sürüyor? Verilen bir problemi çözmek için kullanabileceğiniz birçok algoritma vardır, bunların bazıları diğerlerinden daha hızlıdır. Sizin algoritmanız kaç adımda çözüme ulaştıracak? Sürekli olarak yazdığınız talimatları gözden geçirin ve gelişmesini sağlayarak 15 hamlede çözebilecek duruma gelin.

Çözüm:

1. **2 nolu bölmedeki beyaz taşı, 3 nolu bölmeye kaydır.**
2. **4 nolu bölmedeki siyah taşı, 2 nolu bölmeye taşı.**
3. **5 nolu bölmedeki siyah taşı, 4 numaralı bölmeye kaydır.**
4. **3 nolu bölmedeki beyaz taşı, 5 nolu bölmeye taşı**
5. **1 nolu bölmedeki beyaz taşı,3 nolu bölmeye taşı**
6. **0 nolu bölmedeki beyaz taşı, 1 nolu bölmeye taşı**
7. **2 nolu bölmedeki siyah taşı, 0 nolu bölmeye taşı**
8. **4 nolu bölmedeki siyah taşı,2 nolu bölmeye taşı**
9. **6 nolu bölmedeki siyah taşı, 4 nolu bölmeye taşı**
10. **5 nolu bölmedeki beyaz taşı, 6 nolu bölmeye taşı**
11. **3 nolu bölmedeki beyaz taşı, 5 nolu bölmeye taşı**
12. **1 nolu bölmedeki beyaz taşı, 3 nolu bölmeye taşı**
13. **2 nolu bölmedeki siyah taşı 1 nolu bölmeye taşı**
14. **4 nolu bölmedeki siyah taşı, 2 nolu bölmeye taşı**
15. **3 nolu bölmedeki beyaz taşı, 4 nolu bölmeye taşı**

BÖLÜM

ALGORİTMALAR, ÇOCUKLARA NASIL AÇIKLANIR?

BÖLÜM 2:
ALGORİTMALAR, ÇOCUKLARA NASIL AÇIKLANIR?

Algoritma kelimesi günlük yaşamımızda pek de çocuklarla alakalı görünmeyebilir, fakat gerçek şu ki algoritmalar her yerdeler. Kullandıkları tabletlerden, oynadıkları oyunlara, yaşamlarının her alanında algoritmaları kullanıyorlar. Basit, zor veya karmaşık olmasına rağmen oldukça basit bir temel üzerine inşa edilen algoritmaları inceleyelim.

2.1: Algoritma Nedir?

Algoritma; bir problemi çözmek veya bir görevi tamamlamak için adım adım detaylandırılmış talimatlardır. Programcılar, bilgisayarın bir görevi nasıl yapılacağını tarif eden algoritmalar yazarlar.

Genel olarak düşündüğümüzde algoritmalar hayatımızın her yerindedir. Sabah uyanıp okula gelinceye kadar ki süreçten tutun, akşam uyumadan önce diş fırçalamamıza kadar her aşamada birçok algoritma, biz farkında olmadan, beynimiz tarafından oluşturulur, test edilir, uygulanır, güncellenir ve sonlandırılır. Şekil 8'de bir öğrencinin sabah uyandığında gerçekleştirebileceği görevlerin basit bir algoritması görülmektedir.

Şekil 8: Sabah kalkıp okula gidinceye kadar yapılacak işlerin algoritması

2.2 Çocuklar Kendi Algoritmalarını Yazabilir

Öğrencilerinizin kahvaltı hazırlamak, diş fırçalamak, yemek yemek, ayran hazırlamak gibi basit görevler için algoritma yazmalarını sağlayabilirsiniz. Bu sayede çocuklar; farkına bile varmadan, tekrar eden döngüleri (3 kez üst dişleri fırçala), görevleri sıralamayı (önce yoğurdu karıştır, ardından üzerine su koy), koşula bağlı mantık yapısını(yemek soğuk ise ısıt) öğrenecektir.

Öğrencilerimiz, algoritmaları mümkün olduğunca küçük adımları düşünerek hazırlamalılar. Bilgisayarlar, bizim niyetlerimizi okuyamayacağı için yapılması gereken bir görevin tam olarak tanımlı hale getirilmesini bekler. Eğer ki ayran bardağına ne kadar ayran doldurulması gerektiğini belirtmezseniz ayranın yere döküldüğünü görmeniz mümkündür. Bunun için *"bardak doluncaya kadar..."* şartı sunularak ne kadar doldurulması gerektiğinin belirtilmesini sağlayabilirsiniz.

Etkinlik Örneği: Robotun kontrolü sende...

Öğrencilere kodlama terimlerinden olan algoritma ve sıralama ifadeleri öğretmek amacıyla *"robotun kontrolü sende"* etkinliği düzenlenir.

İçerik:

Öğrenciler iki gruba ayrılır. Gruplardan birisi programcı diğeri ise robotu canlandıracaktır. Programcı olan öğrenciler, robot olan öğrenciye sırasıyla kapıyı aç, ilerle, tahtayı sil, kapıyı kapat gibi komutlar vererek çeşitli görevleri yerine getirmesini sağlayacaktır.

Öğrenci gruplarındaki görevleri değiştirerek her iki rolü de öğrencilerin oynamasını sağlayınız.

Etkinlik tamamlandıktan sonra komutların basitliğinin öneminden ve komutların adım adım sırasıyla verilmesinin öneminden bahsedebilirsiniz.

2.3 Algoritmik Düşünmenin Faydaları

Algoritmik düşünme veya bir problemi adım adım çözebilme yeteneği özellikle matematik ve fen bilimlerinde oldukça önemlidir. Çocuklar farkında olmadan özellikle matematikte algoritmaları kullanırlar. Bir bölme işlemi yaparken, çarpma, çıkarma gibi işlemlerden faydalanırlar. Algoritmik düşünme; çocukların, problemleri parçalara ayırmalarını ve çözüme ulaşabilmek için farklı adımları bir süreç içinde yürütebilmelerini sağlar.

Bir problemi parçalara ayırabilmek için problemin veya çözülmesi gereken durumun öncelikle tanımlı hale getirilmesi gerekir.

Etkinlik-2*: Sabah uyandığımda...*

Öğrenciler sabah uyandıklarında yaptıkları işlerin bir listesini oluşturacaklardır.

Ön Bilgi:

· Algoritmanın bir problemi çözmek veya bir görevin yerine getirilmesini sağlamak için gerekli talimatlar olduğunu açıklayınız.

· Bir problemi çözmek için plan yaptığınızda bunun birçok yolu olabilir.

· Bilgisayarların kendilerine verilen komutları anlayabilmesi için onun anlayacağı dilde konuşmamız gerektiğini ve bunun için insan ile bilgisayar arasında tercümanlık görevini yerine getiren birçok programlama dilinin bulunduğunu açıklayınız.

İçerik:

· Öğrencilerinizin *"sabah uyandığımda"* etkinliğinde bir algoritma hazırlayabilmesi için konuları belirleyiniz.(Kahvaltı yapmak, Dişleri fırçalamak, Okula gitmek gibi)

· Öğrenciler seçmiş oldukları etkinliğin algoritmasını hazırlarken bu etkinliği daha önce hiç görmemiş, duymamış birisine anlatıyormuş gibi her adımı en ince ayrıntısına kadar yazmasını isteyiniz.

· Öğrencilerinizden algoritmayı yazarken bir veya iki tane seçim yapılması gerektiren durumun da olmasını isteyiniz. Örneğin diş fırçalarken suyun soğuk olup olmadığını veya kahvaltı yaparken çayın sıcak olup olmadığını sorgulayarak farklı seçimler yapabilirler. Şarta bağlı talimatları sarı renge boyayabilirler.

· Sonraki adımda ise tekrarlayan görevlerin olup olmadığını öğrencilerinize sorabilirsiniz. Örneğin diş fırçalarken 3 defa fırçalanıp fırçalanmadığını veya 5 dilim ekmek yiyip yemediklerini sorgulayabilirler. Tekrarlayan ifadeleri mavi renge boyayabilirler.

· Sarı renkte olan ifadeler bilgisayarın karar verme noktalarıdır. Bu noktalarda verilen kararlardan sonra program akışı değişmektedir.

· Mavi renkte olan ifadeler ise döngülerdir. Bilgisayar döngülerin belirtildiği sayı kadar görevleri tekrar tekrar yerine getirir.

· Bilgisayarların gerçekten akıllı gözüktüğünü ama aslında sadece yazmış olduğumuz algoritmada olduğu gibi adım adım talimatları takip ettiklerini açıklayabilirsiniz.

2.4 AKIŞ ŞEMASI

Akış şemaları, algoritmaların görseller kullanılarak ifade edilmesidir. Bu işlemi yaparken farklı geometrik şekiller kullanılır. Algoritmaları yazarken emir kipi ile kurduğumuz cümlelerden faydalanırız. Ancak akış şeması ile yapılması gereken görevler net bir şekilde ifade edilir ve görseller ile desteklenir. Örneğin bir veri girişinin yapılması gereken adım için kullanılan bir şekil kullanıcıyı daha etkili uyarabilir. Şekiller net görevleri ifade ettiği için farklı bir şekilde yorumlanma olasılığı da ortadan kalkmış olur.

Akış şemalarının kullanım alanları;

· Algoritmalarda

· Bir ürünün ortaya çıkıncaya kadar ki aşamalarını gösterebilmek için.

· Bir projenin planlanın aşamalarını göstermek için.

Akış şemalarında semboller, oklar ve diğer elemanlar kullanılır:

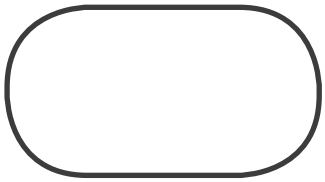
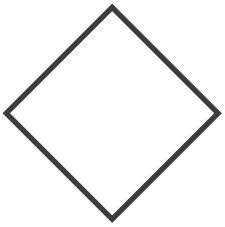
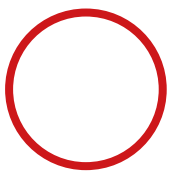
· Semboller bir sürecin geçirdiği işlemleri, çeşitli adımları ve hareketleri gösterir.

· Oklar adımların sırasını veya farklı seçenekleri gösterir.

· Diğer unsurlar, kararlar, insanlar veya süreçler.

2.4.1 Akış Şeması Oluşturma

Aşağıdaki tabloda akış şemalarında kullanılan ortak şekiller gösterilmektedir. Genellikle akış şemaları yukarıdan aşağıya doğru ve soldan sağa doğru okunur. Akış şemaları okunurken "akış" olarak ifade edilebilir.

Şekil	İsim	Açıklama
	Akış Çizgisi	Akış şemasının akacağı yönü gösteren oktur.
	Terminal	Terminaller, daire, oval, köşeleri yuvarlak dikdörtgenler olarak kullanılabilir. Akış şemasının başlangıcını veya bitişini belirtmek için kullanılır.
	İşlem (Süreç)	Bir işlem dikdörtgen ile gösterilir. Örn: Sayının karekökünü al, Ekrana şekil çiz gibi
	Karar	Akış devam ederken bir noktada, Evet/Hayır sorusu veya Doğru/Yanlış ifadesi ile bir karar verilir. Eşkenar dörtgen ile gösterilir. Karar kutusundan çıkacak olan her bir ok "Evet/Hayır", "Doğru/Yanlış" ile gösterilmelidir.
	Giriş/Çıkış	Giriş/Çıkış işlemleri için bir paralelkenar kullanılır. Bu kutu, bir bilginin alınması ve işlenen bilginin görüntülenmesi amacıyla kullanılır.
	Konnektör (Bağlayıcı)	Bu sembol akış şemalarında, akışın bir noktadan diğerine zıplamasını göstermek için ve Ana programdan ayrı bir alanda tanımlanan alt süreçlerden atlamak için de kullanışlıdır.

Tablo 1: Akış şeması sembolleri

Etkinlik-3: ***Üç sayının ortalamasını hesaplayan algoritma ve akış şeması***

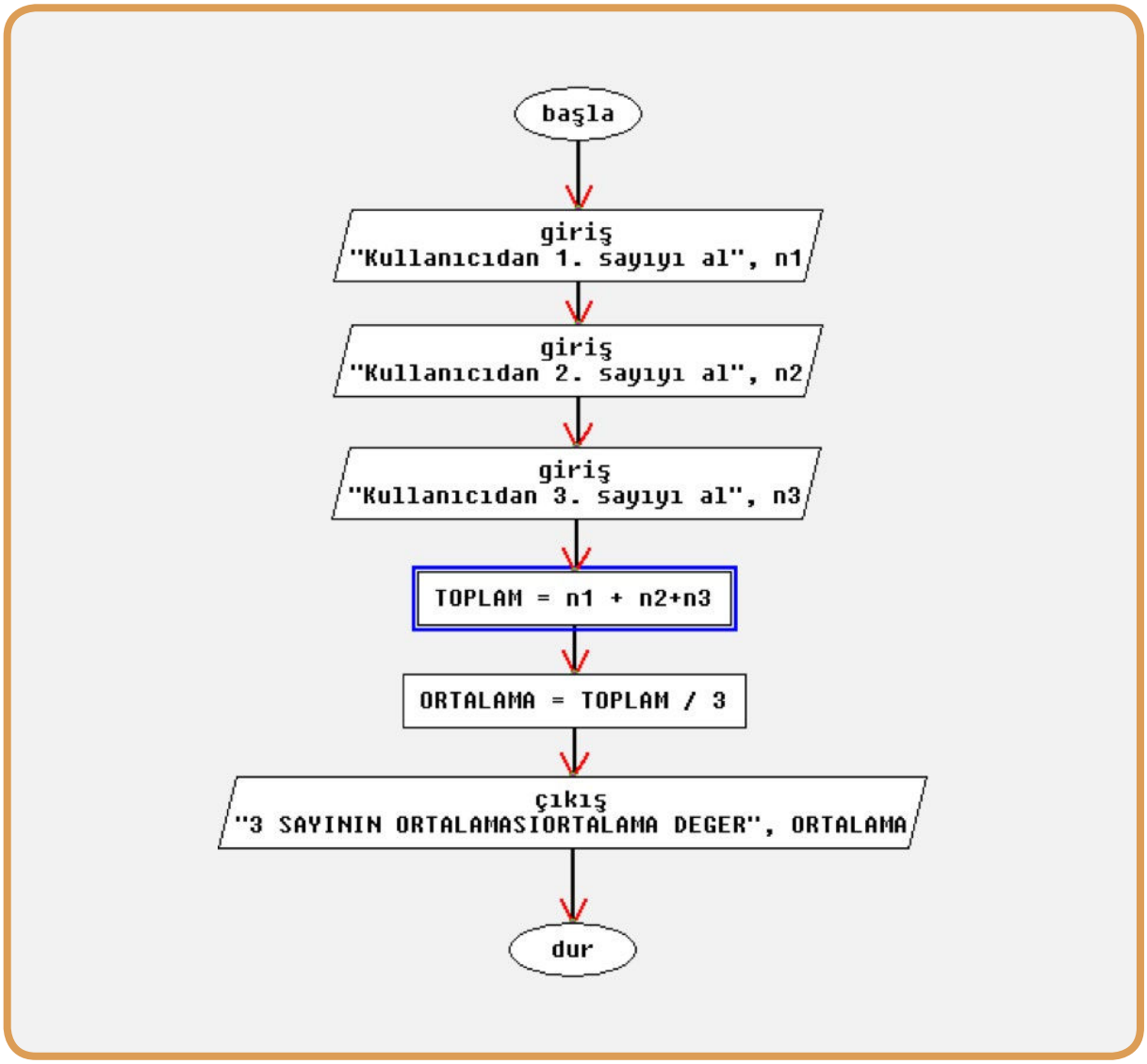
Etkinliğimizde kullanıcıdan alınan üç sayının ortalamasını hesaplayan ve görüntüleyen bir algoritmayı hazırlayacağız ve bu algoritmanın akış şemasını çizeceğiz.

Kullanıcıdan alınan üç sayının ortalamasını hesaplayan algoritma:

1-Başla
2-n1 sayısını kullanıcıdan al
3-n2 sayısını kullanıcıdan al
4-n3 sayısını kullanıcıdan al
5-Toplam =n1+n2+n3 işlemini yap
6-Ortalama= Toplam/3 işlemini yap
7-Ortalama'yı görüntüle
8-Bitiş.

Akış Şeması:

Kullanıcıdan alınan üç sayının ortalamasını hesaplayan algoritmanın akış şeması şekil 9'da verilmiştir.



Şekil 9: Kullanıcıdan alınan üç sayının ortalamasını hesaplayan algoritmanın akış şeması

Akış şeması oluşturmak için ofis programlarını kullanabileceğiniz gibi ücretsiz olarak sunulan Flow chart visual programming language'de kullanabilirsiniz.

BÖLÜM

PROGRAMLAMA

BÖLÜM 3 PROGRAMLAMA

3.1 PROGRAM NEDİR?

Bilgisayar; bazı bilgileri alan, bunları işleyen ve sonuç olarak kullanıcıya bir çıktı veren makinedir. Kendimize ait olan adına "beyin" dediğimiz bir bilgisayara sahibiz. Öğretmen 4x4 = ? şeklinde bir soru sorduğunda beynimiz hemen 4+4+4+4 işlemini yapar. Sonuç olarak hesaplanan değeri kullanıcıya gösterir.

Bilgisayar verileri işlemek için bilgisayar komut ya da komutlar almalıdır. Bir dosyayı açmak veya bir müzik dosyasını dinlemek için bile milyonlarca kod satırı yazılması gerekmektedir. Her yapılacak işlem için kullanıcıdan komut almak ise bilgisayarın pratikliğini ortadan kaldıracaktır. Bu soruna çözüm olarak; programların bilgisayar belleğine kaydedilmesi uygun görülmüştür. Program çalıştırıldığında bilgisayar, saklanan programdan her komutu sırayla alır ve çalıştırır.

1950 yılında kaydedilmiş programı çalıştırabilen ilk bilgisayarın(UNIVAC 1101) geliştirilmesinden bu zamana kadar hem bilgisayar olarak adlandırdığımız cihazların hem de çalıştırılan programların büyük değişimler geçirdiğini söyleyebiliriz. Günümüzde çok farklı elektronik tasarımlara sahip cihazların, pek çok farklı dilde yazılmış programları çalıştırabildiğini görüyoruz.

Bilgisayar veya akıllı cihazların istediğimiz iş ve işlemleri yerine getirebilmesi için programlanması gerekir. Bir çalar saatin sabah 7'ye kurulması, elektrikli su ısıtıcısının, suyun sıcaklığı 90 dereceye ulaştığında kapanması, elektrikli sobanın 40-50 derece arasında ısındığında ısıtma işlevini durdurması gibi olaylar da aslında mekanik olarak cihazın programlanmasını ifade etmektedir.

Bir cihazın yetenekleri göz önünde bulundurularak, yerine getirilmesi gereken görevleri, cihaza aktarma ve anlatma işlemini programlama olarak tanımlayabiliriz. Bilgisayarı programlayabilmek için bilgisayarın anlayabileceği dilden konuşmamız gerekir. Bu amaçla bilgisayar ve programcı arasında iletişimi sağlayacak bir tercümana ihtiyaç vardır. Derleyici olarak adlandırılan bu tercüman, herhangi bir programlama dilinde yazmış olduğumuz komutları bilgisayarın anlayacağı makine diline çevirir. *Not: Makine dili; elektronik olarak programlanan herhangi bir cihazın iletişim kurduğu ve programlandığı dildir. 1 ve 0 karakterleri bu dilin alfabesini oluşturur. Her* türlü bilgi 1'ler ve 0'lar ile ifade edilir. Böylece insan ve makine arasında bir iletişim sağlanmış olur.

3.2 PROGRAMLAMA DİLLERİ:

Çocukların, öğrencilerimizin ve ilgi duyan amatörler için Scratch gibi blok tabanlı programlama dillerinden, Python veya JavaScript gibi metin tabanlı programlama dillerine doğru bir öğrenme süreci uygun görülür, önerilir. Çocukların metin tabanlı programlama diline geçip geçemeyeceğini merak eden ebeveynler için özet olarak;

· Blok tabanlı veya metin tabanlı programlama dili arasında tercih yapmak o kadar da önemli değil.
· Önemli olan, belirli bir dil ile çocuğunuzun yapmak istediği şeyi, verimli ve zevkli bir şekilde yapabilmesidir.
· Çocuğunuzun/Öğrencinizin uzmanlık seviyesine ve yapmayı düşündükleri projeye uygun olan dili bulmaya çalışın.
· Blok tabanlı programlama dilleri sadece çocuklar için geliştirilmemiştir. Yetişkinler de kullanabilir.

3.3 ÇOCUKLARA NEDEN KODLAMA ÖĞRETİYORUZ?

Bazı çocuklar geleceğin programcılarıdır. Bazı çocuklar ise becerilerini desteklemek için kodlamanın bazı unsurlarını gerektiren işlere sahip olacaklardır. Kodlama; günümüz ve geleceğimizin insan davranışlarını şekillendiren, alışkanlıklarımızı, hareketlerimizi, işlerimizi belirli önceliklere göre sıralayarak yapmamızı sağlayabilen, makineler ile aramızdaki iletişimi kurabilen yeni bir dildir.

3.4 Blok Tabanlı? Metin Tabanlı?

Metin tabanlı programlama dillerini tanımlamak daha kolaydır. Bunlar bilgisayar klavyesini kullanarak yazdığımız ve metin dosyası olarak sakladığımız dillerdir.

Blok tabanlı bir dil genellikle "yazma" eyleminden ziyade sürükle ve bırak işlevini kullanır. Bloklar veya elemanlar üzerinde simge-metin etiketlerini kullanabilir. Diyaloglar ve açılır menü seçenekleri sıklıkla kullanılır.

Scratch ve Blockly gibi blok tabanlı programlama dilleri oldukça popülerdir. Metin tabanlı diller ise profesyonel yazılım geliştiriciler tarafından kullanılan, "gerçek" programlama dili olan C/C++, Python, JavaScript gibi programlama dilleridir.

Blok tabanlı dillerin önemli özelliklerinden biri, çocukların bir komut listesini veya karmaşık bir söz dizinini hatırlamak zorunda kalmamalarıdır. Birçok profesyonel yazılım geliştirici ise saatlerce çalışırken, kullandığı metin tabanlı programlama dilinin söz dizimini (syntax) ezberlemek ve kurallara dikkat etmek zorundadır.

Metin tabanlı bir programlama dilinde karşılaşabileceğiniz, noktalı virgülü unutma, süslü veya kapalı parantezi açtıktan sonra kapatmayı unutmak gibi birçok muhtemel hata ile blok tabanlı programlamada karşılaşmayız. Böylece öğrenciler proje mantığını düşünmek için daha fazla zaman ayırabilirler.

3.5 NASIL ÖĞRETEBİLİRİZ?

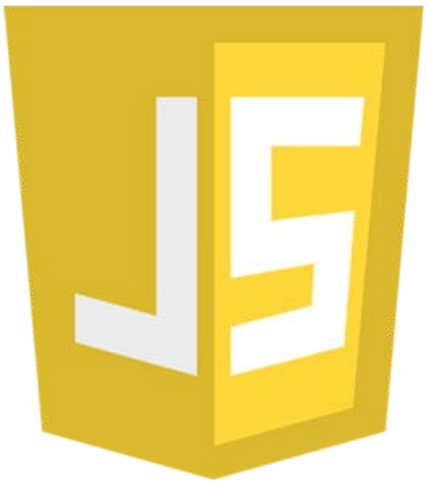
Öğrencilerimize/ çocuklarımıza kodlamayı öğretirken iki temel konu özellikle göz önünde bulundurulmalıdır:

· Faydalı, ilginç, eğlenceli projeleri üretebilme

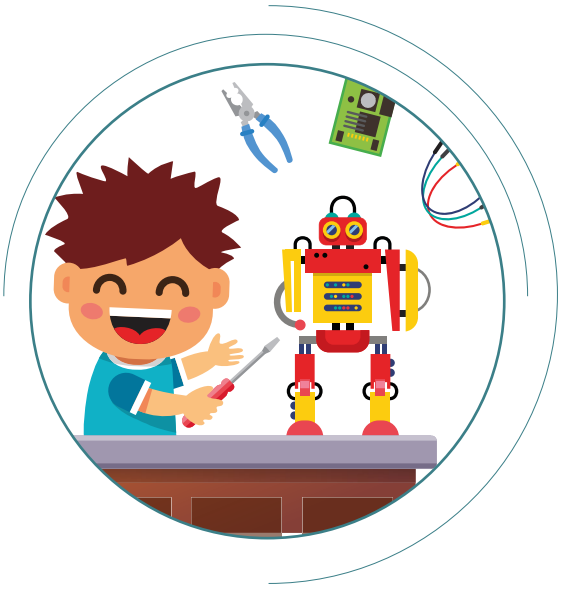
· Bu projeleri üretilebilmelerini sağlayacak **analitik düşünce** becerisini geliştirme

Kodlama eğitimi süreci, bilgisayar üzerinde belirli programları çalıştırmaktan ibaret değildir. Robotik uygulamalar, 3D tasarımlar, hayatın her alanında karşımıza çıkan problemlere yönelik farklı çözümler üretebilme gibi birçok disiplini kapsayan bir eğitim-öğretim-uygulama sürecidir. Uygulama ağırlıklıdır. Mümkün olduğunca mantık yürüterek, öğrenilen her bilgiyi sonraki aşamaya taşımayı hedefler. Bu esnada yeni bilgiler ile destekleyerek devam edilmesi gerekmektedir.

Öğrencilerimizin kodlama eğitimine başlayabilmeleri için bir sonraki bölümde blok tabanlı Scratch yazılımını inceleyeceğiz. Metin tabanlı kodlama konusunda öğrencilerimize yardımcı olabilecek programlama dilleri ise python, visual basic, java script olabilir. Metin tabanlı programlama dili mevcut kılavuzun içeriği dahilinde değildir.



BÖLÜM



SCRATCH İLE KODLAMA

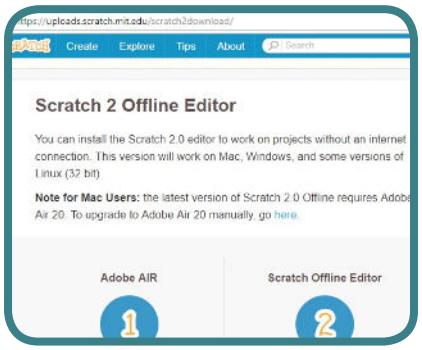
BÖLÜM 4 SCRATCH İLE KODLAMA

Scratch ile animasyonlar, hikayeler, oyunlar ve etkileşimli uygulamalar geliştirebilirsiniz. Kod bloklarını kullanarak hazırlayacağınız uygulamaları başka kullanıcılar ile paylaşabilirsiniz.

Bilgisayarı programlamak; onun anlayacağı dilden konuşmayı gerektirir. Scratch basit, güçlü ve öğrenmesi çok kolay olan bir tercümandır. Bilgisayarın hoparlöründen, klavyesine, kamerasından fareye kadar çeşitli donanımlarıyla etkileşimli uygulamalar geliştirebilirsiniz. Ayrıca hazırlayacağımız oyunlarla eğlenceli vakit geçirebilir, animasyonlar hazırlayıp bir hikayeyi sahnede canlandırabilirsiniz.

Scratch; öğrencilerin, çocukların ve ilgili yetişkinlerin kullanabileceği blok tabanlı kodlama aracıdır. MIT Medya tarafından geliştirilen Scratch projesi kullanıcılarına iki adet çalışma ortamı sunar.

İlk çalışma ortamı Offline Editor olarak isimlendirilen masaüstü uygulamasıdır. Windows, Linux ve Mac Os X işletim sistemlerine çalışabilir. Offline editörün çalışabilmesi için gerekli olan Adobe AIR ve yükleme dosyalarını https://uploads.scratch.mit.edu/scratch2download/ resmi sitesinden erişebilmektesiniz. Şekil 10'da ilgili sayfanın ekran görüntüsü görülmektedir.



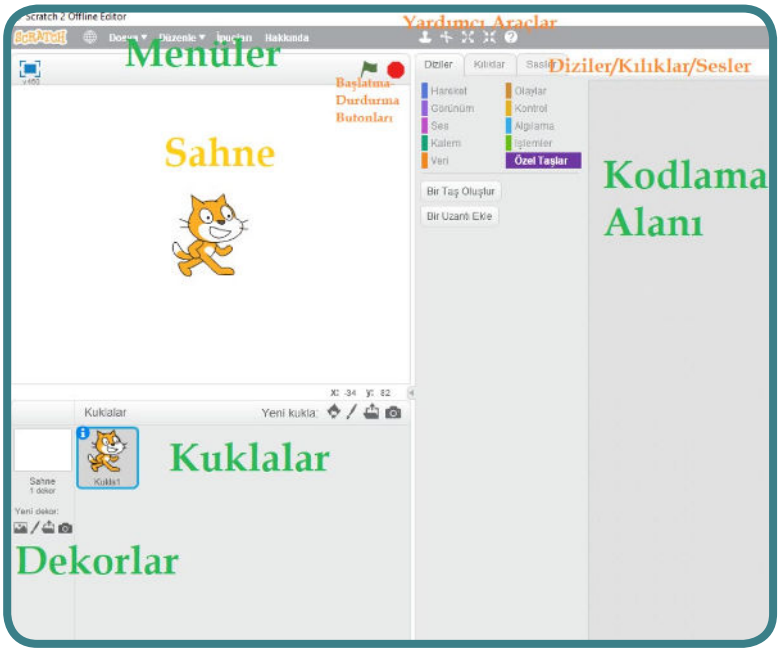
Şekil 10: Scratch kurabilmek için gerekli Adobe AIR ve offline editor kurulum dosyası indirme linkleri

Öncelikle bilgisayarınızda Adobe AIR'in son sürümü yüklü değilse 1 numaralı sekmede gösterilen indirme linklerinden Adobe AIR uygulamasını indirip kurmanız gerekmektedir. Ardından offline editörü kurarak uygulama geliştirmeye başlayabilirsiniz. Mevcut kılavuz hazırlandığında Adobe Air 20, Scratch ise 460 versiyonu ile kullanıma sunulmuştur.

İkinci çalışma ortamı ise Çevrimiçi (Online) Editördür. Web sayfası üzerinden scratch'ın son sürümü ile çalışabilir, çalıştığınız dosyaları online olarak kaydedebilir istediğiniz anda tekrar ulaşabilirsiniz.

4.1 ÇEVRİMDIŞI (OFFLINE) EDİTÖR

Şekil 11'de Scratch için geliştirilen çevrimdışı editörün ekran görüntüsü gösterilmektedir. Scratch editöründe Menüler, Yardımcı Araçlar, Diziler/Kılıklar/Sesler, Sahne, Kodlama alanı, Kuklalar ve Dekorlar bölgeleri bulunur.



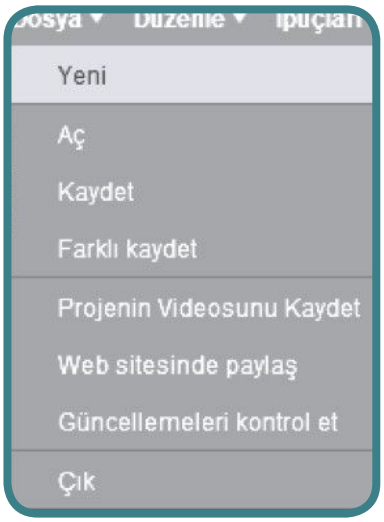
Şekil 11: Scratch çevrimdışı editör

4.1.1 MENÜLER

Scratch ile çalışırken menüleri kullanmaktayız. Yeni bir dosyayı oluşturma, var olan scratch dosyasını açma, video kaydetme gibi özelliklere menüler üzerinden ulaşabilirsiniz. Scratch editöründe Dosya menüsünün sol tarafında bulunan dünya simgesi editörün dilini değiştirmek için kullanılır ve Şekil 12'de bu simge gösterilmiştir.

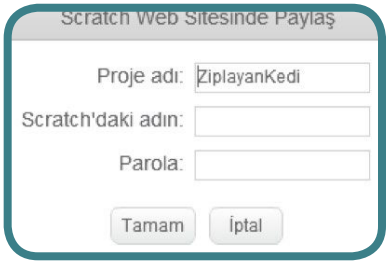
Şekil 12: Dünya simgesine tıklandığında editör dili değiştirilebilir.

4.1.1.1 Dosya Menüsü: Dosya menüsünde; Yeni butonuna tıklayarak yeni bir çalışma sayfası açabiliriz. Önceden kayıtlı scratch dosyalarını açmak için Aç butonu, Üzerinde çalıştığımız dosyayı kaydetmek için Kaydet butonu, çalışılan dosyanın farklı bir isim veya farklı bir dosya konumuna kaydetmek için Farklı Kaydet butonu kullanılır. Şekil 13'te Dosya menüsü gösterilmektedir.



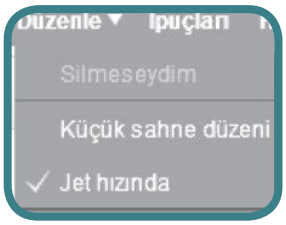
Şekil 13: Dosya menüsü

Scratch programında hazırlanan uygulamalar .sb2 uzantısı ile kaydedilir. Üzerinde çalıştığımız uygulamanın videosunu otomatik olarak kaydetme özelliği de Scratch uygulamasında sunulmuştur. 60 saniyelik bir video kaydı, flv formatında oluşturularak kaydedilir.
Web sitesinde paylaş seçeneği ile uygulamanızın scratch.mit.edu adresinde paylaşılmasını sağlar. Bu seçenek için scratch.mit. edu adresine kayıtlı olmanız ve kullanıcı adı ve şifre bilginizin açılan pencereye girilmesi gerekir. Şekil 14'te "Web sitesinde paylaş" penceresi gösterilmektedir. Güncellemeleri kontrol et sekmesi ise uygulamanın güncel sürümü olup olmadığını kontrol eder. Çık sekmesi ise uygulamadan çıkmayı sağlar.



Şekil 14: Web sayfasında proje paylaşım ekranı

4.1.1.2 Düzenle Menüsü: Sahnede silinen bir öğeyi geri almaya yarayan "Silmeseydim" sekmesinin yanısıra Küçük sahne düzeni" ve "Jet hızında" (Turbo mod) seçenekleri de bulunur. Matematiksel işlemlerin daha hızlı yerine getirilmesini sağlamaktadır. "Silmeseydim" butonuna tıklandığında en son yapılan işlem geri alınır. "Küçük sahne düzeni" ise sahnenin sol üst köşeye sığdırılarak kodlama alanı için daha fazla yer açılması sağlanır. "Jet Hızında"
(Turbo Mod) butonu ise kodların çalışabileceği en yüksek hızda çalışmasını sağlar. Matematiksel işlemlerin daha hızlı yerine getirilmesini sağlamaktadır.



Şekil 15: Düzenle menüsü

4.1.3 YARDIMCI ARAÇLAR

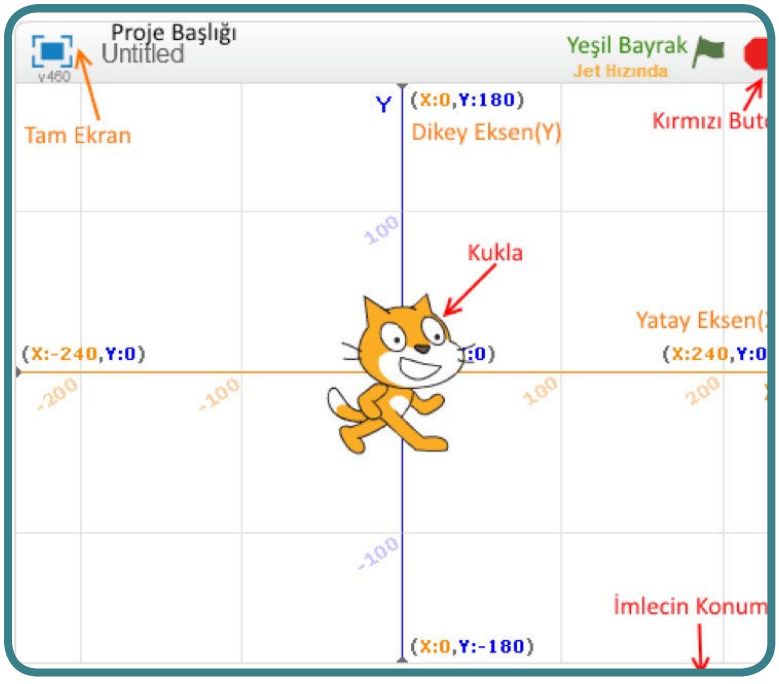
Sahnedeki kuklanın; kopyasını çıkartma, kesme, görüntüsünü büyütme ve görüntüsünü küçültme özelliklerini sağlar.

Şekil 16: Yardımcı araçlar

1:Kopyasını Çıkart: Sahnedeki kuklanın üzerine tıklandığında bir kopyasını kuklalar bölümüne ekler.
2:Kes: Sahnedeki herhangi bir kuklanın sahneden kesilmesini sağlar.
3:Büyüt: Sahnedeki kuklanın büyütülmesini sağlar.
4:Küçült: Sahnedeki kuklanın küçültülmesini sağlar.

4.1.4 SAHNE

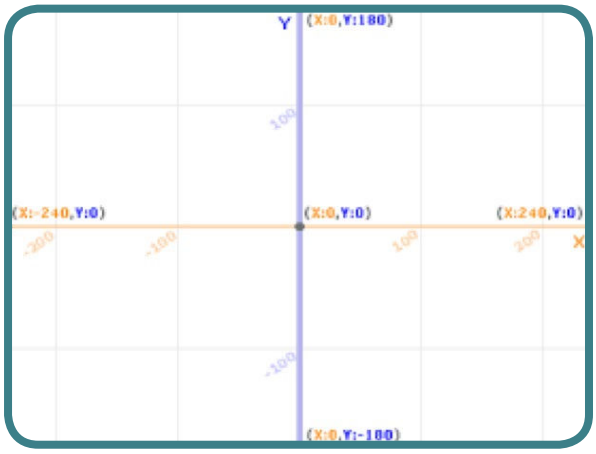
Sahne; kuklanın hareket edebildiği, dekorların görüntülendiği, belirli sınırlar ile çerçevesi çizilmiş olan çalışma alanıdır. Scratch ile hazırlayacağımız bir uygulama sahne olarak isimlendirilen alanda çalışır. Sahne üzerinde bulunan kuklalar, sahne tarafından izin verilen sınırlar içinde görülebilir. Sahne yatay ve dikey olmak üzere iki eksenden oluşur. Koordinat sistemi olarak isimlendireceğimiz bu sistemde, yatay ekseni X harfi, dikey ekseni ise Y harfi ile temsil edilir.



Şekil 17: Sahne ekran görüntüsü

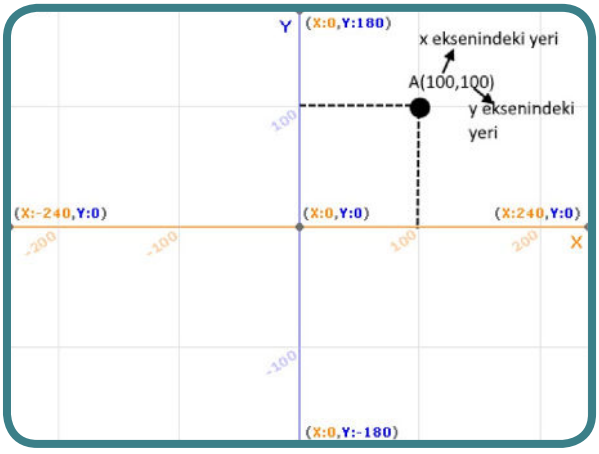
Sahne üzerinde bulunan butonları incelediğimizde;
Tam Ekran: Sahnenin ekranın tamamını kapsamasını sağlar.
Proje Başlığı: Çalıştığımız projenin adı bu bölümde görüntülenir. Kaydedilmemiş projeler Untitled ismi ile görüntülenir.
Yeşil Bayrak: Uygulamanın çalışması için kullanılan tetikleyici bayraktır.
Kırmızı Buton: Uygulamanın durdurulması için kullanılan butondur.
İmlecin Konumu: Fare imlecinin X ve Y eksenlerindeki yerini gösterir.
Kukla: Sahnede hareket eden, kullanıcı ile etkileşimde bulunan karakterdir. Yazılan kodlara göre hareket edebilir.

Koordinat Sistemi:



Şekil 18: Sahne, koordinat sisteminin gösterimi

Sahneyi oluşturan yatay ve dikey eksenler sahne üzerinde vardır ancak görüntülenmezler. Sahnede bulunan kuklanın hareket edebileceği sınırlar; yatay eksende (X ekseninde) sağ tarafta 240, sol tarafta -240'dır. Dikey eksende (Y ekseninde) ise sahnenin üst kısmında 180, sahnenin alt kısmında ise -180'dir.
Sahnedeki her kukla, bir nokta ile temsil edilir. Bu noktanın yerini belirtmek için iki sayı kullanırız. Birinci sayı, noktanın yatay eksendeki yerini, ikinci sayı ise noktanın dikey eksendeki yerini belirtir. Şekil 19'da A noktasının yatay ve dikey eksendeki yeri gösterilmektedir. Yatay eksende 100 dikey eksende 100 noktalarının kesiştiği nokta A noktasının bulunduğu konumu ifade eder.

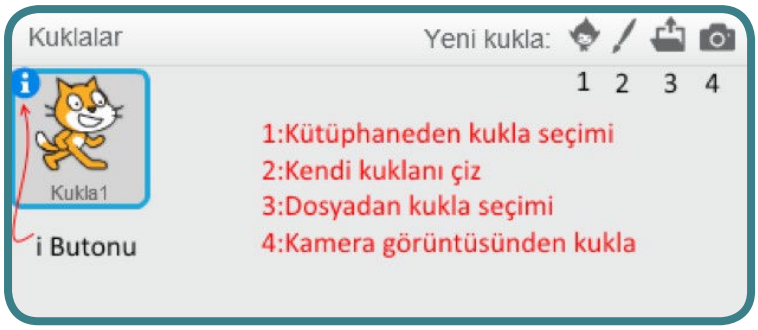


Şekil 19: Sahnede bir noktanın gösterimi

Sahnedeki kuklayı hareket ettirmek için, kuklayı temsil eden noktayı hareket ettirmemiz gerekir. Bu nokta her kuklanın tam orta noktasını ifade eder. Sahnede kuklanın sağa gitmesi için, kuklayı temsil eden noktanın X eksenindeki değerini arttırırız. Kuklanın sola gitmesi için, noktanın X eksenindeki değerini azaltırız. Kuklanın yukarı gitmesi için, noktanın Y eksenindeki değerini arttırırız. Kuklanın aşağı hareket etmesi için ise noktanın Y eksenindeki değerini azaltırız. Kuklanın herhangi bir yöne hareket etmesi için Hareket kod bloğundaki komutları kullanırız.

4.1.5 KUKLALAR

Sahnede bulunan bir kuklayı nasıl oluşturacağımızı, özelliklerini nasıl inceleyip değiştirebileceğimizi bu bölümde inceleyeceğiz. Şekil 20'de Kuklalar sekmesi gösterilmektedir.

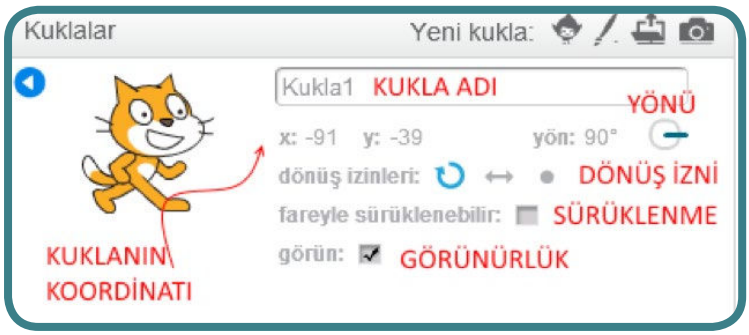


Şekil 20: Kuklalar ve özellikleri

Sahneye eklediğimiz her kuklayı bu sekme içerisinde görmekteyiz. Birden fazla kukla eklendiğinde o anda hangi kukla seçili ise o kuklaya kodların yazılacağıdır. Seçili olan kuklanın etrafında mavi çizgili bir çerçeve görüntülenir.

Kukla Özellikleri:

Sahneye eklediğimiz bir kuklanın özelliklerini incelemek için kuklanın sol üst köşesinde bulunan ***i*** butonuna basılır. Butona basıldığında şekil 21'de gösterilen özellikler penceresi açılır.

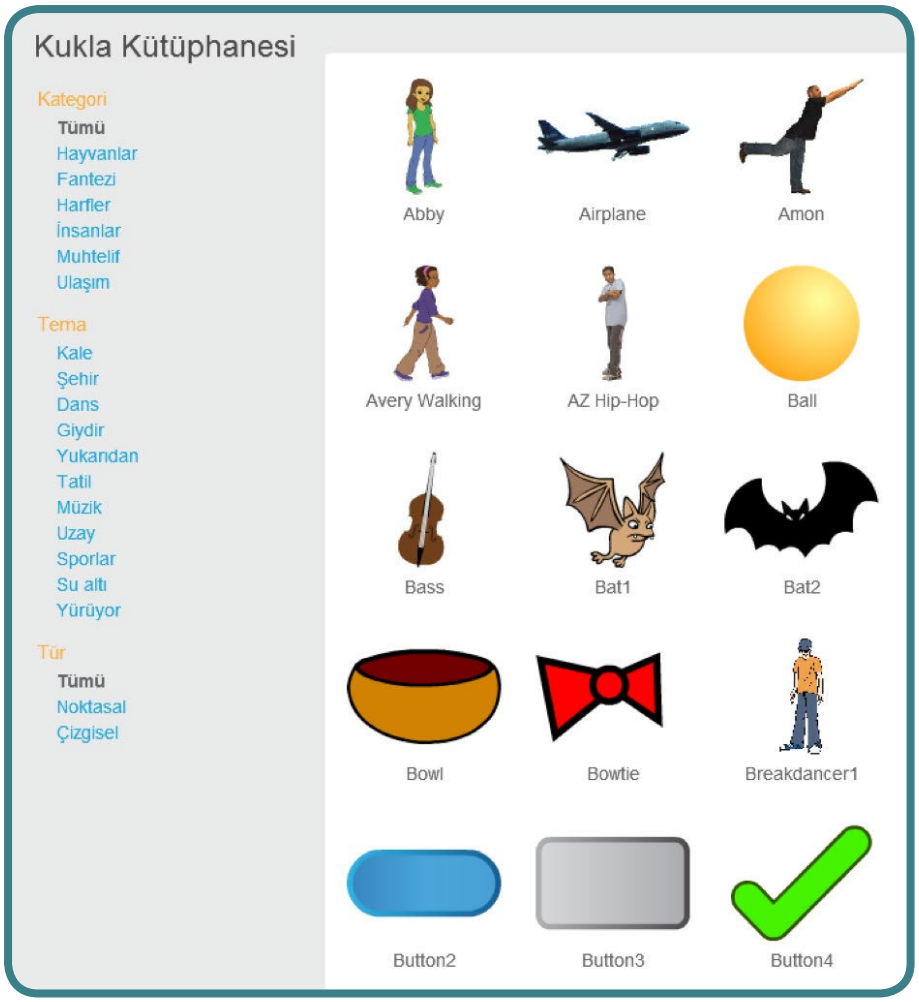


Şekil 21: Kukla özellikleri penceresi

Kukla Adı: Kuklanın sahneye eklendiğinde Kukla1, Kukla2... şeklinde isimlendirilmesi projeniz ilerlediğinde önemli sıkıntılara sebep olmaktadır. Bu amaçla kuklaya uygun isim verilmesi gerekir. İstenilen isim kukla adı bölümüne yazılır.
Kuklanın Koordinatı: Sahnede bir kuklanın konumunu bildiren özelliktir. Kuklanın X eksenindeki ve Y eksenindeki yerini gösterir.
Dönüş İzinleri: Kuklaya 3 dönüş izni verilmiştir; 1. izin her yöne dönebilmesi, 2. sadece sağa ve sola dönebilmesi ve 3. izin ise hiçbir yöne dönmemesidir.
Yönü: Kuklanın yönü burada gösterilen buton ile ayarlanabildiği gibi kod blokları ile de ayarlanabilmektedir. Kuklanın yön butonu dönüş izinlerine bağlı olarak çalışır. Eğer ki sadece sağa-sola dönebilmesine izin verilmiş ise kukla 90 ila -90 derece arasında yönünü değiştirebilir.
Sürüklenme: Kuklanın fare ile sürüklenebilmesine izin veren butondur. Bu özellik sahne tam ekran yapıldığında aktif olarak çalışmaktadır.
Görünürlük: Kuklanın sahnede görünüp görünmeyeceğini belirleyen özelliktir.

4.1.5.1 KUKLA EKLEME:

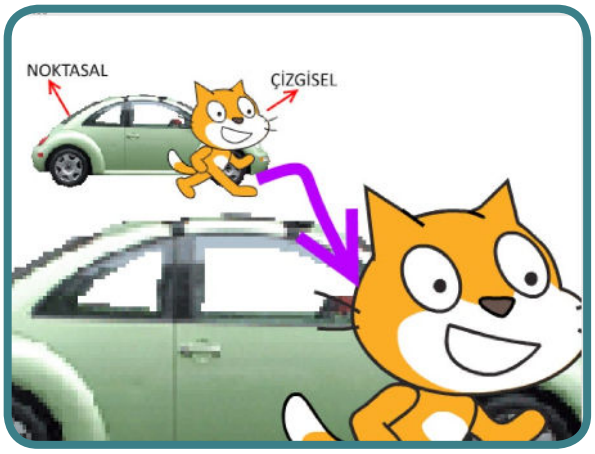
Uygulamaya yeni bir kukla eklemek istediğimizde 4 farklı yöntem kullanabiliriz.1numaralı yöntem Scratch programının kendi kütüphanesini kullanmaktır. Bu amaçla Şekil 20'de gösterilen 1 numaralı butona basmamız yeterlidir. Butona basıldığında Şekil 22'de gösterilen scratch kütüphanesi penceresi açılır. Kütüphane penceresini incelediğimizde sol tarafta kategorileri görebiliriz. Burada seçeceğimiz kategoriye göre, temaya göre ve türe göre kuklalar sınıflandırılmıştır. Kuklaların türe göre sınıflandırılmasını inceleyelim. Kuklalar, Noktasal(Bitmap) ve Çizgisel (Vector) olarak iki türe ayrılmıştır.



Şekil 22: Kukla Kütüphanesi

Çizgisel (Vector) Grafik Nedir?
Vektör grafikler çözünürlükten bağımsız olarak her bir nesne matematiksel ifadeler ile oluşturulan ve detay kaybetmeden herhangi bir boyuta yeniden ölçeklendirilebilen grafik türüdür.

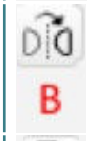
Noktasal (Bitmap) Grafik Nedir?
Bir imaj; her bir renk bilgisini içeren piksellerin(noktaların) yan yana ve alt alta dizilmesiyle oluşturulur. Bir fotoğrafı oluşturan en küçük nokta piksel olarak isimlendirilir. Fotoğraf çekildiği anda seçilen çözünürlük o fotoğraf karesinin piksel sayısını belirtir. Fotoğraftaki detayları görmek için zoom yapıldığında çekilen fotoğrafın piksel piksel görüntülendiğini görürüz. Şekil 23'de hem noktasal hem de çizgisel resimlerin büyütüldüğü bir çalışma alanı görülmektedir.



Şekil 23: Noktasal ve çizgisel grafikler arasındaki fark

Sahneye kukla eklemek için kullanılacak ikinci yöntem ise yeni kukla çiz seçeneğidir. Bu yöntemi kullanmak için 2 numaralı butona basılır. Ardından Şekil 24'de gösterilen çizim penceresi açılır. Çizim penceresini incelediğimizde sol tarafta "Kılıklar" sekmesi bulunur. Bu sekmede her kuklaya birden fazla kılık ekleyebilme imkanı sunar. Buna göre her kılık için ayrı ayrı çizimler yaparak kılıklar arasında geçiş yaptırdığımızda hareket edebilen kuklalar oluşturabiliriz.
Kuklalar bölümünde 1 numaralı buton kütüphaneden bir kılık eklemenizi sağlar. 2 numaralı buton yeni kılık çizmenizi sağlar. 3 numaralı buton bilgisayarda kayıtlı bir resmi kılık olarak ekleyebilmenizi sağlar. 4 numaralı buton ise bilgisayara bağlı fotoğraf makinesinden fotoğraf çekerek kılık olarak eklemenizi sağlar.
Kılık Adı: Çizilen kılığın adını belirtmemizi sağlar. Kılık1, Kılık2 şeklinde her yeni kılık otomatik olarak isimlendirilir.
Geri Al: Çizim alanında yapılan son değişikliği geri alır.
Yinele: Yapılan son değişikliği geri alma işleminden ileriye doğru değişiklikleri yerine getirir.
Temizle: Tüm kılık çizim sahnesini temizler.
Ekle: Sahneye kütüphaneden bir kılık ekleyebilmemizi sağlar.
Dışarıdan Al: Bilgisayarın dosya sisteminde kayıtlı resimleri/fotoğrafları kılık olarak ekleyebilmemizi sağlar.

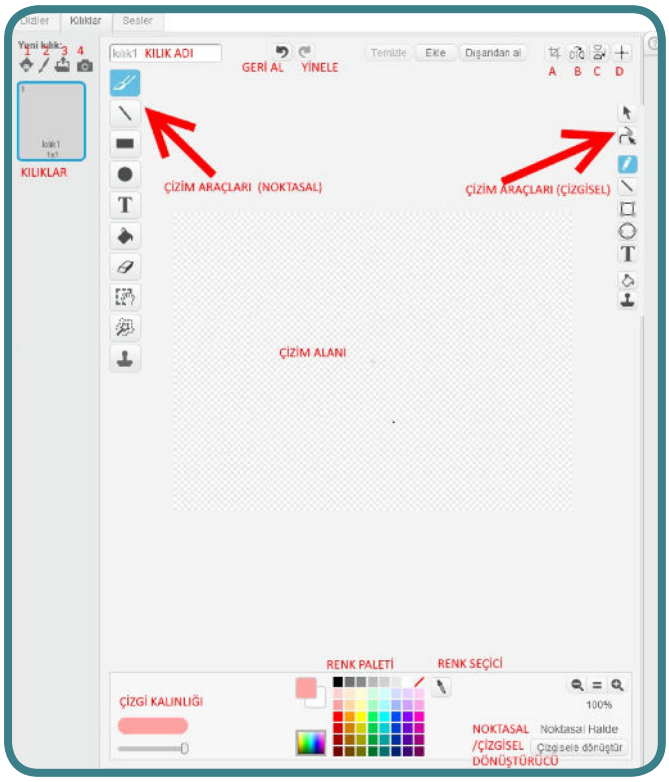
Kırpma butonu: Noktasal çizim yapıldığına oluşturulan imajın bir kısmını kesmek için kullanılır.



Sağ-Sol Yap: Kuklanın sağa ve sola çevrilmesini sağlayan komuttur.

Başaşağı Döndür: Kuklanın baş aşağıya döndürülmesini sağlayan komuttur.

Kılık Referansını Belirle: Sahnede çizdiğimiz kuklanın bazen sayfanın üst, alt veya yan kenarlarında görüntülendiğini görmekteyiz. Bunun için referans butonuna tıklar, ardından kuklanın tam ortasına tıkladığımızda kuklanın sayfanın ortasına ulaşmasını sağlarız.



Şekil 24: Kukla-kılık çizim alanı

Çizim araçları: Noktasal ve çizgisel çizim araçları birbirinden farklı görevlere sahiptir. Tablo2 ve Tablo3'de kukla çizim araçlarının görevleri belirtilmiştir

Simge	Adı	Görevi
	FIRÇA	Sahne üzerindeki çizgi kalınlığı ayarlanarak ve renk paletinden istenilen renk seçilerek, fare ile istenilen şekilde çizgiler çizmemizi sağlar.
	ÇİZGİ	Fare ile istenilen noktalara tıklanarak düz çizgi çizmemizi sağlar. Shift tuşuna basılarak çizildiğinde yatay veya dikey düz çizgi çizmemizi sağlar.
	DİKDÖRTGEN	 İstenilen boyutta dikdörtgen çizmemizi sağlar. Eğer shift tuşuna basılarak çizilirse kare çizer. Şekildeki butonlara basarak içi dolu veya içi boş olacak şekiller çizmemizi sağlar.
	ELİPS	 Sahnede istenilen boyutta elips çizmemizi sağlar. Eğer Shift tuşuna basılır ise çember çizilmesine imkan verir. Şekildeki butonlara basarak içi dolu veya içi boş şekiller çizmemizi sağlar.
	YAZI	 Sahnede istenilen noktaya yazı yazmamızı sağlar. Yazının fontnu ayarlamak için istenilen seçim yapılır.
	RENKLE DOLDUR	 Sahnede etrafı kapalı bir çimin içini doldurmak için kullanılır. Renk paletinden renk seçerek istenilen renk ile şekiller doldurulabilir. Butonlarını kullanarak doldurulan rengin dağılımı da seçilebilmektedir.
	SİL	 Sahnedeki şekillerin silinmesi için kullanılır. özelliği değiştirilerek silginin ni değiştirilebilir.
	SEÇ	Sahnedeki şeklin bir kısmını veya tamamını seçmemizi sağlar.
	ARKAPLANI SİL	Çizilen kılığın seçilen kısmını sahnede bırakarak arka planını siler.
	KOPYASINI ÇIKART	Seçilen kılığın kopyasını çıkartır.

Tablo 2: Noktasal çizim araçları

Simge	Adı	Görevi
	SEÇ	Scratch üzerinde çizgisel grafikler çizerken çizilen her nesne ayrı ayrı seçilebilir.
	ŞEKLİ DEĞİŞTİR	 Seçilen şeklin değiştirilebilmesi için üzerinde noktalar oluşturur. Böylece noktalardan tutup çekerek farklı şekiller elde edebiliriz.
	KALEM	Sahnede kalem ile şekiller çizebilmemizi sağlar.
	ÇİZGİ	Sahnede şekiller çizebilmemizi sağlar. Noktaları birleştirerek çizgilerin kapalı şekler oluşturması sağlanabilir.
	DİKDÖRTGEN	 İstenilen boyutta dikdörtgen çizmemizi sağlar. Eğer shift tuşuna basılarak çizilirse kare çizer. Şekildeki butonla ra basarak içi dolu veya içi boş olacak şekiller çizmemizi sağlar.
	ELİPS	 Sahnede istenilen boyutta elips çizmemizi sağlar. Eğer Shift tuşuna basılır ise çember çizilmesine imkan verir. Şekildeki butonlara basarak içi dolu veya içi boş şekiller çizmemizi sağlar.
	YAZI	 Sahnede istenilen noktaya yazı yazmamızı sağlar. Yazının fontunu ayarlamak için istenilen seçim yapılır.
	BİR ŞEKLİ BOYA	 Sahnede herhangi bir şekli boyamak için kullanılır. seçeneklerini kullanarak renk dağlımı değiştirilebilir.
	KOPYASINI ÇIKART	Herhangi bir şeklin üzerine tıklandığında kopyasını çıkartır.
	BİR KATMAN ÜSTE ÇIKART	Çizgisel (vektörel) grafiklerde çizim yaparken sahnede birden fazla şekil var ise bu şeklerin önde mi arkada mı görüneceği tasarım aşamasında belirlenebilir bu amaçla bir katman yukarı çıkaracağımız nesneler için bu buton kullanılır.
	BİR KATMAN ALTA İNDİR	Çizgisel (vektörel) grafiklerde çizim yaparken sahnede birden fazla şekil var ise bu şeklerin önde mi arkada mı görüneceği tasarım aşamasında belirlenebilir bu amaçla bir katman alta indireceğimiz nesneler için bu buton kullanılır.

Tablo 3: Çizgisel (Vektörel) çizim araçları

Renk Paleti: Herhangi bir çizim yaparken renk paletinden bir renk seçilebilir.
Renk Seçici: Bir çizim yaparken başka bir şeklin rengini yeni çizdiğimiz nesneye aktarabilmek için renk seçici kullanılır. Üzerine tıklanılan rengin değerleri kopyalanır ve yeni nesneye aktarılabilir.
Noktasal/Çizgisel Dönüştürücü: Çizilen şeklin noktasal (Bitmap) veya çizgisel (vektörel) grafik türü arasında değişim yapılmasını sağlar.

Sahneye kukla eklemek için kullanılacak 3. yöntem ise bilgisayarda bulunan resimleri kukla olarak eklemektir. Bunun için 3 numaralı butona basılır. Açılan pencereden bir resim dosyası seçilerek kukla alanına yeni bir kukla eklenmiş olur.

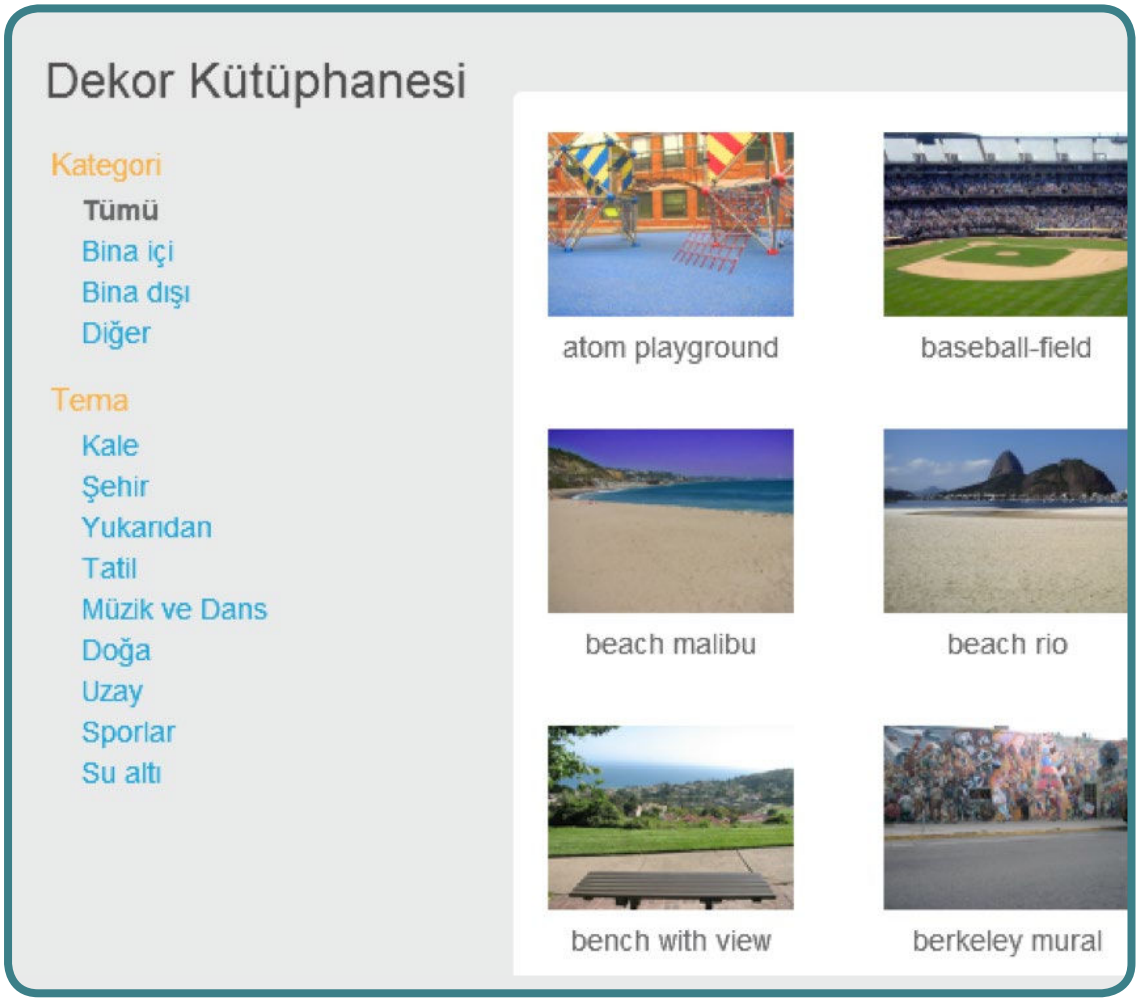
Sahneye kukla eklemek için kullanılacak 4. Seçenek ise bilgisayara bağlı kameradan fotoğraf çekerek kukla alanına eklemektir.

4.1.6 DEKORLAR

Sahne arka plan resmi dekor olarak adlandırılır. Sahne dekorunu hazırladığımız içeriğe göre değiştirebiliriz. Örneğin tasarladığımız bir oyunun her seviyesinde farklı bir dekor kullanarak kullanıcının dikkati daha fazla çekilebilir. Dekor oluşturmak için Kukla oluşturmada kullandığımız yöntemlerin aynısı kullanılır. Şekil 25'de dekor eklemek için kullanılacak butonlar gösterilmektedir.

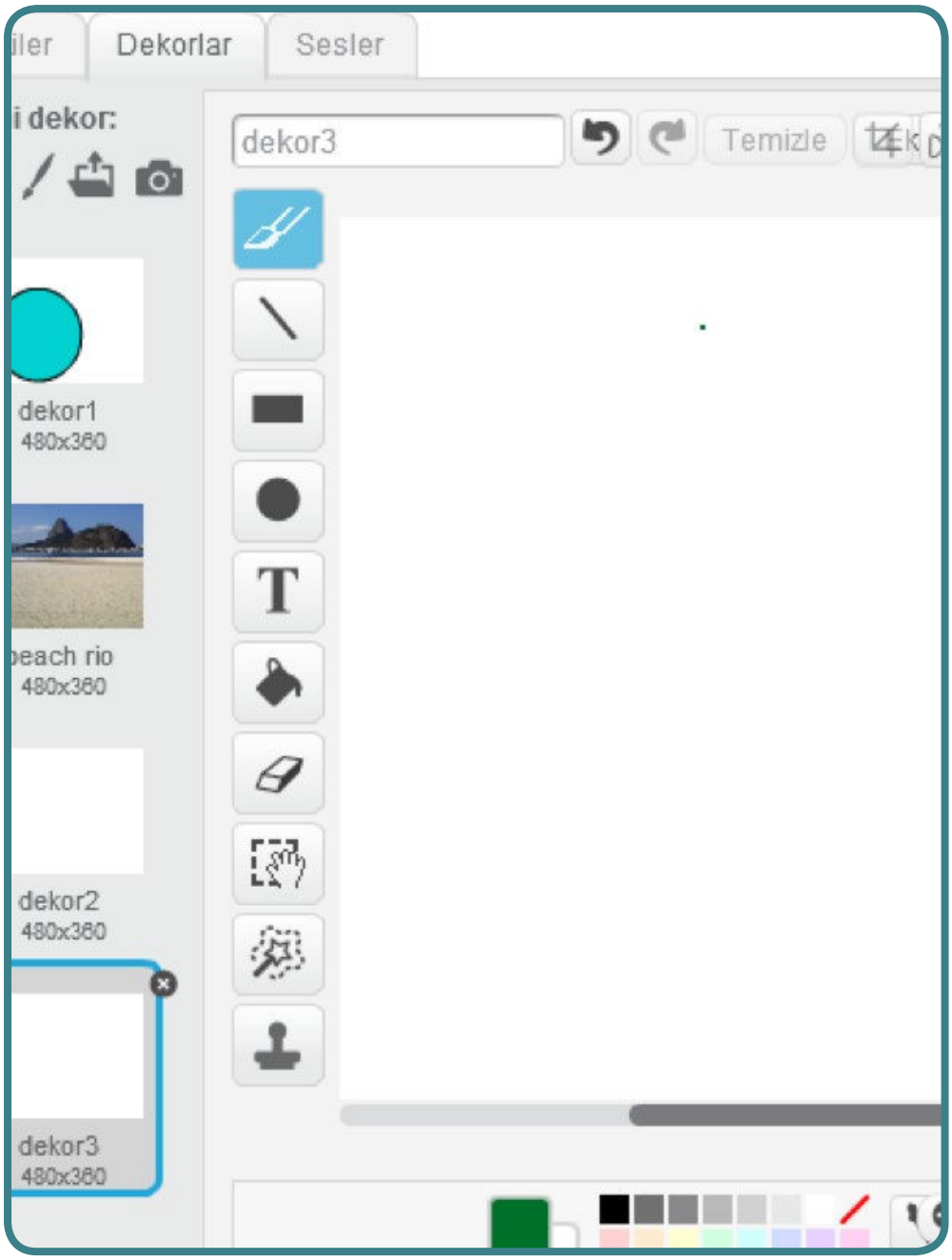
Şekil 25: Dekor ekleme butonları

butonuna basılarak kütüphaneden yeni dekor sahneye eklenebilir. Şekil 26'da kütüphaneden dekor seçimi gösterilmektedir.



Şekil 26: Dekor kütüphanesinden dekor seçimi

 butonu kullanılarak yeni dekoru kendimiz çizebileceğimiz dekor çizim alanı açılacaktır. Bu çizim alanı ve araçları daha önceden gördüğümüz kukla çizim alanı ile aynı özelliklere sahiptir. Şekil 27'de dekor çizim alanı gösterilmektedir.



Şekil 27: Dekor çizim alanı

 butonunu kullanarak bilgisayarda bulunan resimleri dekor olarak kullanabiliriz.

 butonunu kullanarak ise bilgisayara bağlı kameradan fotoğraf çekerek dekor olarak kullanabiliriz.

4.1.7 DİZİLER

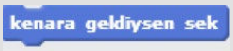
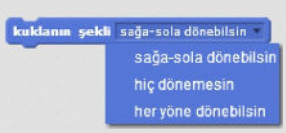
Diziler sahne ve kuklalar için scratch tarafından sunulan kod bloklarıdır. Sahne için Hareket kategorisi hariç diğer tüm kod blokları kullanılabilir. Sahne yani arka plan dekorlarının hareket özelliği bulunmadığı için hareket kod blokları, sahne seçili iken pasif durumdadır.

Not: Bir kukla veya sahne için kod yazmak istediğimizde istenilen nesnenin seçili olması gerekir.

Diziler sekmesi altında bulunan kategoriler şunlardır: Hareket, Görünüm, Ses, Kalem, Veri, Olaylar, Kontrol, Algılama, İşlemler, Özel Taşlar.

4.1.7.1 Hareket Kategorisi Kod Blokları

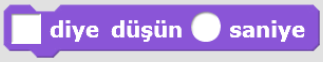
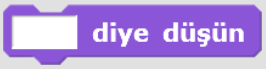
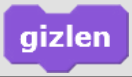
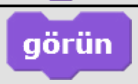
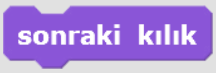
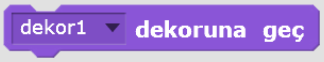
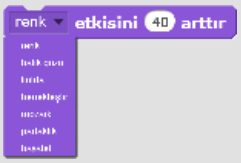
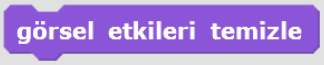
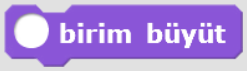
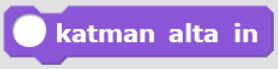
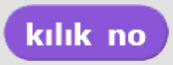
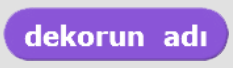
Adından da anlaşılacağı üzere hareket kod blokları sahnedeki kuklanın hareket etmesini sağlayan kod bloklarını içerir. Tablo 4'de hareket kod blokları sırasıyla açıklanmıştır.

KOD BLOĞU	ÖZELLİKLERİ
	Kuklanın yönüne göre belirtilen adım sayısı kadar hareket etmesini sağlar.
	Kuklanın belirtilen açı değeri kadar sağa dönmesini sağlar.
	Kuklanın belirtilen açı değeri kadar sola dönmesini sağlar.
	Kuklanın sabit açı değeri ile sağa, sola yukarı veya aşağı dönmesini sağlar. Bu kod bloğu kuklanın özellikler penceresindeki "dönüş izinleri" seçimine bağlı olarak çalışır.
	Sahnedeki kuklanın fare okuna veya sahnedeki diğer kuklalara doğru dönmesini sağlar.
	Kuklanın X ve Y ekseninde belirtilen koordinata gitmesini sağlar. Git komutu kuklayı belirtilen noktaya ışınlar.
	Kuklanın fare okuna veya sahnedeki diğer kuklaların bulunduğu koordinata gitmesini sağlar.
	Kuklanın belirtilen sürede, belirtilen koordinata süzülmesini sağlar. Süzül komutu kuklanın sahnedeki hareketinin, kullanıcı tarafından görülebilir kılar.
	Kuklanın sahnenin Yatay ekseninde sağ tarafa doğru hareket etmesini sağlar. Kuklanın sahnenin sol tarafına doğru gitmesi için ise x değeri -10 arttırılır. şeklinde de kullanılabilir.
	Kuklanın X eksenindeki yerini sabit bir noktaya götürmek için kullanılır.
	Kuklanın sahnenin Dikey ekseninde yukarı doğru hareket etmesini sağlar. Kuklanın sahnenin alt tarafına doğru hareket etmesi için ise Y değeri -10 arttırılır. şeklinde de kullanılabilir.
	Kuklanın Y eksenindeki yerini sabit bir noktaya götürmek için kullanılır.
	Kukla sahnenin kenarına geldiğinde sekerek sahne içine dönmesini sağlar.
	Kuklanın özellikler penceresinden değiştirebildiğimiz dönüş izinlerini kod bloğu yardımıyla değiştirmemizi sağlar.
	Kuklanın sahnedeki mevcut X konumunun yerini görüntüleyen değişkendir. Sahnede seçili olan kuklanın X eksenindeki yerini sahnede göstermek için yandaki kutucuk tıklanır.
	Kuklanın sahnedeki mevcut Y eksenindeki konumunun yerini görüntüleyen değişkendir. Sahnede seçili olan kuklanın Y eksenindeki yerini sahnede göstermek için yandaki kutucuk tıklanır.
	Seçili olan kuklanın yönünü belirtir.

Tablo 4: Hareket Kod Blokları

4.1.7.2: Görünüm Kategorisi Kod Blokları

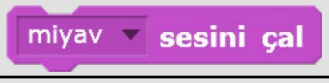
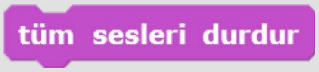
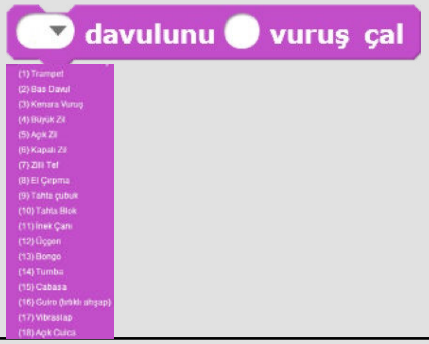
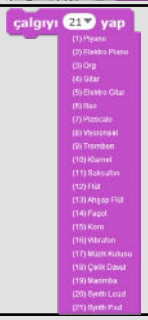
Görünüm kod blokları, sahnedeki kuklanın görünümüyle ilgili değişiklikleri yapmamızı sağlayan kod bloklarıdır. Tablo 5'de bu kod blokları gösterilmektedir.

KOD BLOĞU	Özellik
	Kuklanın 2 saniye boyunca Hello demesini sağlar.
	Kuklanın sürekli olarak Hello demesini sağlar.
	Kuklanın 2 saniye boyunca yazılan ifade diye düşünmesini sağlar.
	Kuklanın sürekli olarak belirtilen ifade ile düşünmesini ağlar
	Sahnedeki kuklanın gizlenmesini sağlar.
	Sahnedeki kuklanın görünmesini sağlar.
	Kuklanın kılığının belirtilen kılığa geçmesini sağlar.
	Kukla kılığının, mevcut kılıktan sonraki kılığa geçmesini sağlar.
	Projeye eklenen dekorlardan birine geçmeyi sağlar. Sonraki veya önceki dekora da isim belirtmende geçiş sağlanabilir.
	Kukla veya sahneye uygulanabilecek efektlerin değerini belirten kod bloğudur. Örneğin benekleştir efekti kedi kuklasınışekline getirebilmektedir.
	Seçilen efektin etkisini belirtilen değere ayarlar.
	Görsel etkileri temizlemek için kullanılır.
	Kuklanın belirtilen birim kadar büyütülmesini sağlar. – ifade ile bir değer yazılırsa kuklayı küçültür.
	Kuklanın büyüklüğünü % olarak belirtilen değere ayarlar.
	Kuklanın tüm kuklaların üstüne çıkmasını sağlar.
	Kuklanın belirtilen değer kadar alt katmana inmesini sağlar.
	Kuklanın hangi kılığının seçili olduğunu gösteren değişkendir.
	Seçili olan dekorun adını gösteren değişkendir.
	Kuklanın büyüklüğünü gösteren değişkendir.

Tablo 5: Görünüm Kod Blokları

4.1.7.3: Ses Kategorisi Kod Blokları

Uygulama geliştirirken ses efektleri çalışmanızın daha etkili olmasını sağlar. Tablo 6'da ses kod blokları gösterilmektedir.

KOD BLOĞU	ÖZELLİK
	Kuklaya ait sesi çalmayı sağlar.
	Belirtilen sesi bitene kadar çalar.
	Çalan tüm sesleri durdurur.
	Listeden seçilen çalgıları belirtilen vuruş kadar çalmayı sağlar.
	Belirtilen vuruş kadar susmasını sağlar.
	Seçilen notayı belirtilen vuruş kadar çalmayı sağlar.
	Çalgı aletini seçmeyi sağlar.
	Çalınan çalgının sesini yükseltmek veya azaltmak için kullanılır.
	Ses şiddetini % olarak ayarlamayı sağlar.
	Mevcut ses şiddetini gösteren değişkendir.
	Tempoyu belirtilen değer kadar arttırır.
	Temponun dakikada kaç vuruş olacağını belirtir.
	Mevcut tempo değerini gösteren değişkendir.

Tablo 6: Ses kategorisi kod blokları

4.1.7.4: Kalem Kategorisi Kod Blokları

Sahne üzerinde kuklanın hareket ederken ardında kalem ile belirtilen renkte, kalınlıkta çizgi çizmesini sağlayan kod bloklarıdır. Tablo 7'de Kalem kategorisi kod blokları gösterilmiştir.

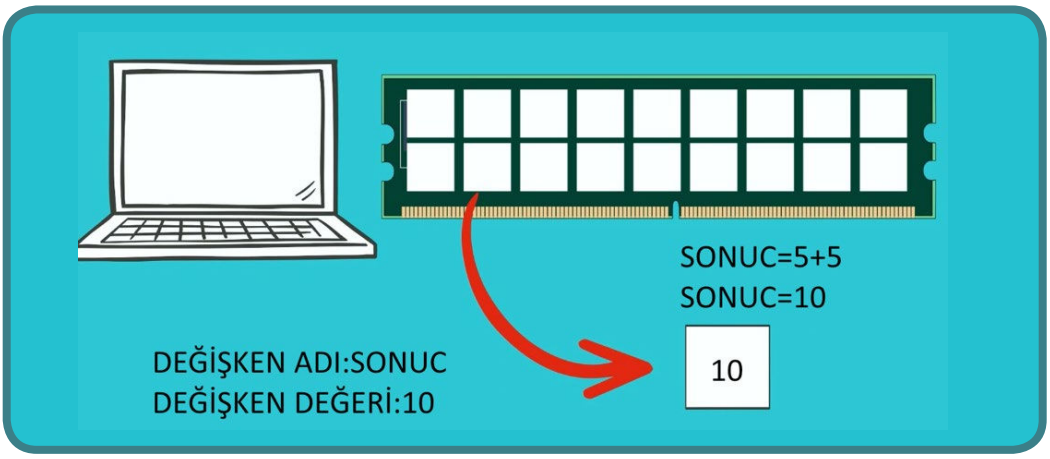
KOD BLOĞU	ÖZELLİK
temizle	Kalem kod blokları ile oluşturulan görsel etkilerin temizlenmesini sağlar.
iz bırak	Kuklanın görsel kopyasının kodun kullanıldığı koordinat noktasına bırakılmasını sağlar.
kalemi bastır	Kukla hareket ederken arkasında bir kalem ile çizgi çizilmesini sağlar. Çizgi kuklanın izlediği yolu takip eder.
kalemi kaldır	Kalemi bastır komutunun görevinin bittiği noktada kullanılarak kalem izi bırakılmasını engeller.
kalem rengini ☐ yap	Renk paletinden seçilecek renk ile kalemin hangi renkte iz bırakacağını belirtir.
kalem rengini ( ) arttır	Kalem renk değerini arttırarak farklı renkler elde edilmesini sağlar.
kalem rengini ( ) yap	Kalem rengini sabit bir değere ayarlar.
kalem tonunu ( ) arttır	Kalem renk tonunu belirtilen değer kadar arttırır.
kalem tonunu ( ) yap	Kalem renk tonunu sabit bir değere ayarlar.
kalem kalınlığını ( ) arttır	Kalemin çizim kalınlığını belirtilen değer kadar arttırır.
kalem kalınlığını ( ) yap	Kalemin kalınlığını sabit bir değere ayarlar.

Tablo 7: Kalem kategorisi kod blokları

4.1.7.5 Veri Kategorisi Kod Blokları

Veri kategorisinde iki temel blok bulunmaktadır. Bunlar "Bir değişken oluştur" ve "Bir Liste Oluştur" kod bloklarıdır. Bu kod bloklarını incelemek için değişken ve liste kavramlarını inceleyelim.

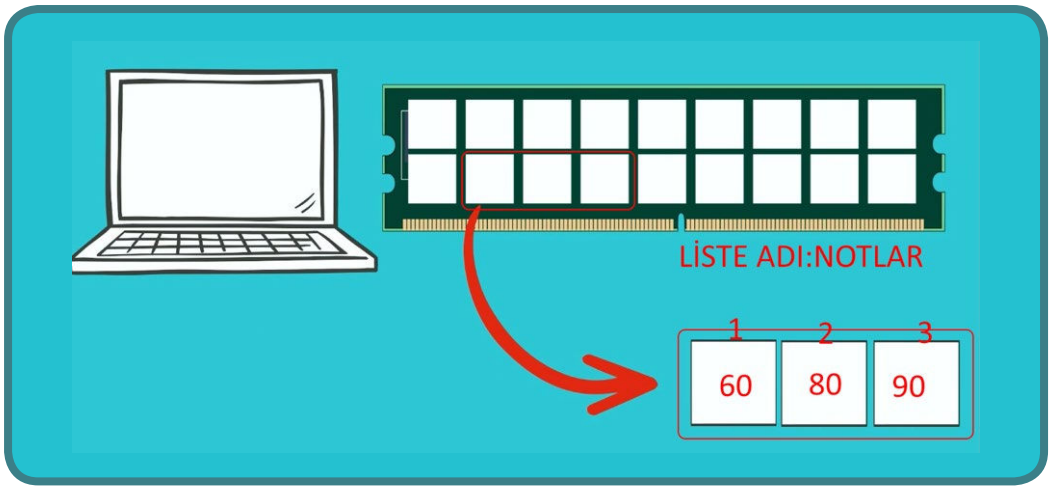
Değişken (Variable): Program yazarken kullanıcıdan alınan bilgiler, hesaplama sonrası ortaya çıkan sonuç değerleri, bir kuklanın koordinat sistemindeki yeri gibi birçok bilgiyi geçici olarak bilgisayarın belleğinde saklamak gerekir. Bunun için değişkenler kullanılır.



Şekil 28: Bellekte saklanan değişkenlerin gösterimi

Şekil 28'de görüleceği üzere, bir değişken oluşturduğumuzda bilgisayarın belleğinde değişken için bir alan ayrılır. Bu alana ulaşmak için değişkenin ismini kullanırız. Bu değişkenin adını kullandığımız her yerde değişkenin değeri geçerlidir. Bir değişkene değer atamak için "=" operatörünü kullanırız. Örnek olarak sonuç değişkenine değer atamak için sonuc=5+5 ifadesi kullanılmıştır. Bellekteki sonuc isimli kutucuğa (alana) 10 değeri aktarılır. Bir sonraki işlemde "sonuc=sonuc+10" şeklinde bir atama yapsaydık sonuc değişkinin son değeri 20 olacaktır. Bellekteki kutucukta artık 20 değeri tutulmaktadır.

Liste: Programlama dillerinde dizi(array) olarak bilinen listeler, birden fazla değişkenin tekbir isim altında birçok kutucuğa sahip değişken olarak ifade edebiliriz. Şekil 29'da "Notlar" adında oluşturulmuş bir listenin bellekteki yerleşimi gösterilmektedir.



Şekil 29: Notlar adında oluşturulan listenin bellekteki yerleşimi

Notlar listesine 3 eleman eklenmiştir. Bu elemanların değerleri sırasıyla 60, 80 ve 90 olarak verilmiştir. Notlar dizisinin 1. elemanını uygulamamızda kullandığımızda 60 değerini ifade edecektir. İstersek bu elemanın değerini değişkenlerde olduğu gibi değiştirebiliriz.

Değişken Oluşturma:
Scratch programlama dilinde bir değişken oluşturmak istediğimizde Veri kategorisinden "Bir Değişken Oluştur" butonuna tıklarız. Şekil 30'da gösterilen pencere açılır. Bu pencerede değişkenin tüm kuklalar için mi yoksa seçili olan kukla için mi oluşturulacağı belirtilir.

Şekil 30: Yeni değişken oluşturma penceresi

Değişkenin tüm kuklalar için seçilmesi durumunda tüm kuklalar oluşturduğumuz değişkene erişebilecektir. Sadece seçili kukla için oluşturursak diğer kuklalar için yazacağımız kodlarda bu değişkene erişemeyiz. Pencerede değişkenin adı kısmına gireceğimiz ifade ile aynı isimde bir daha değişken oluşturamayız. Bir değişken oluşturduğumuzda kullanabileceğimiz kod blokları Tablo 8'de gösterilmiştir.

KOD BLOĞU	ÖZELLİK
SONUC , 0 olsun	Değişkene belirtilen değeri aktarır.
SONUC 'i 1 arttır	Değişkenin değerini belirtilen değer kadar arttırır. Değerini azaltmak için – değer kullanmak gerekir.
SONUC değişkenini göster	Değişkeni sahnede göstermek için kullanılır.
SONUC değişkenini gizle	Sahnede gösterilen değişkeni gizlemek için kullanılır.

Tablo 8: Değişkenler için kod blokları

Liste Oluşturma:
Liste oluşturmak için Veri kategorisi altında Bir liste oluştur butonuna basılır. Ardından Şekil 31'de ki pencere açılır. Değişken oluşturma sırasındaki adımların aynısı liste için de geçerlidir. Oluşturulacak liste tüm kuklalar için seçildiğinde tüm kuklalar listeye erişebilir. Sadece seçili olan kukla için liste oluşturulduğunda ise sadece o kukla listeye ulaşabilecektir.

Şekil 31: Liste oluşturma penceresi

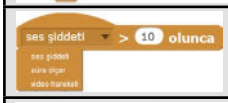
Tablo 9'da oluşturacağımız listeler için kullanılabilecek kod blokları gösterilmiştir.

KOD BLOĞU	ÖZELLİKLER
thing i notlar e ekle	Seçilen listeye belirtilen ifadeyi ekler
sil 1 sırasındakini notlar in	Seçilen listenin belirtilen sırasındaki elemanını siler.
koy thing 'yi, 1 sırasına notlar 'nin	Seçilen listenin belirtilen sırasına yazılan değeri ekler.
değiştir 1 sırasındakini notlar listesinin thing ile	Seçilen listenin belirtilen sırasındaki değeri yazılan değer ile değiştirir.
1 sırasındaki notlar 'in	Seçilen listenin belirtilen sırasındaki değeri gösterir.
notlar in uzunluğu	Seçilen listenin kaç elemana sahip olduğunu gösterir.
notlar listesinde thing var (mı?)	Seçilen listenin elemanları içerisinde belirtilen ifadenin olup olmadığı sorgulanır.
notlar listesini göster	Seçilen listeyi sahnede gösterir.
notlar listesini gizle	Sahnede gösterilen listeyi gizler.

Tablo 9: Liste kod blokları

4.1.7.6 Olaylar Kategorisi Kod Blokları

Bir uygulamanın çalışmaya başlayabilmesi için bir tetikleyiciye ihtiyacımız vardır. Uygulamanın başlangıcı için genellikle sahnenin sağ üst köşesindeki yeşil bayrağı kullanırız. Benzer şekilde uygulamayı sonlandırmak için sabit görevi bulunan kırmızı buton görevlendirilmiştir. Ancak yeşil bayrağın haricinde farklı seçeneklerimiz de mevcuttur. Tablo 10'da Olaylar kod blokları gösterilmektedir.

KOD BLOĞU	ÖZELLİK
	Bir kodun çalışmaya başlayabilmesi en sık kullanılan tetikleyicidir. Yeşil bayrağa basıldığında bu kod bloğuna eklediğimiz diğer bloklar çalışmaya başlar.
	Klavyeden basılacak herhangi bir tuşu tetikleyici olarak seçtiğimizde kullanacağımız kod bloğudur.
	Bir kuklaya tıklandığında çalışmasını istediğimiz kodları bu kod bloğunun altına ekleriz.
	Sahne dekorlarını yeri ve zamanı geldiğinde program içerisinde değiştirebiliriz. Dekor belirtilen bir dekor olduğunda çalışmasını istediğimiz kodları bu kod bloğunun altına ekleyebiliriz.
	Ses şiddeti, süre ölçer ve video hareketi değerlerinin belirli bir sayının üzerinde olduğunda çalışmasını istediğimiz kodları bu kod bloğunun altına ekleriz.
	"Haber1" haberi geldiğinde yapılmasını istediğimiz işlerin kodlarını bu kod bloğunun altına ekleriz.
	"Haber1" haberi tüm kuklalar ve dekorlar için çalışma ortamında yayımlanır. Herhangi bir kukla haber1 haberi geldiğinde yapacağı görev var ise görevini yerine getirir.
	"Haber1" haberini tüm kuklalara gönderir ve kuklanın kodu bitirmesini bekler.

Tablo 10: Olaylar Kod Blokları

Haber Sal Komutu: Kuklalar veya sahne dekorları bizim belirleyeceğimiz durumlar meydana geldiğinde buna cevap verebilmelidir. Örneğin bir oyun tasarlıyorsunuz ve oyun her 100 puan ve katlarına ulaştığında yeni bir seviyeye geçecek şekilde ilerlemesini istiyorsunuz. Böyle bir durumda her yeni seviye haberi için puan değişkeninin değeri sorgulanır ve yeni seviye haberi tüm kuklalara yayımlanır (salınır). Bu haberi alan kukla kendisi ile ilgili bir haber geldi ise gerekli komutları çalıştırır.

Şekil 32: Kuklaların haber geldiğinde tepkileri

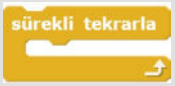
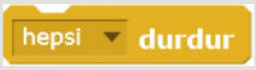
Yeni bir haber oluşturmak için Şekil 33'de gösterildiği üzere yeni haber sekmesine tıklanır, ardından haberin ismi yazılır. Burada dikkat edilmesi gereken nokta ise oluşturulan bir haberin kodlama alanına oluşturulduğu anda yerleştirilmesi gerekir.

Şekil 33: Yeni haber oluşturma

"Haber Sal" ve "Haber Sal ve Bekle" komutları arasındaki fark ise haber sal komutu ilgili haberi tüm kuklalara yayımlar ve diğer komutların çalışmasını beklemeden sürecin devam etmesini sağlar. Haber sal ve bekle komutu ise kuklalara yazılmış olan haber geldiğinde çalışacak komutun çalışmasını ve tamamlanmasını bekler.

4.1.7.7: Kontrol Kategorisi Kod Blokları

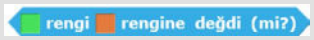
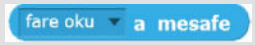
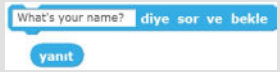
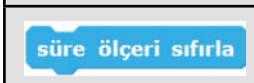
Kontrol kategorisinde bulunan kod blokları programın akışını belirli şartlara göre yönlendirebilir, tekrar eden görevleri yerine getirebilir, şart ifadesi meydana gelene kadar akışı bekletebilir. Tablo 11'de Kontrol kategorisi kod blokları gösterilmektedir.

KOD BLOĞU	ÖZELLİK
	Bir kuklaya yazılan kod akışının belirtilen süre kadar beklemesini sağlar. Bu esnada diğer kuklalara yazılan kodlar çalışmasını devam ettirir.
	Belirtilen sayı kadar, yazılan kodun tekrar etmesi sağlanır. Döngü blogudur.
	Bu kod bloğunun içerisine yazılan komutların sürekli olarak tekrar etmesi sağlanır. Sonsuz döngü olarak da bilinir. Uygulama durmadan döngüden çıkılamaz.
	"Eğer" kod bloğu bir şart ifadenin yerine gelip gelmediğini sorgular. Başka bir deyişle şart ifadenin sonucu "doğru" veya "yanlış" değer üretir. "Eğer" şart ifade yerine geliyor ise yani "doğru" değer üretiyorsa bu kod bloğunun içine yazılan kodlar çalıştırılır. Örnek şekilde bir şart ifadenin yerine gelmesi için sayac değerinin 10dan küçük olması gerekir. 
	Eğer şartı sağlanıyor yani "doğru" değer üretiyorsa eğer bloğu içerisine yazılan komutlar çalıştırılır. Şart ifade yerine sağlanmıyor yani "yanlış" değer üretiyorsa, değilse bloğu içindeki kodlar çalıştırılacaktır.
	Bir şart sağlanıncaya kadar program akışını o kukla için bekletir.
	Bir şart sağlanıncaya kadar tekrar edilmesi gereken komutları çalıştırır. While döngüsü olarak da bilinir.
	Tüm komutların çalışmasını, sadece eklendiği komut dizisini veya eklendiği kuklanın diğer komut dizilerini durdurmak amacıyla kullanılır.
	Bir kuklanın klonunu (ikizini) oluşturduğumuzda bu klonun yapacağı görevleri belirtmek için kullanılan başlangıç bloğudur.
	Sahnede bulunan diğer kuklaların veya mevcut kuklanın klonunu (ikizini) yaratmak için kullanılan kod bloğudur.
	Oluşturulmuş bir klon kuklanın silinmesini sağlar.

Tablo 11: Kontrol kategorisi kod blokları

4.1.7.8 Algılama Kategorisi Kod Blokları

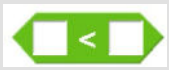
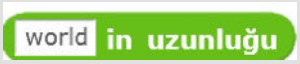
Algılama kod blokları sahnede bulunan kuklalar, fare imleci, klavye tuşları, ses şiddeti, video hareketi gibi birçok olayı algılamak için kullanılır. Kullanıcı ile etkileşimli uygulamalar geliştirmek için oldukça faydalı kod bloklarıdır. Genellikle bir şart ifadesi içerir. Fareye değdi mi? Tuşa Basıldı mı? gibi... Tablo 12'de algılama kod bloklarının görevleri verilmiştir.

KOD BLOĞU	ÖZELLİK
	Sahnedeki kuklaların birbirine, kenara veya fareye değip değmediğini sorgulamak için kullanılır. Eğer ki sorgulanan kukla seçilen nesnelerden birine değiyor ise "Doğru" sonuç üretilir.
	Sahnedeki kuklanın seçilen bir renge değip değmediği sorgulanır.
	Seçilen renkteki bir nesnenin başka renkte bir nesneye değip değmediği sorgulanır.
	Kuklanın fareye veya diğer kuklalara olan mesafesini sorgular
	Kullanıcı ile soru cevap şeklinde etkileşime girmek için kullanabileceğimiz bir komuttur. Kullanıcıya sahnede bir soru sorulur ve kullanıcının verdiği cevap yanıt değişkenine aktarılır.
	Klavyeden basılan tuşu sorgular.
	Farenin sol tuşuna basılı olup olmadığını sorgular.
	Farenin sahne üzerindeki X konumunu sorgular
	Farenin sahne üzerindeki Y konumunu sorgular
	Mikrofondan alınan ses şiddetini sorgular.
	Seçili kukla üzerindeki video hareketini sorgular.
	Webcam/Kameradan alınan görüntüyü açıp kapatır.
	Video saydamlığını %biriminden ayarlar.
	Süre ölçer değerini sorgular.
	Süre ölçeri sıfırlar.
	Sahnede bulunan diğer kuklaların X konumu, Y konumu, yönü, kılık numarası gibi bilgileri elde etmek için kullanılır.
	Bilgisayarınızdaki saat bilgisinin yıl, ay, gün, saat, dakika ve saniye bilgisini almak için kullanılır.
	2000 yılından itibaren geçen gün sayısını gösterir.
	Online editörde sisteme giriş yapan kullanıcının adını gösterir.

Tablo 12: Algılama kod blokları

4.1.7.9: İşlemler Kategorisi Kod Blokları

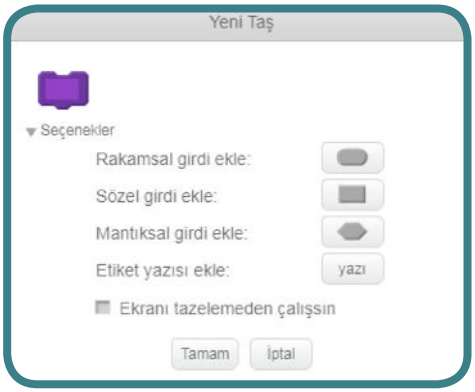
İşlemler kategorisinde matematiksel işlemler, şart ifadelerin sonuçları, metin türünden ifadeler gibi birçok işlemi yaparken faydalanacağımız kod blokları mevcuttur.Tablo 13'de işlemler kod blokları ve özellikleri gösterilmiştir.

KOD BLOĞU	ÖZELLİKLER
	İki sayının toplama işlemini yapar.
	İki sayının çıkarma işlemini yapar
	İki sayının çarpma işlemini yapar.
	İki sayının bölme işlemini yapar.
	Belirtilen iki sayı değeri arasında rastgele bir sayı oluşturur.
	İki ifade arasında soldaki ifadenin sağdaki ifadeden küçük olup olmadığını sorgular. Buraya girilecek değer sayısal ifadeler olabileceği gibi metin ifadeler de olabilir. a<b ifadesi doğru (true) sonuç üretecektir. c<b ifadesi ise (false) sonuç üretir.
	İki ifade arasında soldaki ifadenin sağdaki ifadeye eşit olup olmadığı sorgulanır. Buraya girilecek değer sayısal ifadeler olabileceği gibi metin ifadeler de olabilir.
	İki ifade arasında soldaki ifadenin sağdaki ifadeden büyük olup olmadığını sorgular. Buraya girilecek değer sayısal ifadeler olabileceği gibi metin ifadeler de olabilir. z>a ifadesi doğru (true) sonuç üretecektir. e>k ifadesi ise (false) sonuç üretir.
	VE operatörü iki şart ifadenin de doğru (true) sonuç üretmesini bekler. Her iki şart ifadede doğru(true) sonuç üretiyorsa VE operatörü de doğru sonuç üretir. Şart ifadelerden birisi yanlış(false) sonuç üretiyorsa VE operatörü de yanlış sonuç üretir.
	VEYA operatörü kendisine verilen şart ifadelerden birisi dahi doğru sonuç üretmesi durumunda doğru(true) sonuç üretir. Şart ifadelerden her ikisi de yanlış sonuç üretiyorsa VEYA operatörü de yanlış (false) sonuç üretir.
	Değil operatörü; bir şartın tersi değer üretir. Örneğin şart ifade olarak doğru(true) sonuç üreten bir ifade verilirse DEĞİL operatörü yanlış(false) sonuç üretecektir.
	İki metin ifadeyi birleştirmek amacıyla kullanılır.
	Verilen metin ifadenin belirtilen sıradaki harfini verir.
	Verilen ifadenin toplam karakter uzunluğunu belirtir.
	Soldaki sayının sağdaki sayıya göre modunu alır. Örneğin işleminin sonucu bildiğiniz üzere 4'dür.
	Verilen sayının karekök, sin, değer gibi birçok matematiksel fonksiyon sonucunu verir.

Tablo 13: İşlemler kategorisi kod blokları

4.1.7.10: Özel Taşlar Kategorisi Kod Blokları

Özel taşlar diğer programlama dillerinde kullandığımız fonksiyon/metotlar olarak da bilinir. Kendi taşınızı oluşturarak farklı görevler yapabilen yeni bloklar ortaya çıkarabilirsiniz. Bunun için Şekil 34'de gösterilen taş oluşturma penceresini kullanırız.



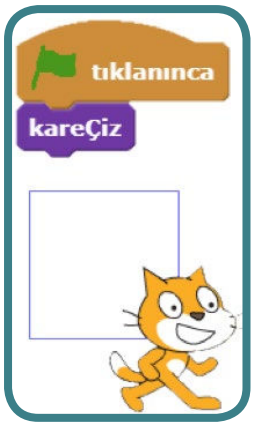
Şekil 34: Yeni taş oluştur penceresi

Şekilde görüleceği üzere bir taşı (fonksiyonu) tanımlamak üzere Yeni Taş penceresinde seçeneklerimiz bulunmaktadır. Taş oluşturmaktaki amacımız kod kalabalığını ortadan kaldırmak, aynı görevi yapan kodları tekrar tekrar yazmamaktır. Bir örnek ile taş oluşturmayı inceleyelim Örneğimizde kuklanın sahnede kare çizmesini sağlayacağız. Bu amaçla iki adet taş oluşturacağız. İlk oluşturacağımız kareÇiz taşında sabit 100 adımlık bir kare çizerken ikinci taşımızda biraz daha geliştirerek adım sayısı değişebilen bir kare çizmeyi inceleyeceğiz.
Bir taş oluştur butonuna tıkladıktan sonra açılan Yeni Taş penceresinde ilk sekmeye taşın ismini yazıyoruz:"kareÇiz".

Şekil 35: kareÇiz taş oluşturmak için taşın ismini ilk sekmeye yazıyoruz

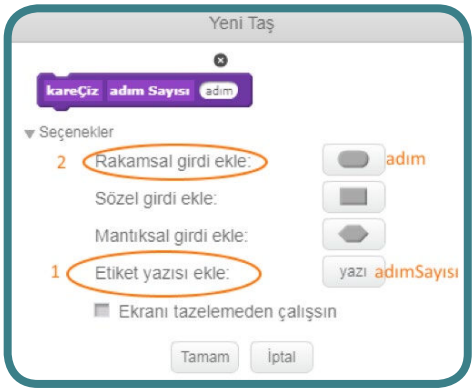
Şekil 36: Sabit kare çizecek olan kareÇiz taşı kodları

İkinci adımda ise, Şekil 36'da gösterilen kod bloklarını kareÇiz taşının tanımla bloğu altına ekliyoruz."kareÇiz taşını çağırdığımızda tanımlı olan kodlar çalışacaktır ve Şekil 37'de gösterilen karenin çizildiğini göreceksiniz.



Şekil 37: kareÇiz taşının kullanımı

İkinci oluşturacağımız kareÇiz metodunda ise adım sayını da kullanıcı belirleyecektir. Yeni bir taş oluştur butonuna tıkladıktan sonra bu sefer rakamsal girdi ekle sekmesine de tıklıyoruz. Şekil 35'de gösterildiği şekilde taşımızı oluşturuyoruz.



Şekil 38: Rakamsal girdiye sahip yeni kareÇiz taşı

kareÇiz taşına ekleyeceğimiz yeni kodlar ise Şekil 39'da gösterilmektedir. Burada kullanıcı kareÇiz taşını çağırırken kaç adımlık kare çizeceğini de sayısal girdi olarak belirtmektedir. Belirtilen bu değer adım değişkenine aktarılır ve bu adım değişkenindeki değer boyutlarında kare çizilir.

Şekil 39: kareÇiz taşı kod blokları

Şekil 40'da ise yeni kareÇiz taşının çalıştırılması gösterilmektedir. 50 adımlık bir kare çizmek istediğimizde taşı çağırırken 50 ifadesini girmemiz yeterli olacaktır.

Şekil 40: Belirtilen sayıda kare çizebilen yeni kareÇiz taşının çalışması

4.2: Çevrimiçi(Online) Editör

Scratch ile uygulama geliştirmek için diğer seçeneğimiz ise Web arayüzüdür. https://scratch.mit.edu/ adresinden online editöre erişebiliriz. Online editörün bize sunduğu en büyük kolaylıklardan birisi çalışmalarımızı, internete bağlı olan her yerden görebilir, değiştirebilir veya paylaşabiliriz. Ayrıca scratch 3 versiyonunda artık offline editör sunulmayacağı bundan sonraki süreçte online olarak kullanılması destekleneceği geliştiriciler tarafından açıklanmıştır.

4.2.1 Scratch Kayıt

https://scratch.mit.edu/ sitesine kayıt olabilmek için Scratch'a Katıl butonuna basarak kayıt ekranına ulaşabiliriz. Şekil41'de kayıt ekranı gösterilmektedir. Bu ekranda önce kullanıcı adı ve ardından şifre bilgisi girilir.

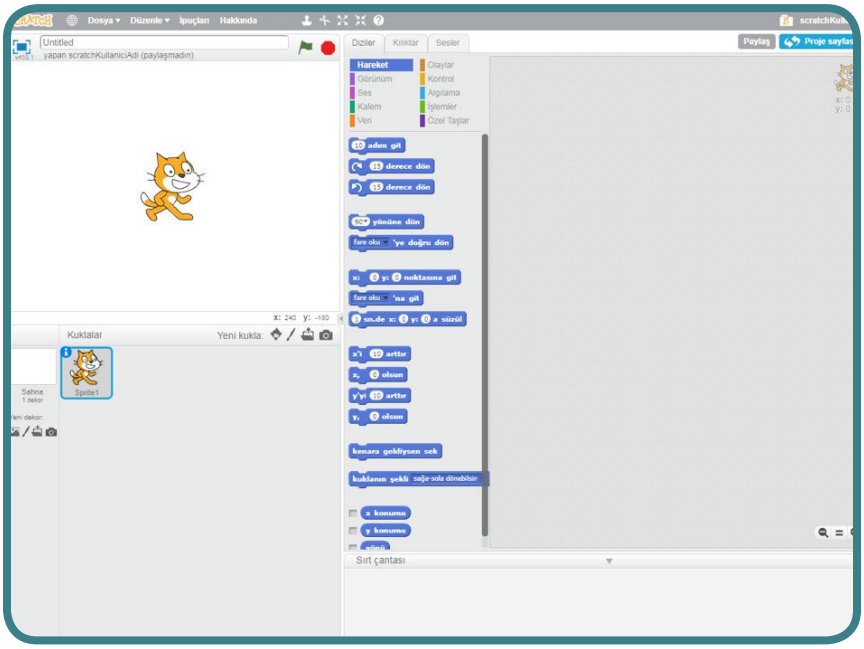
Şekil 41: Scratch kullanıcı kaydı, kullanıcı adı ve şifre belirleme

Şekil 42'de görüleceği üzere sonraki adımda doğum tarihi ve cinsiyet ve ülke bilgisi istenmektedir.

Şekil 42: Doğumtarihi, cinsiyet ve ülke bilgileri

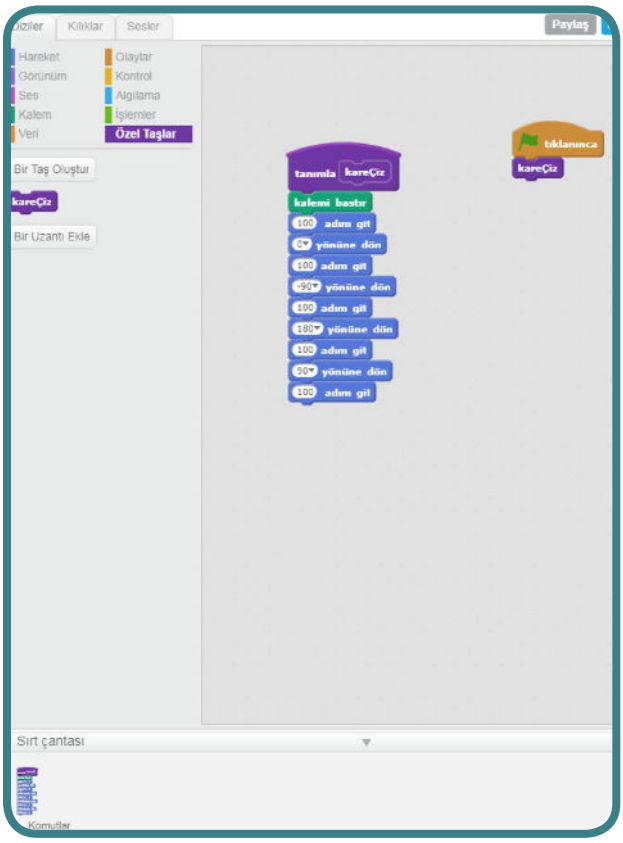
Son adımda ise aktif olarak kullandığınız bir email adresi istenmektedir. Bu adrese gönderilecek e-postada ki “eposta adresimi doğrula” linkine tıklamanız durumunda kullanıcı kaydınız tamamlanmış olacaktır.

Online editörün kod blokları offline editör ile aynıdır. Online editörde kodlarımızı saklayıp ve tekrar kullanabileceğimiz bir “sırt çantası” sunulmaktadır.



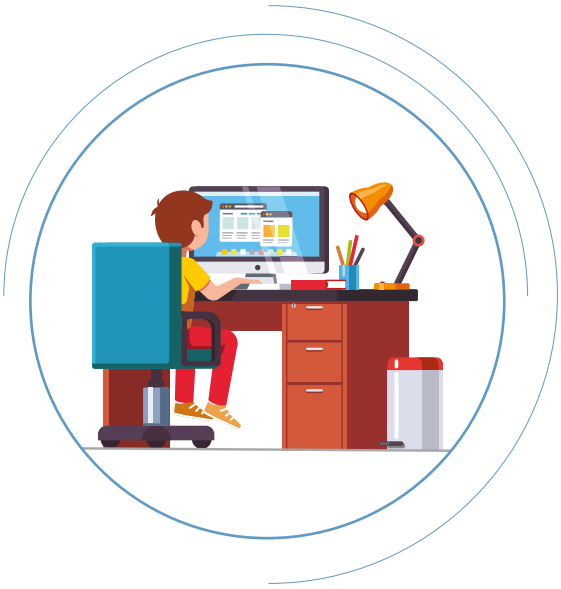
Şekil 43: Online editör

Sırt çantasına eklediğimiz kodları diğer projelerimizde de kullanabilmekteyiz.Şekil 44’de sırt çantasına eklenen kodlar gösterilmektedir.



Şekil 44: Sırt çantasına eklenen kodlar

BÖLÜM



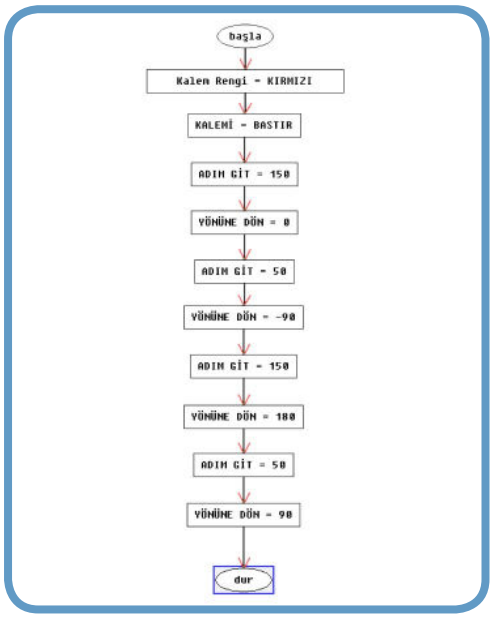
PROBLEM ÇÖZÜMÜNDE DOĞRUSAL MANTIK YAPISI KULLANIMI

BÖLÜM5: PROBLEM ÇÖZÜMÜNDE DOĞRUSAL MANTIK YAPISI KULLANIMI

Bu bölümde günlük hayatta sıklıkla kullandığımız, problem çözümünde doğrusal mantık yapısının kullanımını inceleyeceğiz. Doğrusal mantık yapısı; sınırları belirlenmiş bir görevin adımlarını birbiri ardına sıralayarak çözüme ulaşmayı amaçlar.

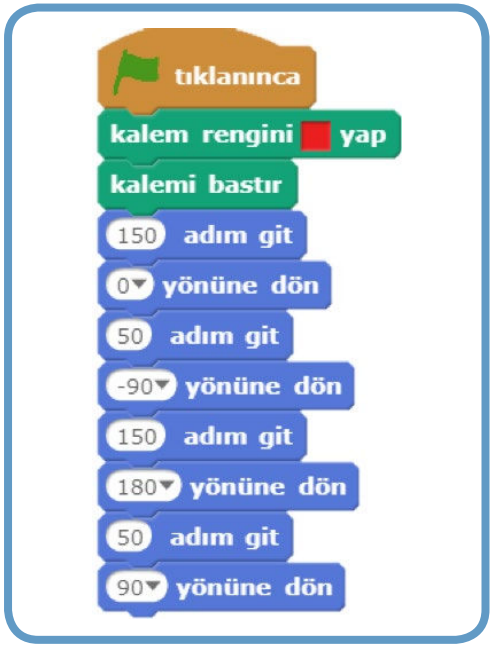
Etkinlik 4: Doğrusal mantık yapısını kullanarak sahnedeki kuklaya kırmızı renkte dikdörtgen şekli çizdirmek.

Etkinliğin amacı sahnede bulunan bir kuklaya sınırları belirlenmiş bir dikdörtgen şekli çizdirmektir. Dikdörtgenin boyutları eni 50 adım boyu ise 150 adım olarak belirlenmiştir. Kalem renginin kırmızı olması istenmektedir. Buna göre öncelikle algoritmayı hazırlayalım. Şekil 45'de adım adım yapılması gereken görevler doğrusal bir şekilde programa yazılmış ve sonlanmıştır.



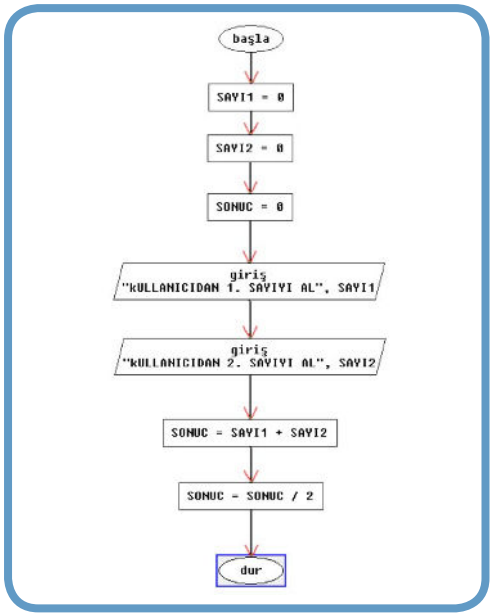
Şekil 45: Dikdörtgen çizim algoritması

Uygulamanın kodlarını Şekil 46'da görebilirsiniz. Öncelikle Kalem kategorisinden "kalem rengini yap" komutunu kullanarak kırmızı rengi seçiyoruz. Ardından "kalemi bastır" komutunu kullanarak kukla sahnede hareket ettikçe kalemin çizim yapmasını sağlarız. Sonraki adımlarda ise kuklanın yönünü adım sayısı ile birlikte dikdörtgen oluşturacak şekilde hareket ettiriyoruz. Kırmızı renkte çizilmiş eni 50 adım, boyu 150 adım olan bir dikdörtgeni çizmiş olduk.



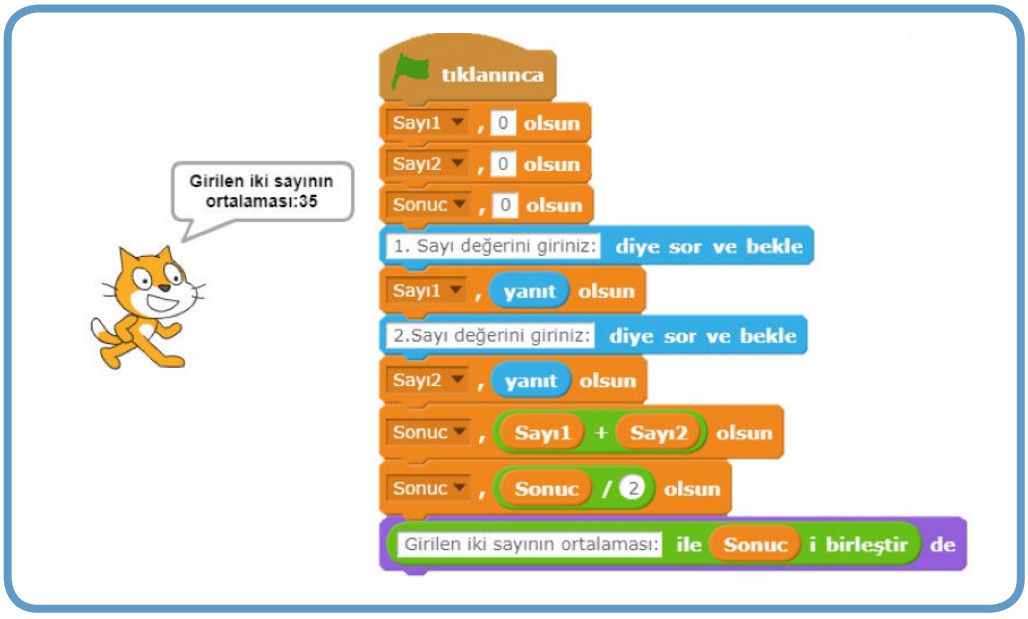
Şekil 46: Dikdörtgen çizimi için gerekli kod blokları

Etkinlik 5: Kullanıcıdan alınan iki sayının ortalamasını hesaplamak.
Bu etkinliğimizde kullanıcıdan alınacak iki sayının ortalamasını hesaplayarak kullanıcıya göstereceğiz. Öncelikle istenen iki sayı ve sonuç olmak üzere üç değişkene ihtiyacımız olacak. Bu sınırlar dahilinde hazırlanan algoritmayı Şekil 47'de görmekteyiz.



Şekil 47: Kullanıcıdan alınan iki sayının ortalamasını hesaplayan algoritma

Algoritmada görüleceği üzere üç değişken tanımlanmıştır. Bunlar Sayı1, Sayı2 ve Sonuc değişkenleridir. Doğrusal mantık ile problemi çözdüğümüzden iki sayının gerçekten sayısal karakter olarak kullanıcı tarafından girildiğini kabul ediyoruz. İki sayının toplamını sonuç değişkenine aktardıktan sonra sonuç değişkenin değerini 2'ye bölerek ortalamayı hesaplıyoruz. Şekil 48'de uygulamanın Scratch kodlarını görmekteyiz.



Şekil 48: Girilen iki sayının ortalamasını alan program. Kullanıcı 45 ve 25 sayılarını girerek 35 ortalama değeri hesaplanmıştır.

BÖLÜM

PROBLEM ÇÖZÜMÜNDE KARAR YAPISI KULLANIMI

BÖLÜM 6: PROBLEM ÇÖZÜMÜNDE KARAR YAPISI KULLANIMI

Gün içinde gerçekleştirdiğimiz eylemlerin hepsi bir karar sonucunda gerçekleşmektedir. Okula gitmek için seçilen yol, öğlen yemekte ne yeneceği, proje çalışmasında hangi konunun çalışılacağı gibi durumlar insanın kendi kararları ile ilgilidir. Bizler hayatın her alanında ve anında hayatımızın nasıl olacağına dair kararlar vermekteyiz. Bir kararı vermeden önce zihnimizde belirli süreçleri işletir, bir karara varır ve buna göre eyleme geçeriz. Örneğin öğle yemeğinde kantinden tost almak istediğimizi farz edelim. Kantine gittiğimizde tost olmadığını, simit olduğunu öğrendik. Bu durumda ihtiyacın ve kararların sonucunu değerlendirerek bir karar verilmesi gerekirdi. Verilecek karar ise değişiklik gösterebilirdi. Mesela tost yerine simit alabilir, başka bir yerden tost alabilir veya başka bir yerden başka bir şey yemeği tercih edebilir. Benzer durum bilgisayar programları içinde geçerlidir. Programın belirli durumlar ile karşılaştığında nasıl çalışacağın belirlenmesi için karar mekanizmaları kullanılır.

6.1: Şart İfadeleri

Program akışının bir noktasında verilen karara göre yapılacak işlem şekil değiştirebilir. Bir durum oluştuğunda, bir değişken belirli bir değeri aldığında veya klavyeden bir tuşa basılı olup – olmadığı durumlarına göre farklı kararlar verilebilir. Dolayısıyla program bu karara göre yeniden şekillenebilir. Örneğin, "Hava karanlık ise lambayı aç" ifadesinde, havanın durumunun kontrol edilmesi ve buna göre bir eylemin gerçekleştirilmesi yer almaktadır. Buna göre yazılan program havayı kontrol edecek eğer hava aydınlık ise lambayı kapalı tutacak, hava karanlık ise lambayı açarak ortamı aydınlatacaktır. Blok tabanlı veya metin tabanlı programlama dillerinin karar yapılarının temeli "şart ifadeleri"dir. Bu ifadeler genel olarak şöyle yazılır: "Eğer... durumu gerçekleşirse ...'yı yap." Bu ifade yazılmak istenen programa göre çok değişik durumlarda kullanılabilir. Örneğin "Eğer klavyede A tuşuna basılı ise sahnedeki kuklayı sola götür." veya "Eğer can değeri 0 ise oyunu durdur" gibi.

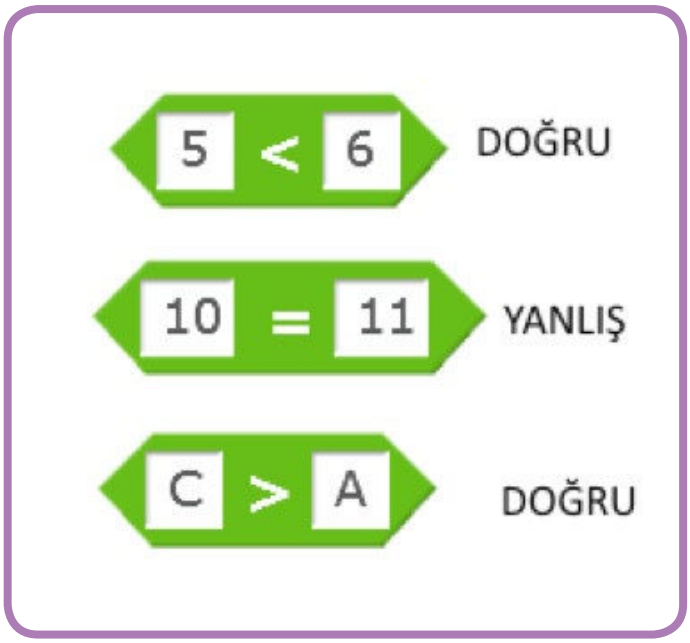
6.1.1- Cevap Olarak "Doğru" Ya Da "Yanlış" Sonucunu Veren Mantıksal İfadeler

Gece ile gündüz, soğuk ya da sıcak gibi ifadeler bir gözlemin sonucunda bizlerin yorumlarıdır. Net bir şekilde ifade edilemezse kişilere göre değişiklik gösterebilir. Bir odanın ısısı kimilerine göre sıcak gelirken, başka birilerine göre ılık veya soğuk gelebilir. Bu gibi net olmayan durumlarda şart ifadeleri kullanılamayacaktır. Matematikte olduğu gibi dijital sistemler de kesinlik içeren durumların programlanmasında "doğru" veya "yanlış" sonucunu veren mantıksal ifadeler kullanılır. Örneğin "Klavyeden B tuşuna basılı mı" sorunusun cevabı "evet doğru" veya "hayır yanlış" şeklinde olacaktır. Yine "Ekrandaki kedi kuklasına fare imleci değdi mi?" sorusunu cevabı "Doğru" veya "Yanlış" olarak dönecektir. Bu cevaplara göre programın akışı şekillenecek ve istenen sonuçlara göre programa yön verilmiş olacaktır. Şekil 49'da Algılama kod bloğu içinde yer alan mantıksal ifadelerin kod blokları gösterilmektedir.

Şekil 49: Cevap olarak "Doğru" ya da "Yanlış" sonucunu veren mantıksal ifadeler

6.1.2-İlişkisel Operatörleri Kullanarak "Doğru" ya da "Yanlış" Sonucunu Veren Mantıksal İfadeler

İlişkisel operatörler iki büyüklüğü, niceliği veya değeri karşılaştırmak için kullanılır. İki veri arasında kıyaslama yapılacaksa ikisinin de aynı türden olması gerekmektedir. İki büyüklük veya değer kıyaslanacağı zaman ortak özellikleri üzerinden bir kıyaslama yapılması gerekir. Örneğin iki nesnesin fiyatı, ağırlığı, boyutu kıyaslanabilirken göreceli olan tadı için bir kıyaslama yapılamaz. Dijital sistemlerde de sayısal bir ifade ile sayısal bir ifadeyi büyüklük/küçüklük ölçeğinde kıyaslanabilir. Metinsel bir ifadeyi sayısal bir ifade ile karşılaştırmak da birbirinden farklı nesneleri karşılaştırmaya benzetilebilir. Bu durumda, metinsel bir ifa¬denin karakter sayısı, sayısal bir ifade ile karşılaştırılır. Şekil 50'de İlişkisel operatörler ve bunların karşılaştırılması sonucu elde edilen çıktılar gösterilmektedir.



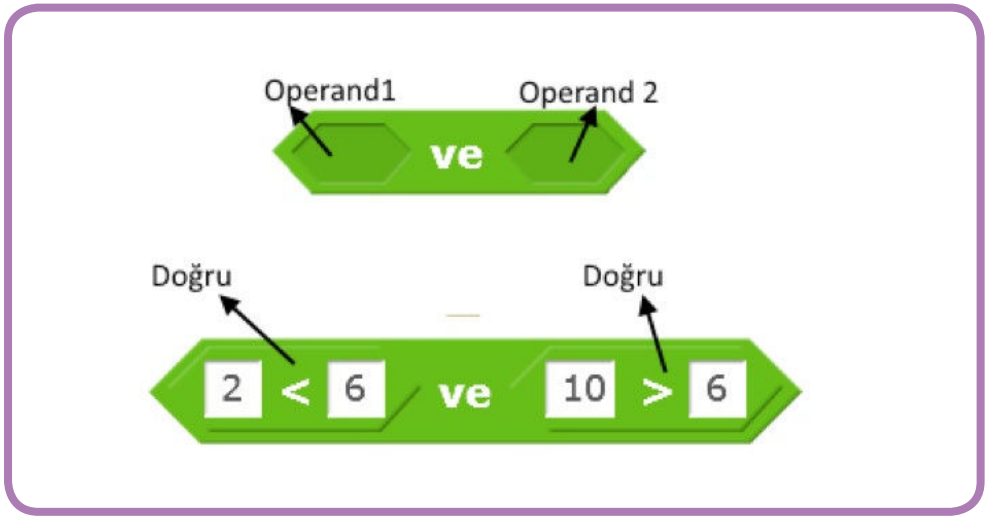
Şekil 50: İlişkisel operatörlerin karşılaştırma sonuçlarının gösterilmesi

Şekilde ilk sırada gösterilen "Küçüktür" operatörü ile "5 sayısının 6'dan küçük olup olmadığı" sorgulanmaktadır. Bu sorgulamanın sonucu 5 sayısı 6'dan küçük olduğu için "doğru" sonucunu üretecektir. İkinci operatör olan "eşittir" operatörü ile "10 sayısının 11 sayısına eşit olup olmadığı" sorgulanmaktadır. 10 sayısı 11 sayısına eşit olmadığı için operatör "Yanlış" sonucunu üretecektir. Son olarak "büyüktür" operatörü ile "C" harfinin "A" harfinden büyük olup olmadığı sorgulanmaktadır. Burada harflerin alfabedeki sıralamalarına bakılacaktır. C harfi alfabede 3. Sırada, A harfi ise birinci sırada yer aldığı için sorgulama sonucunda operatör "Doğru" sonucunu üretecektir.

6.1.3-Mantıksal İfadeler VE – VEYA- DEĞİL Kullanımı

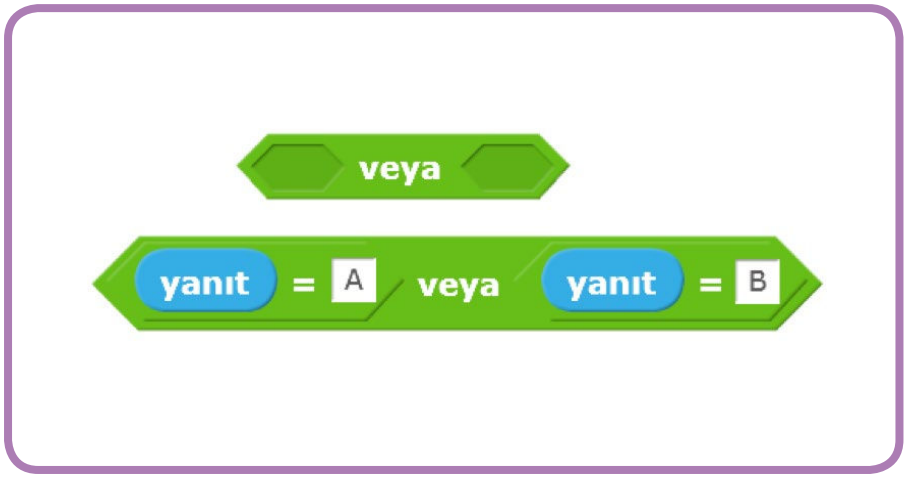
Aynı anda birden fazla durumu içeren ifadeleri sorgulamak için "ve", "veya" ve " değil" mantıksal operatörleri kullanılmaktadır.

1- VE Mantıksal Operatörü: "VE" mantıksal operatörü, iki şartın aynı anda yerine getirilip getirilmediğini sorgulamak için kullanılmaktadır. Şekil 51'de "Ve" operatörünün kullanım örnekleri bulunmaktadır. Örnekte yer alan "Ve" operatörünün sağında ve solunda yer alan ifadelere "Operand" adı verilmektedir. "Operand1" ve "Operand2" "Doğru" sonuç üretmesi durumunda "VE" operatörü de "Doğru" sonuç üretecektir. "Operand1" ve "Operand2"'nin "Yanlış" sonuç üretmesi durumunda "VE" operatörü de "Yanlış" sonuç üretecektir. Her iki operandın da "Yanlış" sonuç üretmesi durumunda da "VE" operatörü "Yanlış" sonuç üretecektir. Örnekte verilen "Ve" operatörü sonuç olarak "Doğru" ifadesi üretmektedir



Şekil 51: VE operatörünün kullanımı

2-VEYA Operatörü: "VEYA" operatörüne bağlı iki şarttan birisi "Doğru" sonuç ürettiğinde operatör "Doğru" sonuç üretecektir. Operandların her ikisi de "Doğru" sonuç ürettiğinde de "VEYA" operatörü yine "Doğru" sonucu üretmektedir. Operandların ikisi de "Yanlış" sonucunu ürettiğinde "VEYA" operatörü "Yanlış" sonucunu üretmektedir. Şekil 52'de "VEYA" operatörünün kullanımı gösterilmektedir.



Şekil 52: VEYA operatörünün kullanımı

Kullanıcıya sorulan bir sorunun cevabının A veya B olması durumunda her iki cevabında kabul edildği bir programın yazımında "VEYA" operatörü kullanılabilir. Böylece kullanıcının vermiş olduğu cevap "A" veya "B" olsun "VEYA" operatörü "Doğru" sonucunu üretecektir.

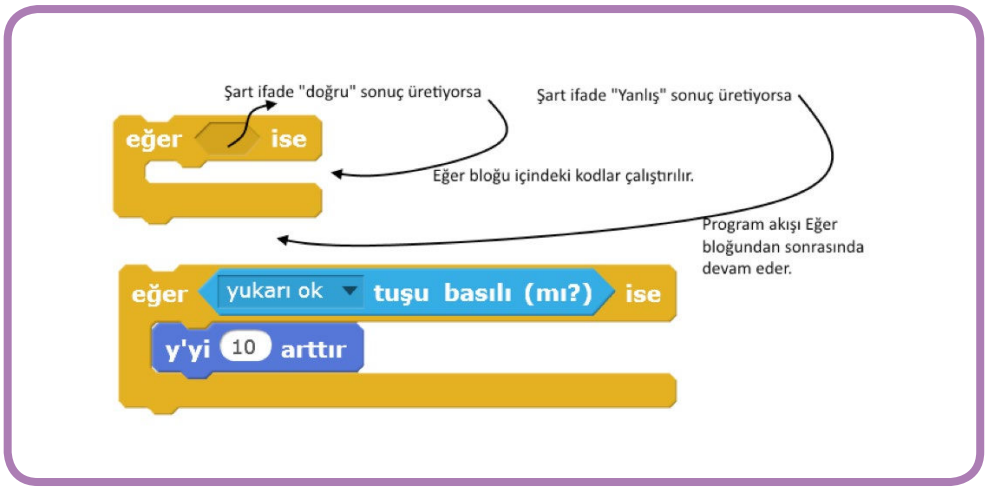
3-DEĞİL operatörü: Değil operatörü bir ifadenin mantıksal olarak tersi sonucunu elde etmek için kullanılmaktadır. Örneğin bir oyunda toplanan elma sayısı 10 olmadığı sürece gizli bir kapı açılmayacaksa, elma sayısının 10'a eşit olmaması sorgulanır. Şekil 53'de"Değil" operatörünün kullanımı gösterilmektedir. Buna göre; Değişken değeri 10 sayısına eşit değil ise "DEĞİL" operatörü "Doğru" sonucunu üretecektir. ElmaSayısı değişkeni 10'a eşit olduğunda ise "DEĞİL" opreatörü "Yanlış" sonucunu üretecektir.

Şekil 53: Değil operatörü kullanımı

Değişken değeri 10 sayısına eşit değil ise DEĞİL operatörü "Doğru" sonucu üretecektir. ElmaSayısı değişkeni 10'a eşit olduğunda ise DEĞİL opreatörü "yanlış" sonucunu üretecektir.

6.2: EĞER... Şart İfadesi

Bir şartın yerine gelip gelmediğini sorgulayarak o durum hakkında bilgi sahibi olunabilir. Bunun için "Eğer..." kontrol yapısı ile şart ifadesinin yerine gelip gelmediği sorgulanır. Şart ifadesi "Doğru" sonuç üreti¬yorsa "eğer " bloğu içeresindeki kodlar çalıştırılır. Şart ifadesi "Yanlış" sonuç üretiyorsa program akışı eğer bloğundan sonrasında devam eder. Şekil 54'de örnek bir eğer bloğunun kullanımı gösterilmektedir. Bu örnekte klavyeden yukarı ok tuşuna basılı olup olmadığı sorgulanmaktadır. Eğer yukarı ok tuşuna basılı ise kuklanın Y ekseninde yukarı doğru hareket etmesi istenmektedir.



Şekil 54: Eğer kod bloğunun kullanımı

6.3: Eğer Değilse Şart İfadesi

""Eğer " şart ifadesi bir sorgulama sonucunda "Doğru" sonuç üretiliyorsa kod bloğu içerisindeki komutları çalıştırmaktadır. Şart yerine gelmiyorsa ya da sorgulama sonucu "Yanlış" değerini üretiyorsa ana programın akışından bağımsız bir görevi yerine getirmek gerekebilir. Bu durumda "Eğer...Değilse..." şart ifadesi kullanılabilir. Örneğin kullanıcıdan alınan bir sayının çift sayı olup olmadığını sorgulamak istendiğinde, sayının mod'u alınır. Kullanıcının girdiği sayı çift ise kuklaya "Sayı Çift", sayı tek ise "Sayı Tek" ifadesini söyletilebilir. Bu amaçla Şekil 55'de ki kod blokları yazılabilir.

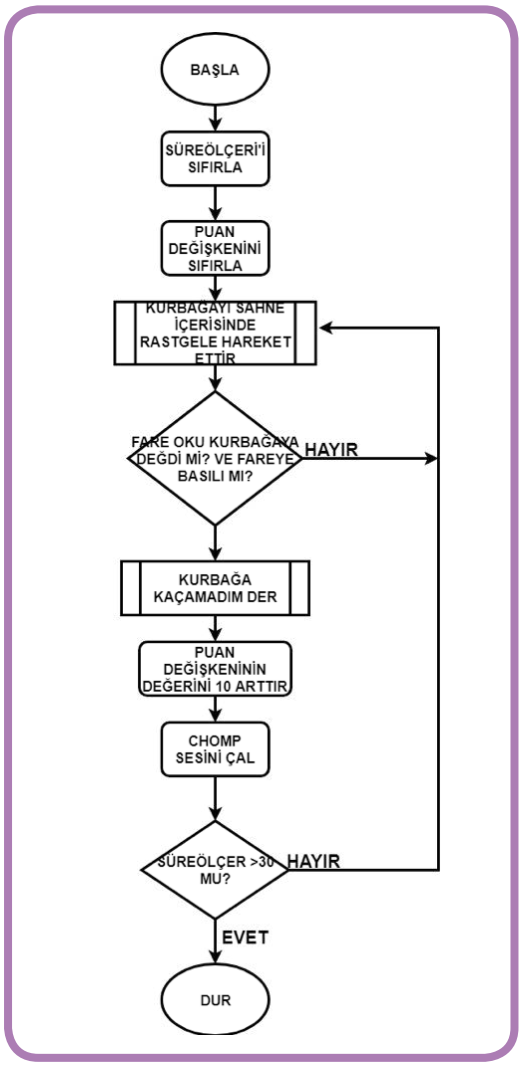
Şekil 55: Eğer... Değilse ... şart ifadesinin kullanımı

Kod bloğunda kullanıcının bir sayı girmesi istenmektedir. Kullanıcının sahnedeki text alanına gireceği her değer yanıt değişkenine aktarılır. Yanıt değişkeninin, 2'ye göre mod'u alındığında sonuç 0 ise sayı çift, sonuç 0 değil ise sayı tektir. Buna göre kodlarımızı çalıştırıp text alanına 10 değerini girdiğimizde Şekil 56'da gösterildiği gibi kukla; "Sayı Çifttir" ifadesini ekranda gösterecektir.

Şekil 56: Kullanıcının girmiş olduğu sayının tek mi çift mi olduğunun gösterilmesi

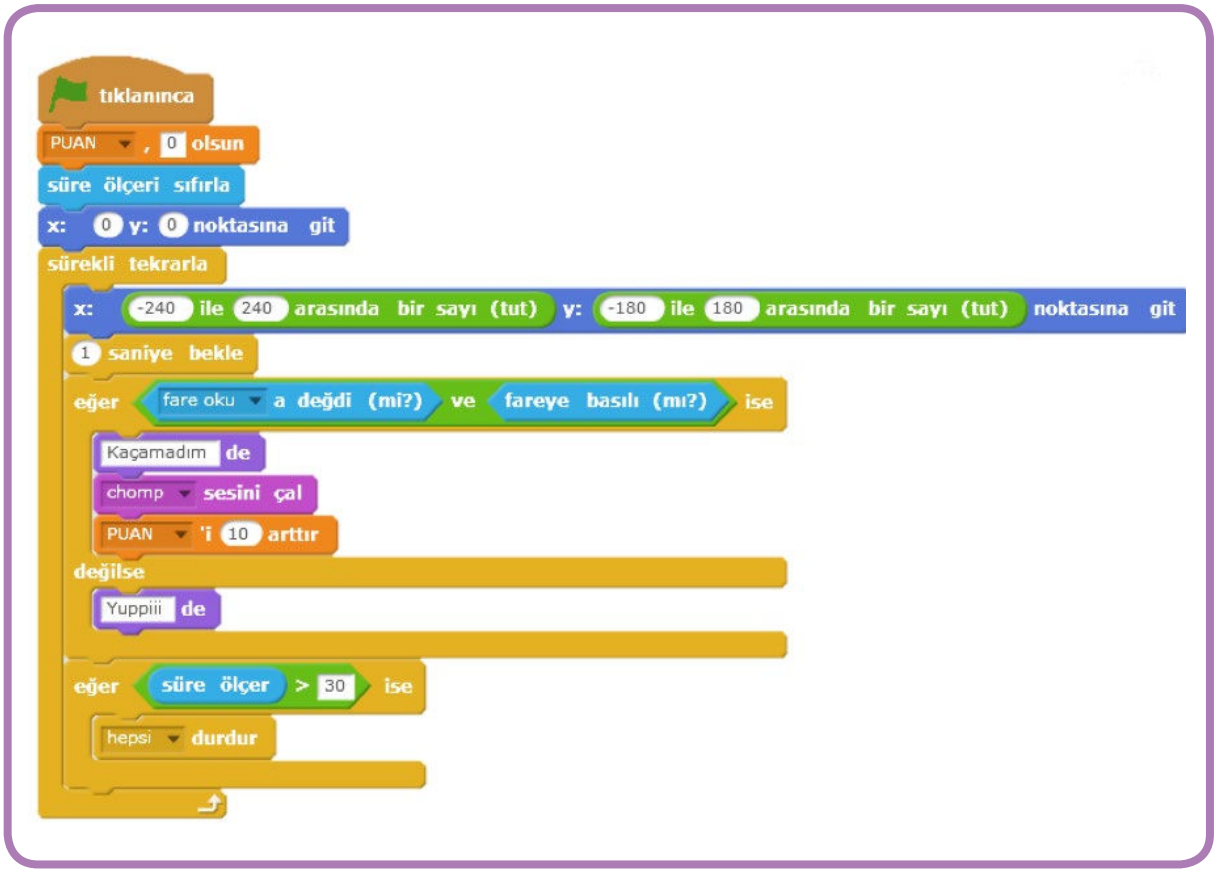
Etkinlik 6: Kurbağa Yakalama Oyunu

Bu etkinliğimizde Scract ile temel bir oyun tasarımı yapılmaktadır. Oyunun tasarlamadan önce problemin sınırları belirlenmelidir. Oyunun amacı, sahnede rastgele notalara hareket eden bir kurbağa kuklasını fare ile yakalamaya çalışılmasıdır. Kurbağanın yakalanmış ol¬ması için farenin sol tuşuna basılı iken sahnedeki kuklaya dokunuyor olması gerekmektedir Bu şartları yerine getiren kullanıcı 10 puan alacaktır ve 1 dakikalık süre içerisinde en çok kurbağa yakalayan oyuncu oyunu kazanacaktır. Bu oyun için hazırlanan algoritma Şekil 57'de gösterilmektedir.



Şekil 57: Kurbağa oyunu algoritması

Şekil 58'de uygulama kodları gösterilmektedir.



Şekil 58: Kurbağa yakalama oyunu

Kod bloklarında, farenin hem tıklanması hem de kurbağaya değiyor olması durumları sorgulanmaktadır. Eğer bu şartlar gerçek¬leşiyorsa Kurbağa kulası, "kaçamadım" yazısını gösterecek, "chomp" sesini çalacaktır. Aynı anda "Puan" değişkeninin değeri 10 artacaktır. Eğer şart gerçekleşmiyorsa kurbağa kuklası "Yuppiii" yazısını gösterecektir. Oyunun 30 saniye sonunda durabilmesi için "Süreölçer" değeri 30'dan büyük olup olmadığı sorgulanmaktadır. Eğer "süreölçer" değeri 30'dan büyük ise oyun durdurulmaktadır.

BÖLÜM

PROBLEM ÇÖZÜMÜNDE DÖNGÜLERİN KULLANIMI

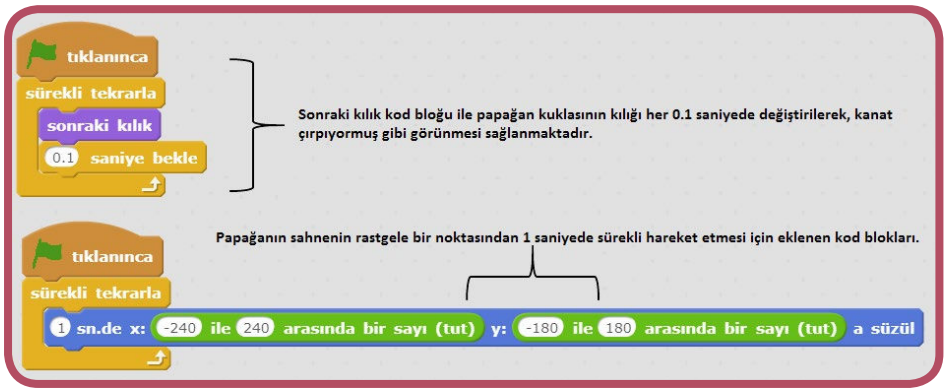
BÖLÜM 7: PROBLEM ÇÖZÜMÜNDE DÖNGÜLERİN KULLANIMI

Günlük hayatımızda döngülerin kullanımına birçok örnek verilebilir. Örneğin, okuldaki fotokopi makinesi bir sayfa kâğıdın bir çok kopyasını çıkarmak için kullanılmaktadır. Bunun için makineye çoğaltmak istenilen kâğıt konulur ve kaç tane çoğaltmak istendiği makine üzerindeki ekrana yazılarak işlem başlatılır. Bunun yerine her seferinde aynı işlemi yapmak durumunda kalındığında işler zorlaşabilir veya içinden çıkılmaz bir hale gelebilir. Fotokopi makinesi böyle bir durumda bir işlemi istenilen kadar tekrarlayacak bir "döngüye" girer ve aynı görevi aynı adımlarla tekrarlar. Döngüler programlarda aynı işlemi defaatle tekrar etmek istediğimizle kullanılan yapılardır ve programların verimli çalışması için çok önemlidir.

7.1: Sürekli Tekrarla Döngüsü

Sürekli tekrarla döngüsü programlarda sıklıkla kullandığımız sonsuz döngüsünü ifade eder. Döngüden ancak programcını hangi şartlar sağlandığında uygulamanın durmasını istediğini belirtmesi gerekmektedir. Şart sağlanmadığı süreci döngü sonsuza kadar devam edecektir. Örneğin bir önceki bölümde yer alan Etkinlik 6'da "süreölçer" değeri 30 sayısından büyük olduğunda uygulamanın durması istenmektedir. Bu durumda "süreölçer" değeri 30 oluncaya kadar döngü tekrarlanır ve 30 olduğunda program durur. Sahnedeki bir papağanın, sürekli tekrarla döngüsü kullanarak, kanatlarının çırpmasının programlanması Şekil 60'da gösterilmektedir.

Şekil 59: Papağanın sahne üzerinde rastgele kanat çırparak hareket etmesi



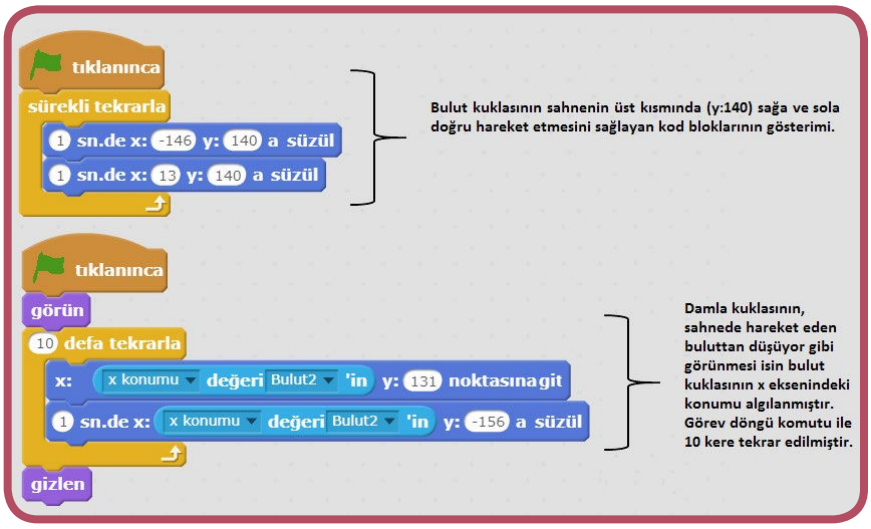
Şekil 60: Papağanın sahnenin rastgele noktalarına, kanat çırparak gitmesini sağlayan kod blokları

7.2: ... Defa Tekrarla Döngüsü

Bir önceki bölümde yer alan sürekli tekrarla döngüsü papağanın ekranda sonsuza kadar kanat çırpmasını sağlamaktadır. Bu durum her zaman istenmeyebilir. Papağan bir ağacın dalına konduğunda kanatlarını çırpmayı bırakması gerekir. Böyle bir durumda belirli bir sayıda döngünün tekrar etmesi ve daha sonra döngünün sonlandırılması istenebilir. Bunun için "... Defa Tekrarla" döngüsü kullanılmaktadır. Şekil 61'de sahnede hareket eden bir bulutun belirli bir sayıda (10) yağmur damlasını düşmesi örneği görülmektedir.

Şekil 61: Buluttan belirli sayıda düşen yağmur damlaları

Uygulama kodları Şekil 62'de gösterilmektedir. Bulut sürekli hareket ederken, damlanın kaç defa düşeceği bizim kontrolümüzdedir. İstenirse şarta veya süreye bağlı olarak farklı sayıda damlayı buluttan yağdırma imkanımız vardır.



Şekil 62: Sahnede kullanılan bulut ve damlanın kodları

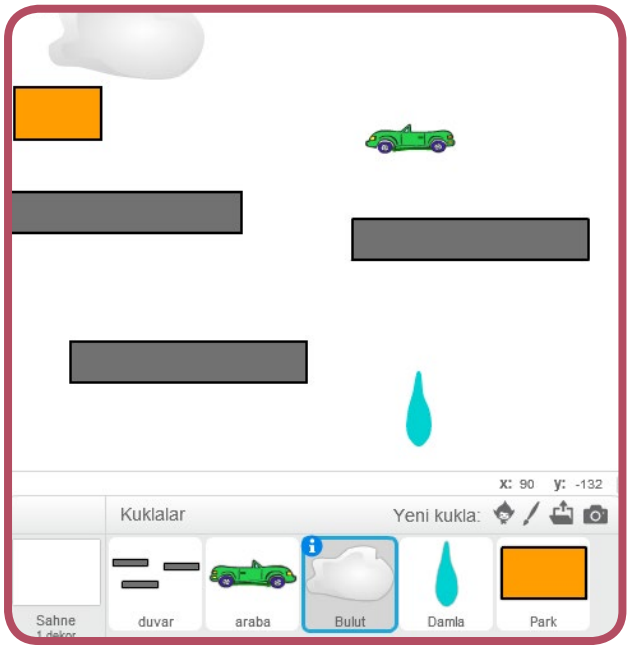
7.3: ... Olana Kadar Tekrarla Döngüsü

Sürekli tekrarla ve belirli sayıda tekrarla döngü bloklarının yanı sıra, belirli bir şart sağlana kadar devam etmesi istenen döngülerde oluşturulabilir. Bu durumda bir şart meydana gelinceye kadar diğer olaylar devam edecektir. Bahçe belirli bir seviyede sulanana kadar musluğun açık kalması veya su kaynadığında çay makinesinin kapanması gibi durumlar buna örnek olabilir. Şekil 63'de gösterilen kod blokları "akıldan sayı tutma" oyununa aittir. Bu oyunda "akılda tutulan sayı" kullanıcı buluncaya kadar döngü devam etmekte ve ekrana "yanlış cevap, tekrar denemelisinin" yazısı çıkmaktadır. Doğru cevap bulunduğunda ise "Tebrikler, doğru cevap" yazısı ekrana çıkmakta ve döngü biterek oyun sonlanmaktadır.

Şekil 63: Sayı bulmaca oyunu kodları

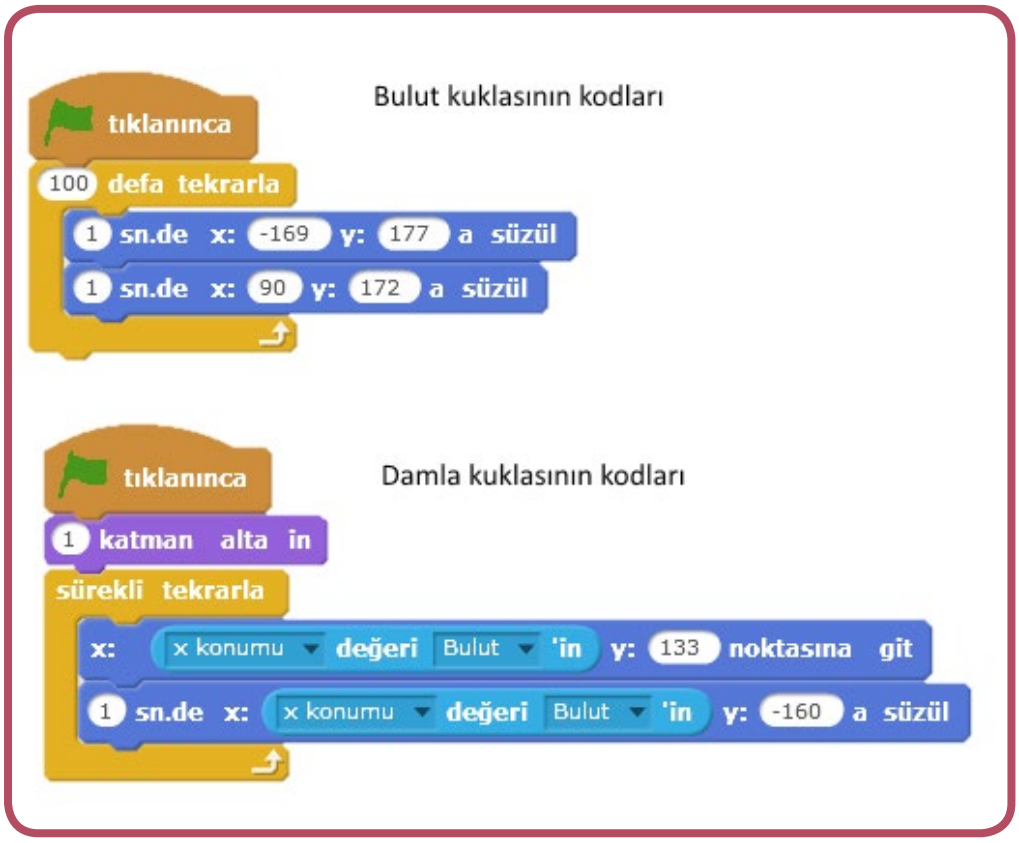
Etkinlik 7: Park Etme Oyunu

Park etme oyunundaki amaç, sahnedeki araba kuklasının, park alanı çizgilerine ve su damlalarına değmeden park etmesini sağlamaktır. Oyun için öncelikle sahne tasarımı Şekil 64'deki gibi yapılmıştır.



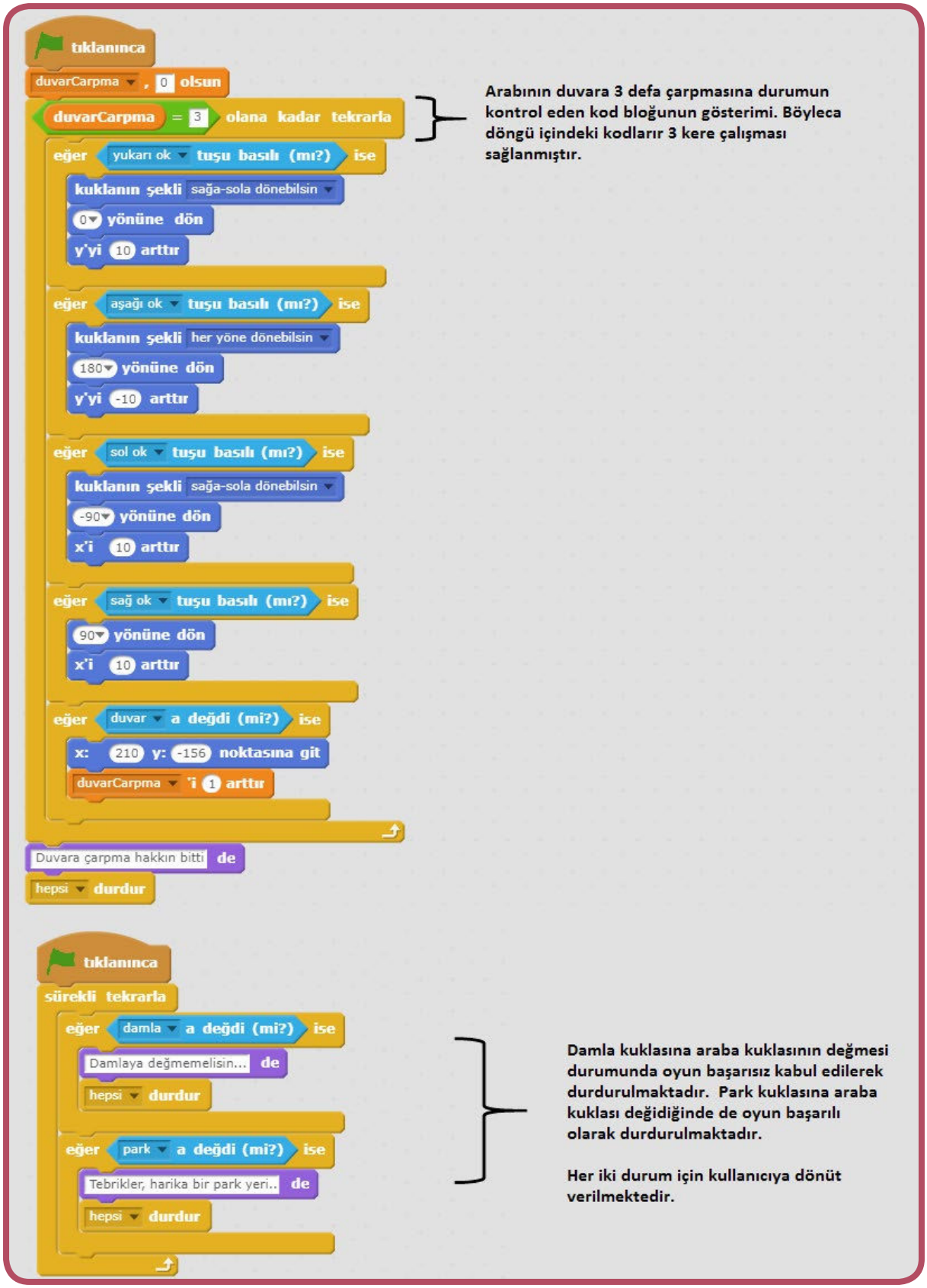
Şekil 64: Araba park etme oyunu sahne tasarımı

Sahne tasarımından sonra amaç doğrultusunda kuklaların çalışabilmesi için kod blokları yazılmıştır. Kodlar Şekil 65'de gösterildiği gibi bulut ve damla kuklaları için ayrı ayrı yazılmıştır. Kod bloklarına göre bulut kuklası 100 defa tekrarla döngüsü içerisinde sah¬nenin sağına ve soluna doğru hareket etmektedir. Damla kuklası ise sürekli tekrarla döngüsü içinde, bulut kuklasının X eksenindeki yerini kullanarak, yukarıdan aşağı doğru hareket etmektedir. Şekil 66'da ise oyunda kullanılan araba kuklasının kod blokları gösterilmektedir.



Şekil 65: Bulut ve damla kuklalarının komutları

Şekil 66'da uygulamada araba kuklasının kodları gösterilmektedir.



Şekil 66: Araba kuklasının kodlarıdır

BÖLÜM

SCRATCH İLE UYGULAMALAR

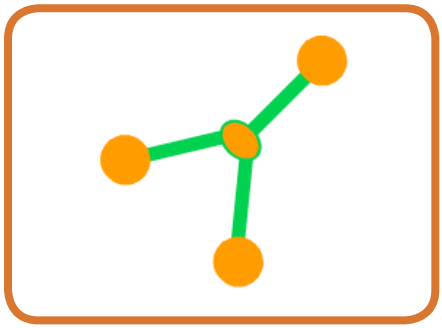
BÖLÜM 8: SCRATCH İLE UYGULAMALAR

Bu bölümde Scratch programlama dile ile yapılmış değişik konuları ve seviyeleri içeren örnekler uygulamalar yer almaktadır. Bu örnekler öğrencilerin seviyesine veya ilgilerine göre kullanılabileceği gibi, değişiklikler yapılarak öğrenciler için farklılaştırılarak kullanılabilir.

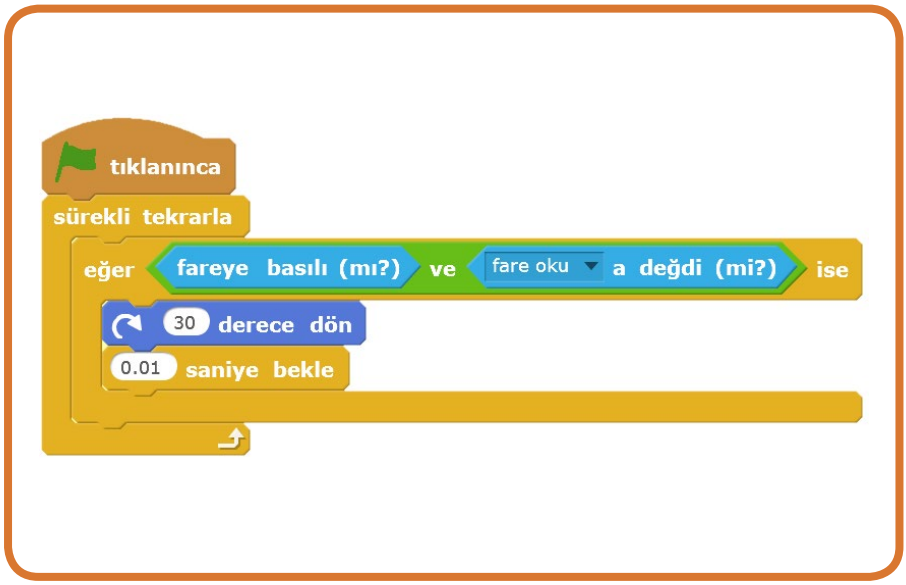
8.1: Stres Çarkı

Bu örnekteki amaç dijital olarak stres çarkı yapmak ve kodlamaktır. Bunun için ilk önce Şekil 67'deki gibi bir stres çarkı çizimi yapılmalıdır. Tasarlanan çarkın çalışması için fare ile üstüne tıklanması gerekmektedir. Stres çarkı, tıklandığından itibaren sağa doğru 30 derecelik açılar ile dönmelidir.

Örneğe ait kod blokları Şekil 68'de gösterilmektedir. Kod blokları incelendiğinde "sürekli tekrarla" ve "eğer" kontrol blokları kullanıldığı görülmektedir. Kod bloklarına göre, fare stres çarkına değdiğinde ve farenin sol tuşu tıklandığında çark sağ doğru, 0,01 saniye aralıklarla dönmeye devam edecektir.



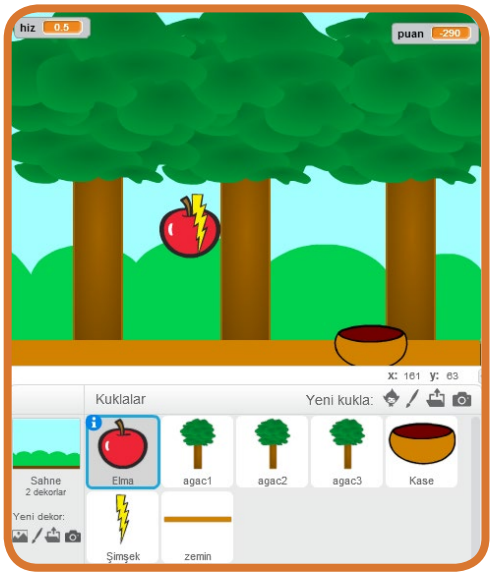
Şekil 67: Stres çarkı kuklası



Şekil 68: Stres çarkı uygulama kodları

8.2: Elma Toplama Oyunu

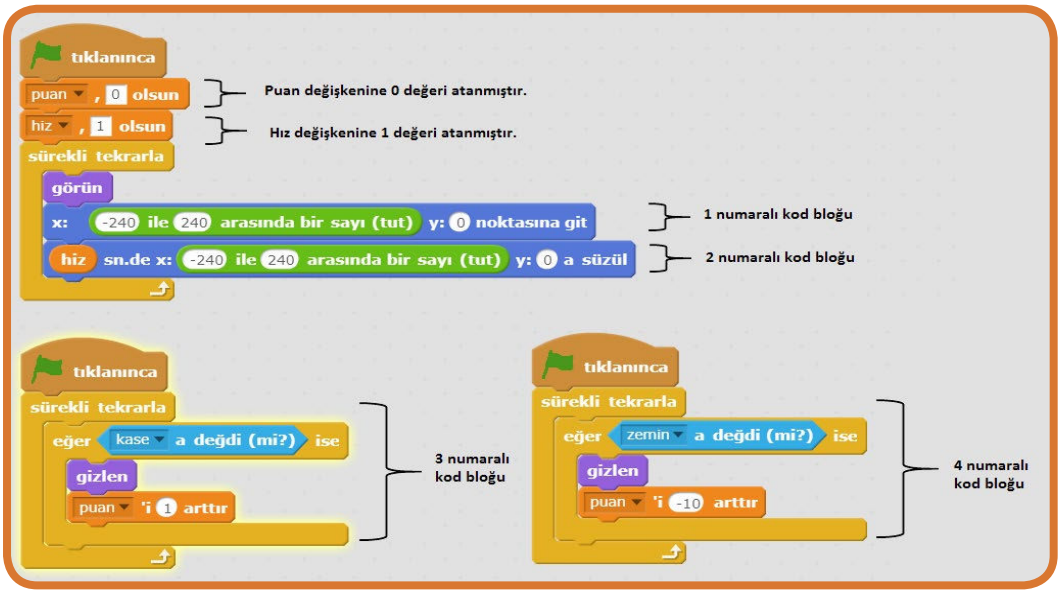
Bu oyundaki amaç sahnenin üstünden düşen elmaların bir kâse içinde toplanmasıdır. Toplanan elmaların her biri oyuncuya 10 puan kazandırmaktadır. Yere düşen elmalar yani yakalanamayan elmalar ise oyuncuya 10 puan kaybettirmektedir. Oyunun amaçlanan şekilde kodlanabilmesi için sahnede elma kuklasının yukarıdan aşağıya düşmesi ve kullanıcın yönettiği kâse ile elmaların yakalanması gerekmektedir. Kullanıcını kâse ile elmaları yakılabilmesi kâsenin elmalara değmesinin kontrolü ile gerçekleşmektedir. Oyunda yer alan bir diğer şart ise şimşek kuklası ile verilmektedir. Şimşek kuklası yakalandığında, elma kuklası iki katı hızlı hareket edecektir. Böylece oyunun zorluk derecesi arttırılmış olacaktır. Oyunun tasarlanabilmesi için ilk önce sahne tasarımı yapılmıştır. Şekil 69'da yapılan sahne tasarımı ve kuklalar görülmektedir. Sahnedeki zemin görseli hariç diğer kuklalara Scracth kütüphanesinden erişilebilir. Arka zemin programcı tarafından farklı şekillerde yapılabilir



Şekil 69: Elma toplama oyunu karakterleri

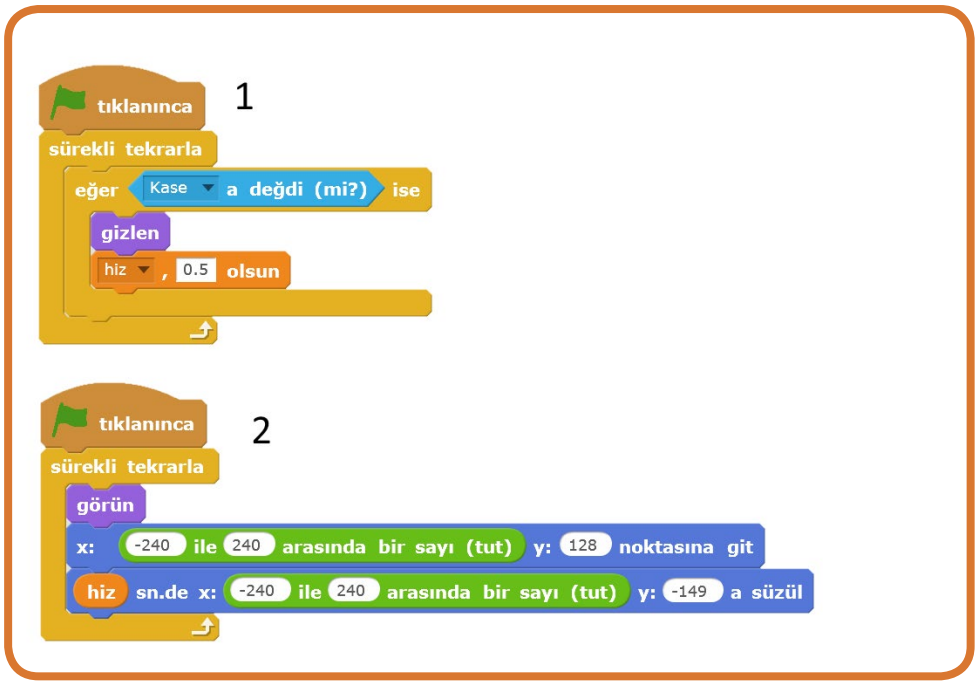
Oyun için yazılan kod blokları Şekil 70'de gösterilmektedir. Şekil 70'de Elma kuklasının komutları gösterilmektedir. Burada yeşil bayrağa tıkladığımızda çalışmaya başlayan 3 ayrı kod gurubu vardır. İlk kod gurubunda elmanın hareketleri sağlanmaktadır. Ayrıca puan ve hız değişkenlerinin de değerleri verilmiştir. 1 numara ile gösterilen kod bloğunda elmanın, sahnenin X ekseninde -240 ila 240 arasında rastgele bir konumda Y ekseninde ise 128 noktasında yer alması sağlanmıştır. Bu sayede elma sürekli aynı yükseklikten fakat yatayda farklı noktalardan, aşağı süzülme hareketine başlaması sağlanmıştır. 2 numaralı kod ile sahnenin alt kısmına doğru yatayda rastgele bir noktaya, dikeyde ise sabit -149 noktasına süzülmesi sağlanmıştır.

Burada "hiz" değişkeni elmanın yukarıdan aşağıya doğru kaç saniyede süzüleceğini belirtmektedir. Bu "hiz" değişkeninin değeri düşürülerek daha hızlı hareket eden bir elma kuklası sağlanabilir. Elma kuklasının 3 numaralı kod gurubunda ise; elma kuklasının kase kuklasına değip değmediği sorgulanmaktadır. Eğer kase kuklasına değiyor ise, elma kuklasını gizleriz, ardından puan değişkeninin değerini 10 arttırırız. Burada elmayı gizlememizin amacı puan değişkeninin sürekli olarak artmasını engellemektir. 4 numaralı kod grubunda gösterilen, "zemine değdi mi?" sorgusu ile elmanın zemine değip değmediği sorgulanmaktadır. Eğer ki elma kuklası, zemine değdi ise elma kuklası gizlenmekte ve puan değişkenin değeri 10 azaltılmaktadır. Yine burada da gizle komutu sürekli olarak puan değişkeninin azalmasını engellemek için kullanılmaktadır.



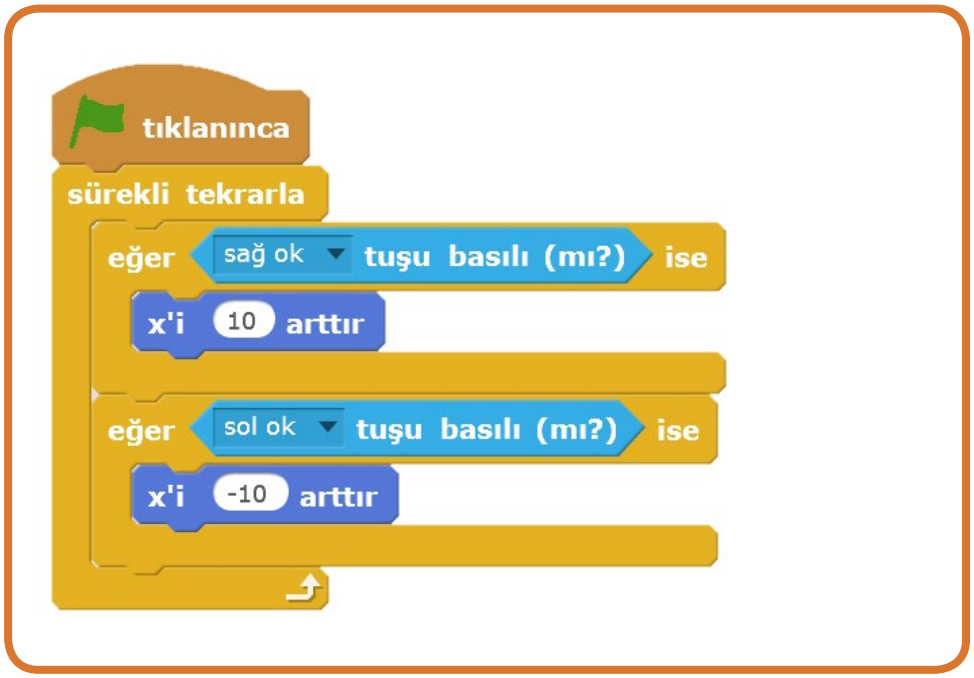
Şekil 70: Elma kuklasının komutları

Şekil 71'de ise şimşek kuklasının kodları gösterilmektedir. Şimşek kuklası oyunun hızını arttırmakla görevli olan kuk¬-ladır. Oyuncu eğer şimşek kuklasına kâse ile değerse elmaların düşüşü hızlanacaktır. Böylece oyun biraz daha zorlaş-mış olacaktır. Şimşek kuklası içine 1 ve 2 numarası ile gösterilen iki kod bloğu yazılmıştır. Bunlardan 1 numaralı kod bloğu şimşek kuklasının kâseye değmesi durumunda şimşek kuklasının gizlenmesi ve "hiz" değişkeninin değerinin 0,5 olarak değişmesi içindir. 2 numaralı kod bloğu ise şimşek kuklasının sahnedeki hareketi sağlamak içindir. Elma kuk-lasında olduğu gibi yatay eksende rastgele bir noktadan, dikey eksende ise sabit 128 noktasından harekete başlayan şimşek kuklası, aşağı doğru süzülürken, yine yatayda rastgele bir noktaya dikeyde ise -149 noktasına süzülmektedir.



Şekil 71: Şimşek kuklası kodları

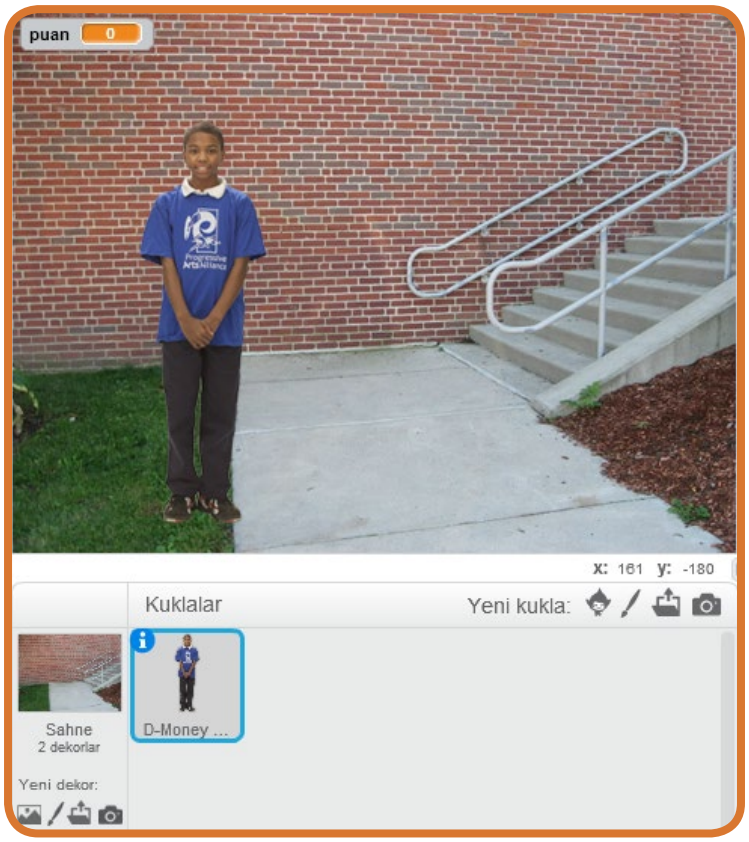
Kase kuklasına ait kod blokları ise Şekil 72'de gösterilmektedir. Kâse kuklasının iki görevi bulunmaktadır: Bunlardan ilki klavyeden sağ ok tuşuna basıldığın¬da sağa gitmesi ve sol ok tuşuna basıldığında ise sola gitmesidir. Kod bloğu üzerinde yatay eksende X değerini arttırdığımızda sağa doğru, X değerini azalttığımızda ise sola doğru hareket ede-cektir.



Şekil 72: Kase kuklası kodları

8.3: Çarpım Tablosu Öğreniyorum

Bu örnekteki amaç sahnede bulunan çocuk kuklasının, oyuncuya rastgele sayıların çarpımını sormasını sağlamaktadır. Oyuncu doğru cevabı verdiğinde ise puan alacaktır.

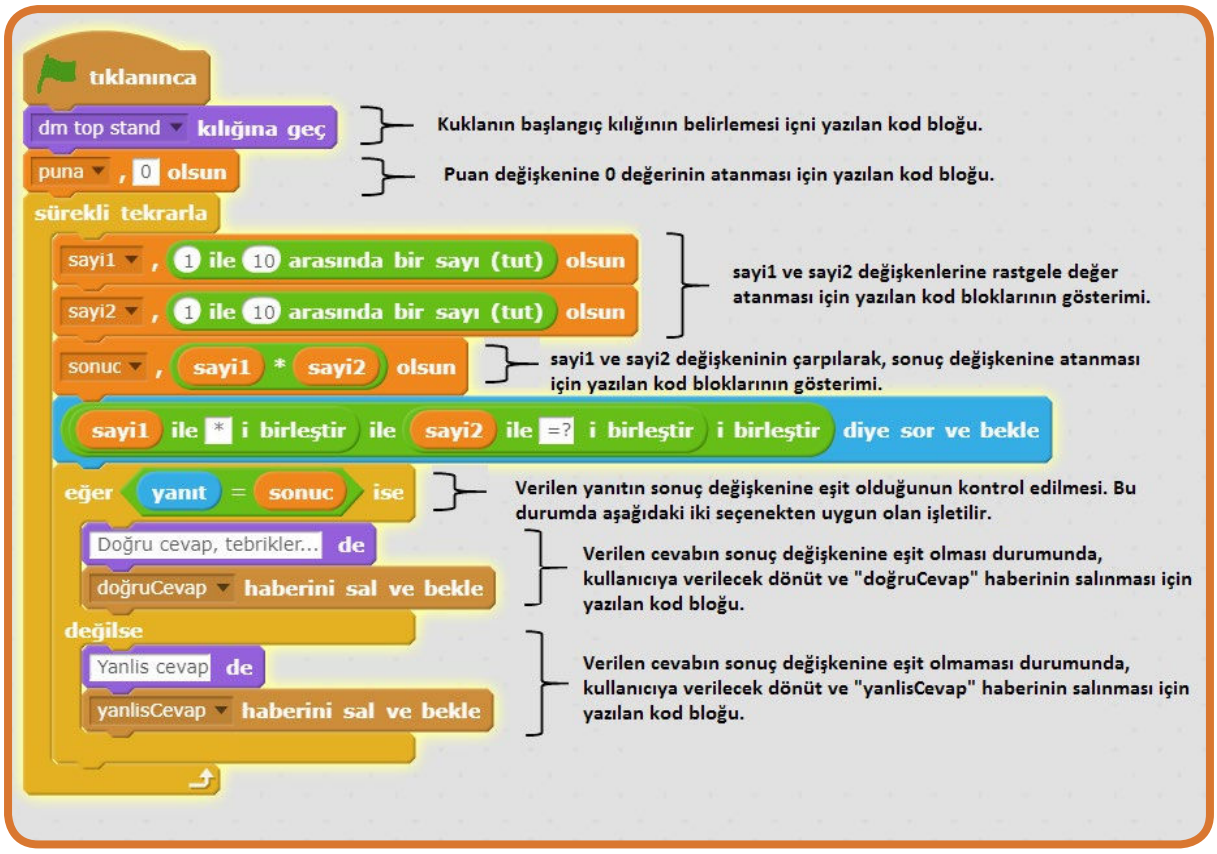


Şekil 73: Çarpım tablosu uygulaması sahne tasarımı

Oyun için gerekli olan tek kukla Şekil 73'de görülmektedir. Sahnedeki kukla, kullanıcıya rastgele iki sayının çarpımını sormaktadır. Kullanıcı bu soruya doğru veya yanlış cevap verdiğinde sahnedeki kuklanın buna göre hareket etmesi, kullanıcıya sesli ve yazılı dönüt vermesi sağlanmaktadır.

Kod blokları incelendiğinde, Sayı1, Sayı2, Sonuc ve Puan olmak üzere 4 değişken tanımlanmıştır. Sayı1 ve Sayı2 değişkenleri rastgele üretilen iki sayıyı temsil etmektedir. Sonuç değişkeni ise bu rastgele üretilen iki sayının çarpım değerini tutmaktadır. Puan de¬ğişkeni ise kullanıcının vermiş olduğu doğru cevaplardan kazandığı puanları ekranda göstermek için kullanılan değişkendir. Şekil74'de uygulamada kullanılan değişkenler gösterilmektedir.

Şekil 74: Çarpım tablosu uygulamasında kullanılan değişkenler



Şekil 75: Sahnedeki kuklanın çarpım tablosunu sorma komutları

Şekil 75'de çarpım tablosunu soracak kuklanın komutları gösterilmektedir. Buna göre öncelikle kuklanın başlangıç kı¬lığı seçilmektedir. Ardından puan değişkeni 0 yapılmaktadır. Uygulamada "say1" ve "sayı2" değişkenlerinin değeri 1 ila 10 arasında rastgele sayılardan belirlenmektedir. "Sayı1" ve "Sayı2" değişkenlerinin değerleri çarpıldıktan sonra sonuç değişkenine aktarılmaktadır. Daha sonra kullanıcıya "Sayı1" ve "Sayı2" değişkenleri kullanılarak **"sayı1 * sayı2 =?**" ifadesi ekranda sorulmaktadır. Kullanıcının vermiş olduğu cevap "yanıt" değişkenine aktarılarak, "yanıt" değişkeninin "Sonuc" değişkenine eşit olup olmadığı sorgulanmaktadır. Eğer eşit ise kullanıcıya "Doğru cevap Tebrikler" ifadesi ile dönülmekte ve "doğruCevap" haberi salınacaktır. Eğer ki "Yanıt" değişkeni "Sonuc" değişkenine eşit değil ise kullanıcıya "Yanlış cevap" ifadesi ile dönülecek ve "yanlışCevap" haberi salınır. "doğruCevap" haberi salındığında çalışacak olan kod blokları ise Şekil 76'da gösterilmektedir.

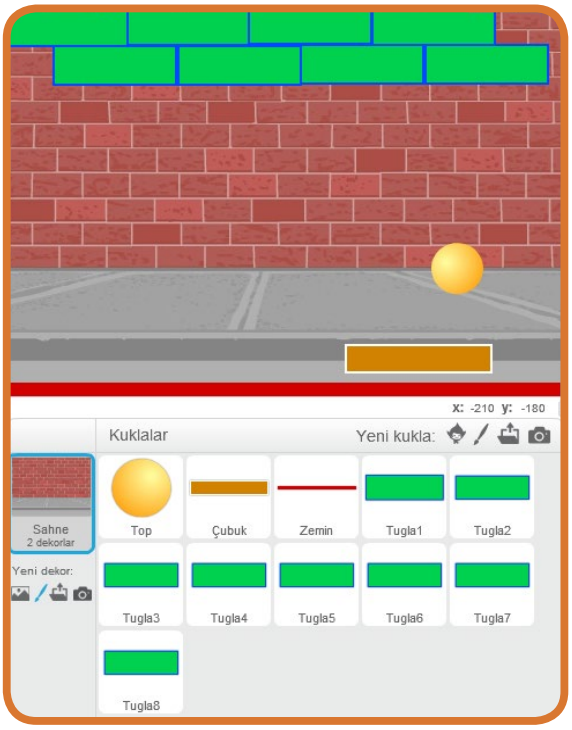
Şekil 76: DoğruCevap haberi geldiğinde çalışacak kodlar

Kullanıcı, vermiş olduğu cevaba uygun olarak metin ile göre dönüt verilmesinin yanı sıra ses ile de geri dönüt verilebilir. Bunun için "DoğruCevap" haberi geldiğinde Şekil 76'de gösterilen 1 numaralı kod bloğu çalışacak ve puan değişkeni 10 puan artarken "clapping" sesi çalacaktır. Aynı zamanda, 2 numaralı kod bloğu çalıştırılacak, sahnedeki kuklanın kılıkları arasında 0.3 saniye de geçiş yapılıp başlangıç kılığına dönülecektir. Benzer şekilde Şekil 76'da gösterildiği gibi, "YanlışCevap" haberi geldiğinde "bell toll" sesi çalacak ve "dm freze" kılığına geçip başlangıca geri dönmesi sağlanmış olacaktır. Şekil 77'de YanlışCevap haberinin kodları görülmektedir. YanlışCevap haberi geldiğinde çalışacak kodları incelediğimizde "bell toll" sesinin çalındığını ve sahnedeki kuklanın kılık değiştirdiğini görmekteyiz.

Şekil 77: YanlışCevap haberi geldiğinde çalışacak kodlar

8.4: Tuğla Kırmaca Oyunu

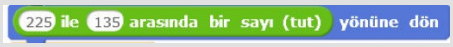
Bu oyunda, sahnenin rastgele bir noktasından gelen topa dokunarak aşağı düşmesini engellemek ve sahnenin üst kısmında bulunan blokları topun hareketiyle kırabilmek amaçlanmıştır. Bu amaçla tasarlanacak oyun için hazırla-nacak sahne tasarımı Şekil 78'de gösterilmiştir. Sahne tasarımını incelendiğinde top, zemin, çubuk, tugla1, tugla2, tugla3, tugla4, tugla5, tugla6, tugla7 ve tugla8 olmak üzere toplam 11 kukla görülmektedir. Sahne dekoru olarak kütüphanede yer alan "brick wall" dekorunu kullanılmıştır. Sesler için ise topun tuğlaya değmesi için "water drop", topun zemine değmesi için ise "buzz whir" seçilmiştir. Top karakteri, oyun başladığında sahnenin ortasındaki bir noktadan aşağı doğru hareket edecektir. Top sahne üzerinde, sahnenin aşağısına değmemek şartı ile her yöne hareket edebilmektedir. Top, sahnenin aşağısına yani zemine değdiğinde ise oyuncunun haklarından bir tane eksilmektedir. Oyuna başlarken, oyuncunun kaç kere oynayabileceğini belirten "can" adındaki değişken 5 olarak belirlenmiştir. Ekrandaki topun, ekranın aşağısına yani zemine değişmemişi için ise sarı renkle gösterilen çubuk kuklasının fare ile hareket ettirilmesi, topa değdirilerek topun sekmesinin sağlanması ve sahnenin üst kısmında bulunan tuğla kuklalarına değerek onları kırmasının sağlanması gerekmektedir. Top kuklası sahnenin üstündeki tuğlalara her değdiğinde sahnenin alt kısmına doğru tekrar inmelidir. Oyunun bitmesi için gerekli olan şartlar ise şöyle belirlenmiştir: Oyuncunun oynama hakkının 0 olması veya puan değerinin 80'e eşit olması. Her değilen yani kırılan tuğla oyuncuya 10 puan kazandırmaktadır. Oyunun farklı versiyonları yapılmak istenirse süre, tuzaklar veya ödüller eklenebilir. Bu şartlar doğrultusunda yazılan kod blokları Şekil 79'da gösterilmektedir.



Şekil 78: Tuğla kırmaca oyunu sahne tasarımı

Top Kuklasının komutları:

Top Kuklasının komutları: Tablo14'de "Top" kuklası için yazılan kod blokları ayrıntılı olarak gösterilmektedir.

	Can değişkeninin değeri 5 yapılmıştır.
	Puan değişkeninin değeri 0 yapılmıştır.
	Top kuklasının başlangıç noktası belirlenmiştir.
	Topun aşağı hareket etmesini sağlayacak şekilde 225 ila 135 arasında rastgele bir değer seçilerek, seçilen yöne dönmesi sağlanmıştır.
	Sürekli tekrarla bloğu içerisinde ise topun yönü hangi tarafa ise o tarafa doğru 15 adım gitmesi sağlanmıştır.
	Top eğer kenara gelir ise sekmesini sağlamak için komutu kullanılmıştır.
	Sürekli tekrarla bloğu içerisinde "Top" karakterinin "Çubuk" karakterine değip değmediği sorgulanmaktadır. Buna göre top karakteri eğer Çubuk karakterine değer ise topun yukarı doğru hareket etmesini isteriz. Bu amaçla topun yönünü -30 ila 30 arasında rastgele bir sayı hesaplanarak o yöne dönmesi sağlanır.
	"Top" karakterinin zemine değmesi durumunda ise hem top karakterinin yönü yukarı doğru çevrilmesi hem oyun oynama hakkının sayısını tutan can değişkeninin değerinin 1 azaltılması hem de kullanıcıyı uyarmak için "buzz whir" sesinin çalınması sağlanmıştır.
	Top karakterine yazdığımız bir diğer kod ise oyunun durmasını sağlayacak kodlardır. Uygulamada toplam 8 tuğlamız bulunur ve her tuğla kırıldığında 10 puan kazanacaktır. Böylelikle oyunun tamamlanması için 80 puan toplanabilir. Ayrıca oyun oynama hakkının da 0 olması durumunda da oyun durdurulacaktır. Her iki şarttan bir tanesi sağlandığında oyunun durdurulur.

Tablo 14: Top kuklasına yazılan komutların açıklamaları

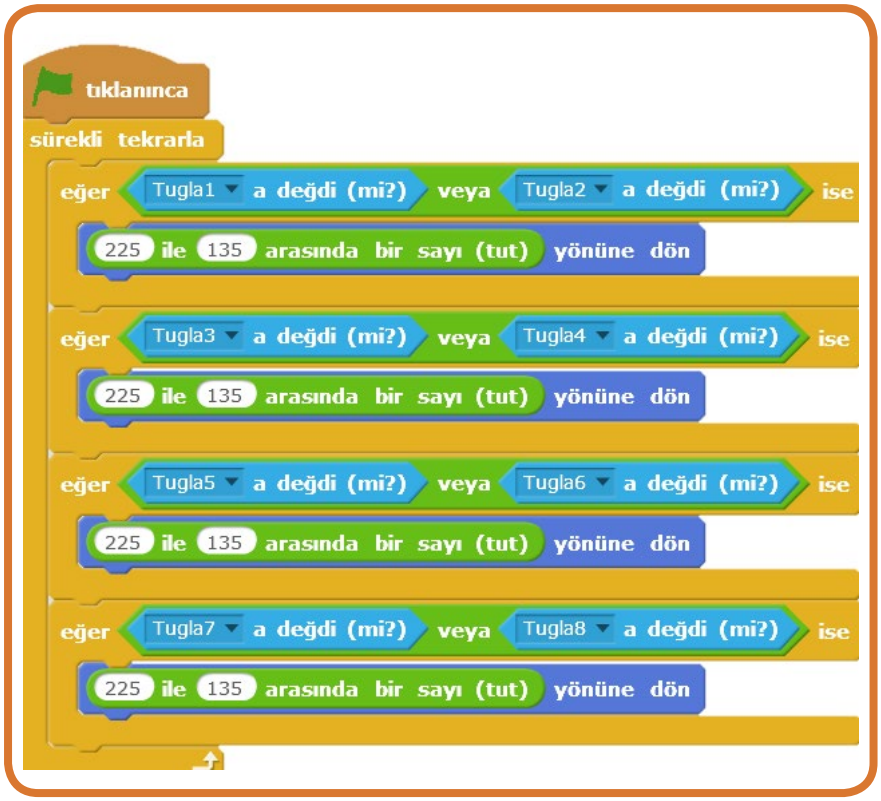
Şekil 79'da top kuklası için yazılan komutlar bir arada gösterilmektedir.

Şekil 79: Top kuklasının kodları-1

Top kuklasına eklenen ikinci kod blokları, top kuklasının tuğlalara çarptığında ekranın aşağısına dönmesini sağlamak içindir. Top kuklası, her bir tuğlaya çarptığında, yönü 225 ile 135 arasında rastgele bir değer alarak, o yöne doğru dönmesi sağlanmaktadır.

Şekil 80: Top kuklasının tuğlalara çarptığında aşağı yöne doğru dönmesini sağlayan şart ifadeleri

Şekil 80'de gösterilen komut bloğu ile "top" kuklası "Tuğla1" veya "Tuğla2" kuklalarına değdiğinde top kuklasının yönü aşağı doğru dönmektedir. Bu şart ifadeleri kod bloğuna eklenmediği durumda, "top" kuklası tüm tuğla kuklalarının içinden geçerek, ekranın bir yerine değinceye kadar yönünü değiştirmeyecektir. "Top" kuklasının yönünü belirlemek için yazılan diğer kod blokları ise Şekil 81'de gösterilmektedir.

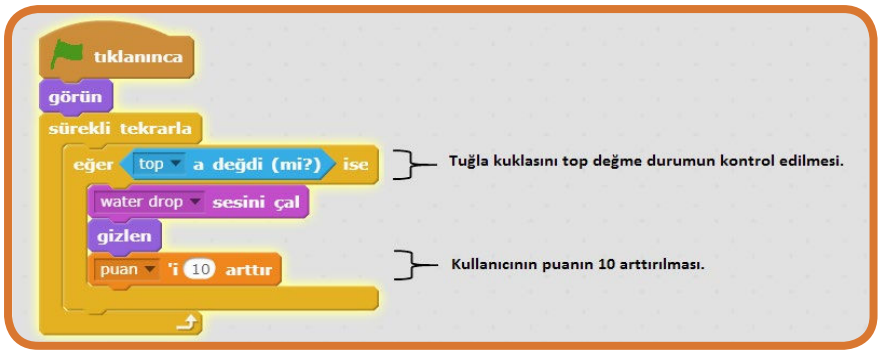


Şekil 81: Top kuklası komutları-2

Çubuk Kuklası Kodları:
Çubuk kuklası sahnenin alt kısmında farenin sağa-sola hareket etmesi ile hareket eden kukladır. Şekil 82'de çubuk kuklasının kodları gösterilmektedir. Kod blokları incelediğimizde çubuk kuklasının X eksenindeki yeri, farenin X esenindeki yeri ile aynı olacak şekilde ayarlandığı görülmektedir. Böylece kullanıcı fareyi sağa veya sola hareket ettirdiğinde çubuk kuklası da fare ile beraber etmesi sağlanmış olacaktır.

Şekil 82: Çubuk kuklası komutları

Tuğla Kuklalarının Kodları:
Şekil 83'de tuğla kuklaları için yazılmış olan kod blokları gösterilmektedir. Bu kod blokları her tuğla kuklası için kullanılmaktadır. Kodlar top kuklasının, tuğla kuklasına değme durumunda "water drop" sesinin çalınmasını, ardından tuğlanın gizlenmesini ve puanın 10 arttırılmasını içermektedir. Bu durum döngü ile sürekli hale getirilmiştir.



Şekil 83: Tuğla kuklası kodları

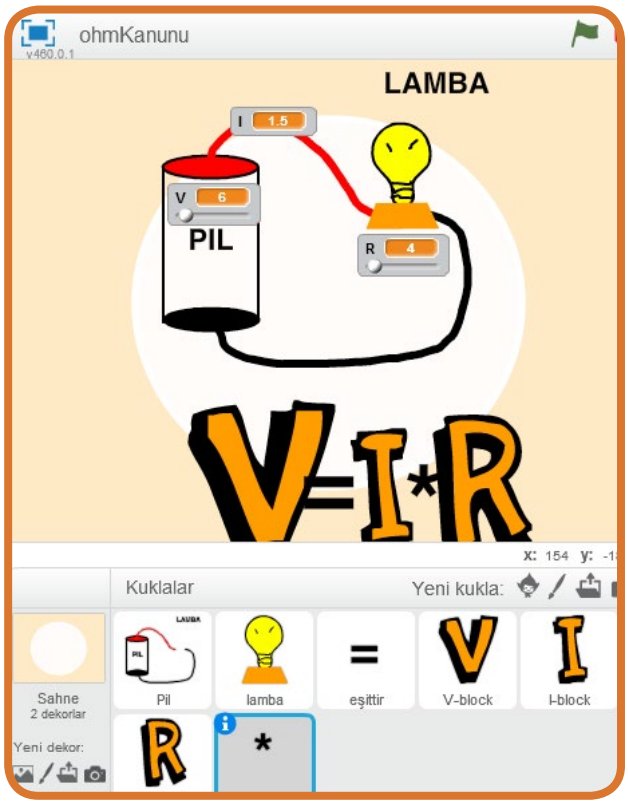
8.5: Ohm Kanunu Simulatörü

Bu uygulamamızda Ohm kanunun açıklayan bir simulatör yapılmaktadır. Bir direncin (R) uçlarına, bir gerilim (V) uygulandığında, direnç üzerinden bir akım (I) geçmektedir. Ohm kanunu olarak bilinen bu kanunun formülü Şekil 84'de gösterilmektedir.

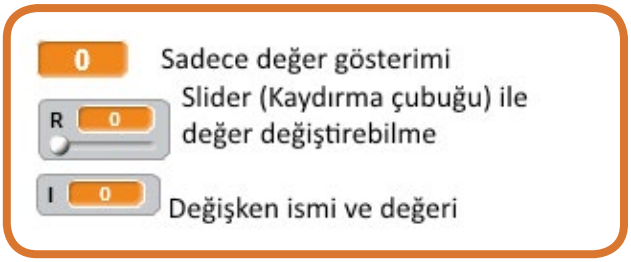
$$Akım(I) = \frac{Gerilim(V)}{Direnç(R)}$$

Şekil 84: Ohm kanunu formülü

Ohm kanunu simule eden bir program için öncelikli sahne tasarımı yapılmıştır. Şekil 85''de görüldüğü üzere gerilim ve direnç değerlerini kullanıcın tarafından belirlenebilmesi için **"kaydırma çubuğu"/"slider"** kullanılmaktadır. Devreden geçen akım değişkeni ise kodlar ile hesaplanarak sahnede sonuç değeri olarak gösterilmektedir.



Şekil 85: Ohm kanunu simulatörü sahne tasarımı



Şekil 86: Değişkenlerin gösterimi

Bir değişkenin sahne üzerinde gösterimi için birden fazla kod bloğu kullanılabilir. Bunlar sadece değerin gösterimi, kaydırma çubuğu ile gösterimi veya değişken ismi ve değerinin gösterimidir. İlk gösterim şeklinde değişkenin sadece değeri gösterilirken, değişkenin ismi gösterilmemektedir. İkinci gösterim şekli olan kaydırma çubuğunda değeşkenin değeri kullanıcı tarafından fare yardımı ile değiştirilebilmektedir. Aynı zamanda değişken ismi gösterilmektedir. Üçüncü gösterimde ise değişkenin ismi ve değeri gösterilmektedir. Şekil 86'da değişkenlerin gösterimi bulunmaktadır.

Bu örnekte kaydırma çubuğu gösterim şekli kullanılmış olup, kullanıcı tarafından belirlen gerilim ve dirence göre devreden geçen akımın değişimi izlenebilmektedir. Pil ve kablolar tek kukla olarak çizilmiş olup, lamba kuklası ayrı bir çizimden oluşmaktadır.

Pil Kuklasının Komutları:
Pil kuklası içinde yer alan kod blokları Şekil 87'de gösterilmektedir. Kodda V ve R değişkenleri için başlangıç değerleri 0 ve 1 olarak atanmıştır. Ardından sürekli tekrarla bloğu içinde I(akım) değeri hesaplanmış ve "akımGuncelle" haberi salınmıştır. Pil kuklası için yazılan ikinci kod bloğu "herhangi bir sayının 0'a bölümü tanımsızdır" bölme kuralını dikkate almaktadır. Buna göre R (direnç) değerinin 1'den küçük olmaması sağlanmıştır.

Şekil 87: Pil kuklasının komutları

Lamba Kuklası Kodları: Şekil 88'de lamba kuklasının kodları gösterilmektedir. "akımGuncelle" haberi geldiğinde lamba kuklasının renk etkisi, hesaplanan I (akım) değerine göre değiştirilmektedir.

Şekil 88: Lamba kuklası kodları

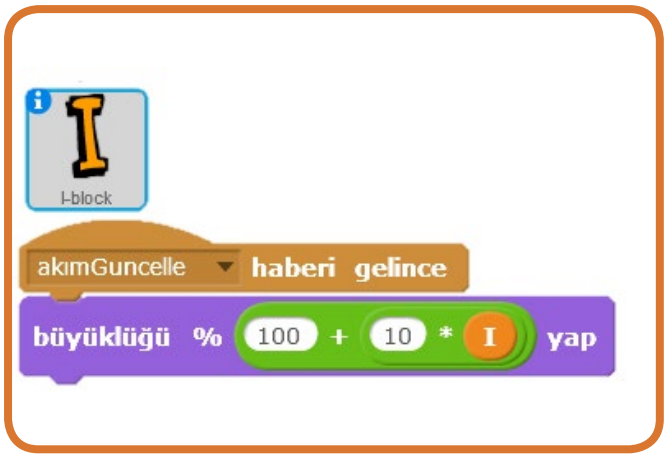
V Kuklası Kodları:
Sahnenin alt kısmında bulunan V kuklası için yazılan kod blokları Şekil 89'da gösterilmektedir. Buna göre V değişkeninin değeri kaydırma çubuğu ile değiştirildiğinde V kuklasının büyüklüğü değişmektedir.

Şekil 89: V kuklasının komutları

R Kuklasının Kodları:
Sahnenin alt kısmında bulunan R kuklasının komutları şekil 90'da gösterilmektedir. Buna göre her akımGuncelle haberi geldiğinde R kuklasının büyüklüğü R değeriyle orantılı olarak değiştirilmektedir.

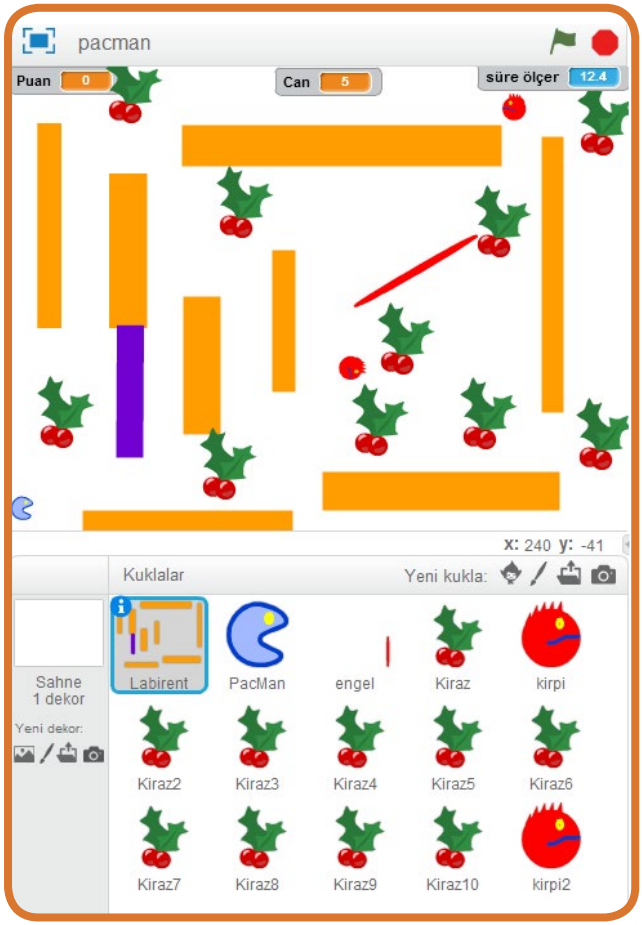
Şekil 90: R kuklasının komutları

I Kuklasının Komutları:
Sahnenin alt kısmında bulunan I kuklası için yazılan kod blokları Şekil' 91'de gösterilmektedir. Buna göre "akımGuncelle" haberi geldiğinde I kuklasının büyüklüğü I değeriyle orantılı olarak değiştirilmektedir.



Şekil 91: I kuklasının komutları

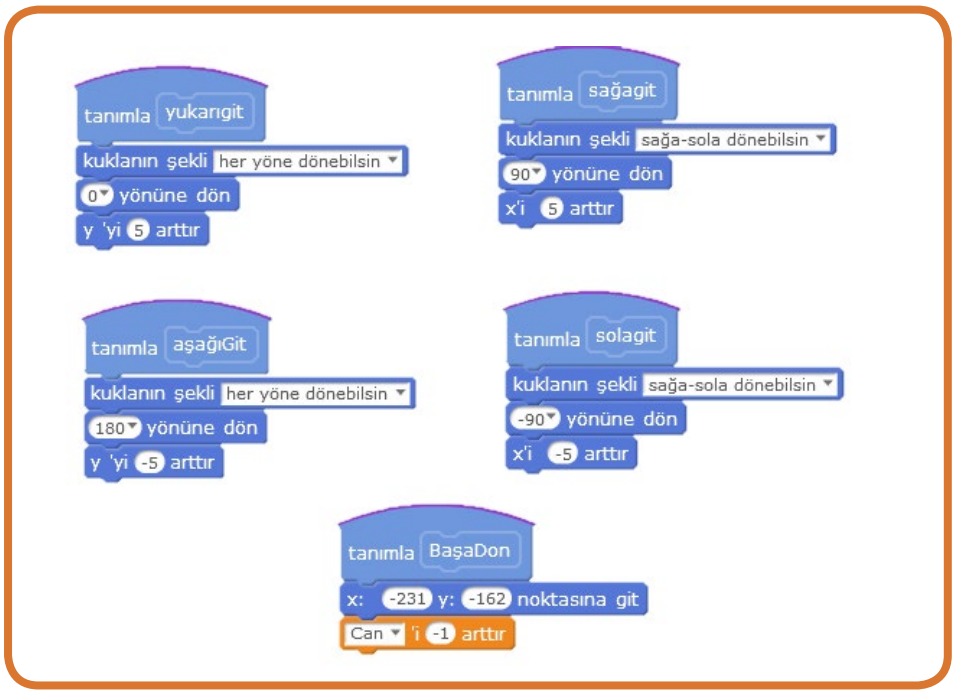
8.6: Pacman
Klasik bir kodlama örneği olarak Pacman adlı oyunun bir benzerini Scratch ile tasarlanmıştır. Oyun; sahnedeki "Pacman" karakterinin "kirazları" toplaması üzerine kuruludur. Görevini yerine getirirken Pacman karakteri sahnedeki "Kirpi" karakterlerinden kaçmalı, "labirent" ve "engel" kuklalarına çarpmamalıdır. Bu karakterlere çarpması durumunda, oyun başladığında 5 olarak belirlenen, "can" değeri 1 azalmaktadır. Karakterin kurallara uygun şekilde sahnedeki tüm kirazları 45 saniyelik bir süre içinde toplaması gerekmektedir. Şekil 92'de oyun için tasarlanan sahne ve kuklalar gösterilmektedir.



Şekil 92: Pacman oyunu sahne tasarımı

Pacman karakteri, görevini yerine getirebilmek için klavyedeki yön tuşları ile sahne hareket ettirilmektedir. "Engel" kuklası rastgele bir sıra ile ekranda sağa veya sola dönmektedir. "Kirpi1" kuklası sahneni üstünden başlayarak sağa sola hareket ederken, "Kirpi2" kuklası sahnenin ortasından başlayarak aşağı ve yukarı doğru hareket etmektedirler.

Pacman kuklasının komutları:
Şekil 93'de pacman kuklasının hareket edebilmesi için oluşturulan özel taşlar gösterilmektedir.



Şekil 93: Pacman kuklası için oluşturulan özel taşlar

Şekil 93'de Pacman kuklasının hareket edebilmesi için oluşturulan özel taşlar gösterilmektedir. Şekil 93 incelediğimizde oluşturulan "yukarıgit", "sağagit", "aşağıgit", "solagit" ve "başadön" özel taşları görülmektedir. Bu taşların görevleri sırasıyla şöyle belirlenmiştir. "yukarıgit" özel taşı kuklanın ekranın yukarısı yönüne döndürmek ve y eksenindeki yerini 5 arttırmak, "aşağıgit" özel taşı kuklanın aşağı yöne dönebilmesini ve kuklanın y eksenindeki yerini 5 adım azaltmak, "solagit" özel taşı kuklanın sadece sağa sola dönebilmesini sağlamak ve sonra kuklanın sol tarafa dönerek x eksenindeki yerini 5 adım arttırmak, "sağagit" özel taşı kuklanın sağa dönerek x eksenindeki yerini 5 arttırmak ve "BaşaDon" özel taşı Pacman kuklasının sahnedeki labirent, kirpi veya engel kuklalarından birine çarpması durumunda başlangıç noktasına dönmesini ve can değişkeninin 1 azaltılmasını sağlamaktır. Oluşturulan bu özel taşların kod blokları içinde nasıl kullanıldığı ise Şekil 94'de gösterilmektedir.

Şekil 94: Oluşturulan özel taşların kullanımı

Şekil 94'de gösterilen kod bloklarını çalışabilmesi için yeşil bayrak tıklandığında, sürekli olarak klavyeden yukarı, aşağı, sağ ve sol ok tuşlarına basılıp basılmadığı sorgulanmaktadır. Yön tuşlarından herhangi birine basılması durumunda, kuklanın basılan yön tuşu yönünde hareket etmesi sağlanmaktadır.

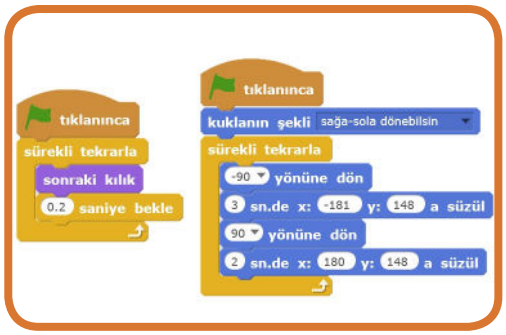
Şekil 95: Pacman kuklasının diğer kuklalara çarpması durumunu ve süreölçer, puan can değişkenlerinin değerini sorgulayan komutları

Pacman kuklasının sahnedeki diğer kukalarla etkileşimi durumunda çalıştırılacak kod blokları ise Şekil 95'de gösterilmektedir. Kodlar incelendiğinde "labiren", "engel", "kirpi" ve "kirpi2" kuklalarına çarpması durumunda "başadön" özel taşı çalıştırılmakta ve Pacman kuklasının sahnenin başlangıç noktasına dönmesi sağlanmaktadır. Aynı zamanda "can" değişkeni bir azaltılmaktadır. "Süreölçer" değişkenin değeri kodlar çalışmaya başlatıldığında sıfırlanmakta ve oyunun devam ettiği her saniye değeri arttırılmaktadır. Bu değişken, arka planda çalışan bir sayaç gibi kullanılmaktadır. "Süreölçer" değişkeninin değeri 45'den büyük olması durumunda Pacman kuklası "süre bitti" ifadesi ile kullanıcıyı dönüt vermekte ve "hepsini durdur" komutu ile tüm kuklaları durdurmaktadır. Pacman kuklası sahnedeki her kiraza değdiğinde (topladığında) 10 puan almaktadır. Puan değeri 100 olduğunda ise Pacman kukla¬sı,"Oyun bitti, tebrikler" ifadesi ile kullanıcıya dönüt vermekte ve sahnedeki tüm kuklaları durdurmaktadır. "Can" değişkeni, Pac¬man kuklasının sahnedeki engellere çarpması durumunda 1 azaltılmakta ve oyun yeniden başlatılmaktadır. "can" değişkenin değeri 0 olması durumunda Pacman, kullanıcıları "Can bitti " şeklinde dönüt vermekte ve ardından sahnedeki tüm kuklaları durdurmaktadır.

Engel Kuklasının komutları:
"Engel" kuklası için yazılan kod blokları Şekil 96'da gösterilmektedir. Bu kodlar ile "engel" kuklasının sağa-sola dönmesi sağlanmıştır. "10 ila 50 arasında rastgele bir sayı tut" komutu ile kuklanın kaç defa 15 derecelik açılar ile her 0.1 saniyede 1 defa döneceği belirtilmektedir. Ardından yine benzer şekilde 10-50 arasında bir değer kadar 15 derecelik açılar ile her 0.1 saniyede1 sola doğru dönmesi sağlanmıştır.

Şekil 96: Engel kuklası komutları

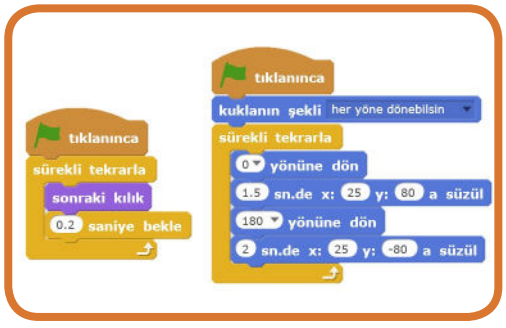
Kirpi Kuklasının Komutları: "Kirpi" kuklasını sahnenin sağ üst kısmından sahnenin sol üst kısmına kadar kılık değiştirerek hareket etmesini sağlayan kod blokları Şekil 97'de gösterilmektedir.



Şekil 97: Kirpi kuklasının komutları

"Kirpi" kuklasının görevi sahnenin üst kısmında sağa ve sola hareket etmektir. Bu amaçla "kuklanın şekli sağa-sola dönebilsin" komutu ile kuklanın sağ ve sola dönebilmesi sağlanmıştır. Ardından sahnenin soluna doğru dönmesi ve 3 saniye x:-181 y: 148 noktasına süzülmesi sağlanmıştır. Kodun devamında ise sahnenin sağına dönmesi ve 2 saniyede x:180 y:148 noktasına süzülmesi sağlanmıştır. Kirpinin sahne üzerinde ağzını açıp-kapama hareketi ise sürekli tekrarla bloğu içinde 0.2 saniyelik gecikmeler ile sonraki kılığa geçmesi ile sağlanmıştır.

Kirpi2 Kuklasının Komutları: "Kirpi2" kuklasının komutları "Kirpi" kuklası ile benzerdir. Farklı olarak "Kirpi2" kuklası sahnenin orta bölümünde yukarıdan aşağı doğru hareket etmektedir. Bu hareket için "Kirpi2" kuklasının her yöne dönebilmesi gerekmektedir. "sürekli tekrarla" döngü bloğu içerisinde ilk önce yukarı yönüne dönmesi, ardından 1.5 saniyede x:25 y:80 noktasına gitmesi sağlanmıştır. Daha sonra "Kirpi2" kuklasının sahnenin aşağısına dönenmesi sağlanarak 2 saniyede x:25 y:-80 noktasına gidilmiştir. Şekil 98'de bu eylemler için yazılmış kod blokları gösterilmektedir.



Şekil 98: Kirpi2 kuklasının komutları

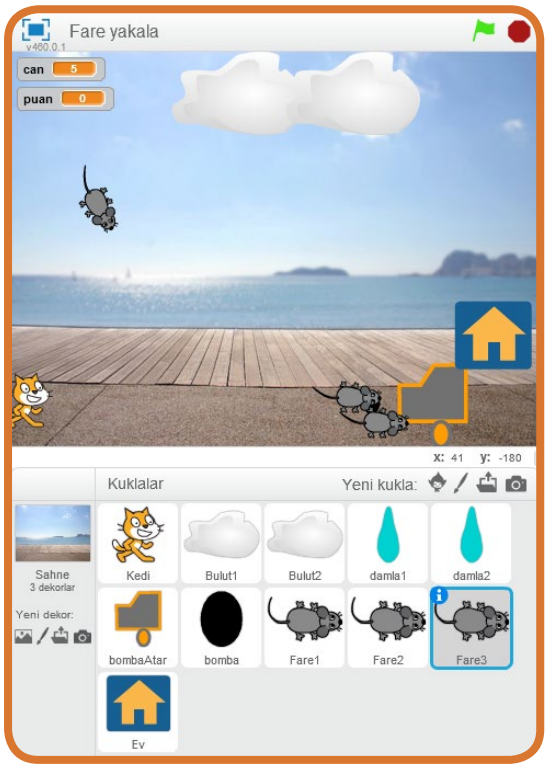
Kiraz Kuklasının Komutları:
Sahnede 10 adet kiraz kuklası kullanılmıştır. Bu 10 adet kuklanın her birine ayrı ayrı kod yazmak yerine bir tanesine kod yazıp, kuklanın kopyasını çıkartmak sonuca daha hızlı ulaşmamızı sağlayacaktır..

Şekil 99: Kiraz kuklası kodları

Kiraz kuklasının görevi oyuncunun puan toplamasını sağlamaktır. Yukarıdaki koda göre puan toplamak için pacman kuklası, kiraza değdiğinde kiraz kuklasının gizlenmesi gerekmektedir. Kiraz kuklası gizlendiğinde puan değişkeni 10 artacaktır.

Uygulamada gizlenen kuklaları göstermek amacıyla yeşil bayrağı tıklatma komutu, görün komutundan önce kullanılmıştır.

8.7: Fareleri Yakala
Bu uygulama da sahnedeki kedi karakteri, rastgele hareket eden fareleri engellere çarpmadan yakalamaya çalışacaktır. Kedi, tüm fareleri yakalayıp eve ulaştığında oyun bir sonraki seviyeye geçmektedir. Böylece birden fazla seviyeye sahip bir oyun tasarlanmıştır. Sahne içindeki tuzaklar yağmur damlası ve "bombaAtar" kuklaları ile gösterilmiştir. Kedi eve ulaştığında ekranda aynı zamanda "1. Seviye sona erdi" şeklinde kullanıcıya dönüt verilmektedir. Şekil 100'de sahne tasarımı gösterilmektedir.

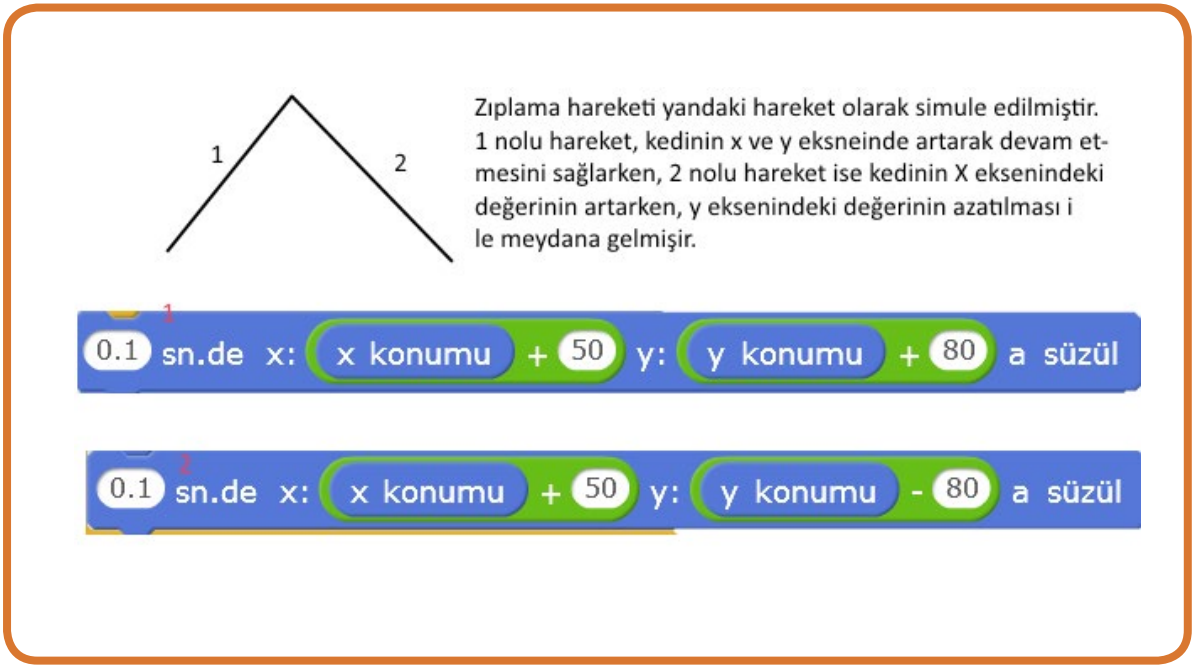


Şekil 100: Fare yakalama oyunu sahne tasarımı

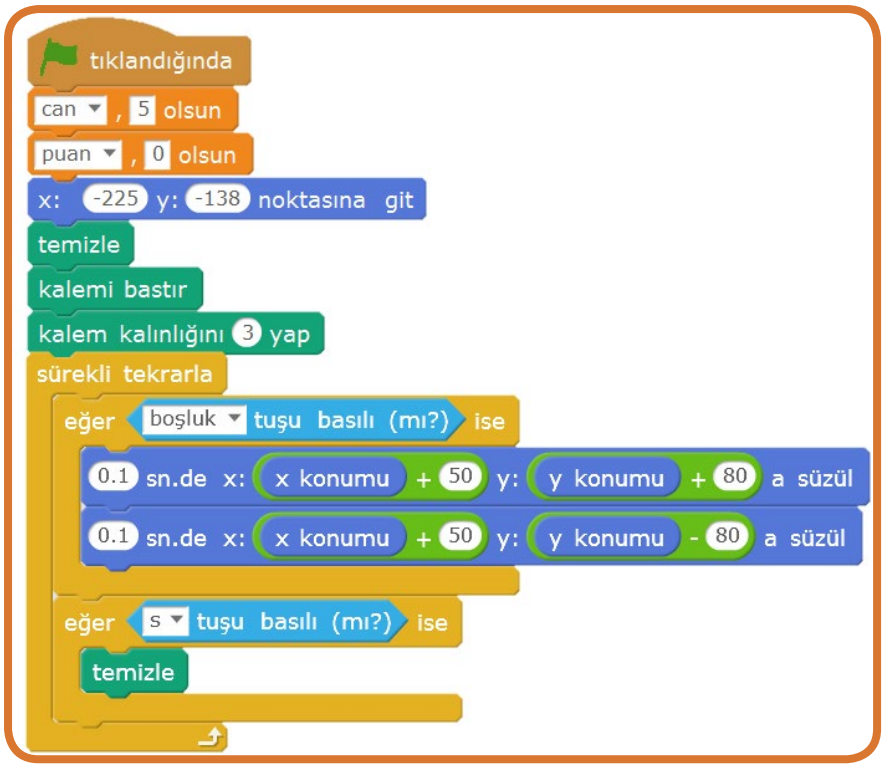
Kedi kuklasının kodları:
Şekil 101'de Kedi kuklasının yön tuşları ile hareket etmesi için gerekli komutlar gösterilmektedir.
Şekilde de görüldüğü üzere yukarı ok tuşuna basıldığında kedi kuklasının Y eksenindeki yeri 10 değer artmakta, aşağı ok tuşuna basıldığında kedi kuklasının Y eksenindeki yeri 10 değer azalmakta, sol ok tuşuna basıldığında kedi kuklasının X eksenindeki yeri 10 değer azalmakta ve sağ ok tuşuna basıldığında ise kedi kuklasının X eksenindeki yeri 10 değer artmaktadır.

Şekil 101: Kedi kuklası yön tuşları ile hareket ettirme komutları

Şekil103'de yazılan kod ile kedinin zıplaması ve arkasında bıraktığı kalem izinin S tuşuna basılarak temizlenmesini sağlanmaktadır. Kodu incelediğimizde can değişkeni 5 değerini alırken puan değişkini sıfır değerini almıştır. Kedinin başlangıç noktası olarak X:-225 Y:-138 seçilmiştir. Kalemi bastır komutu ile kedi hareket ettikçe kedinin arkasında bir ka¬lem izi bırakılacaktır. Boşluk tuşuna basıldığında ise kedinin zıplama hareketine benzer bir hareket etmesi sağlanacaktır.

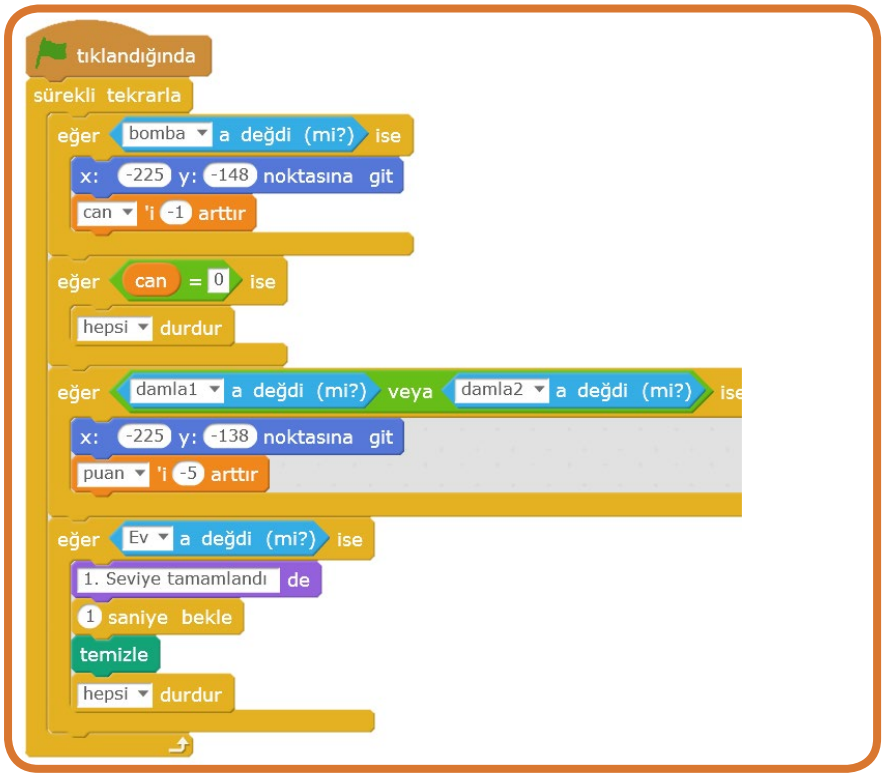


Şekil 102: Zıplama hareketi



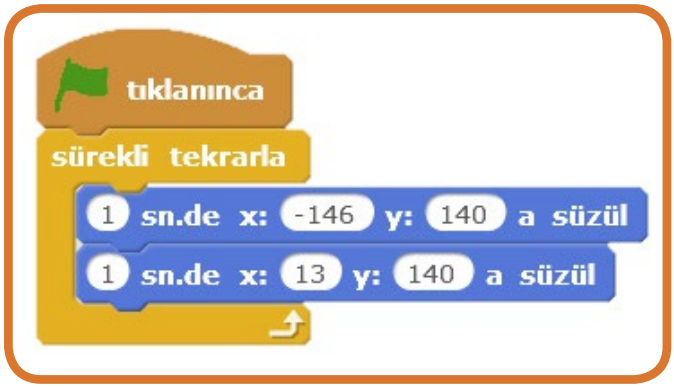
Şekil 103: Kedinin zıplama ve kalem izini temizleme komutları

Şekil 104 kedi kuklasının diğer kuklaları algılaması ile ilgili komutları göstermektedir. Buna göre kedi kuklası bomba kuklasına değdiğinde kedi kuklası başlangıç noktasına gider ve can değişkeninin değeri 1 azalır.
Damla1 veya damla2 kuklalarına değdiğinde ise kedi kuklası yine başlangıç noktasına gider ve puan değişkeninin değeri 5 azalır.
Kedi kuklası ev kuklasına değdiğinde ise oyuncuya 1. seviyeyi tamamladığına dair bilgilendirme yapılır ardından 1 saniye bekledikten sonra kalem izi temizlenir ve oyun durdurulur.



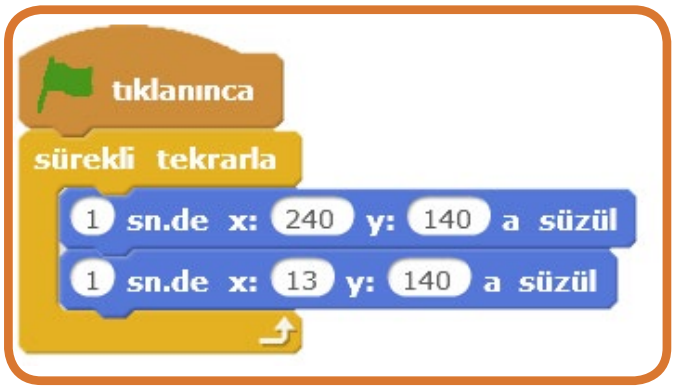
Şekil 104: Kedi kuklasının diğer kuklalara çarpması durumunu algılayan komutlar

Bulut1 kuklasının komutları:
Şekil 105'de Bulut1 kuklasının komutları gösterilmektedir. Komutları incelediğimizde Bulut1 kuklası 1 saniyede X:-146 Y:140 noktasına süzülmekte ardından X:13 Y:140 noktasına süzülmektedir. Yatay eksende sağa ve sola doğru hareket etmektedir.



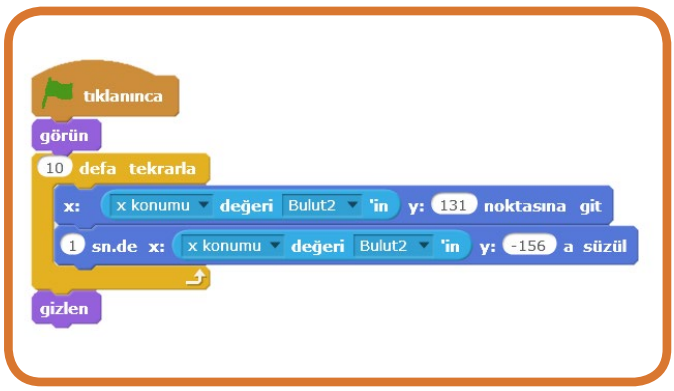
Şekil 105: bulut1 kuklasının komutları

Bulut2 kuklasının komutları:
Bulut2 kuklası da bulut1 kuklası gibi sahnenin üst kısmında sağa ve sola doğru hareket etmektedir. Şekil 106'da Bulut2 kuklasının komutları gösterilmektedir.



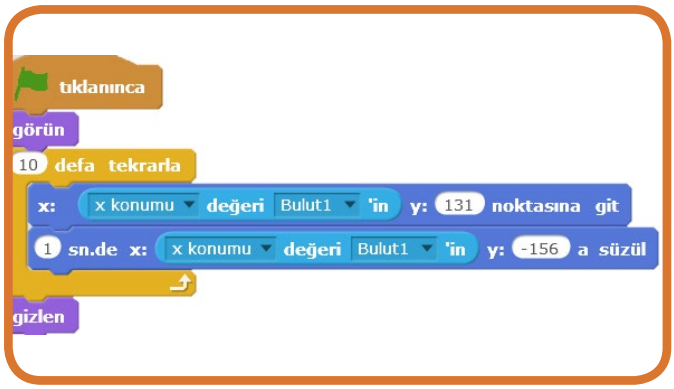
Şekil 106: Bulut2 kuklasının komutları

Damla1 Kuklasının Komutları:
Şekil 107'de damla1 kuklasının komutları gösterilmektedir. Damla kuklalarının sahnenin sağında ve solunda rastgele konumlara düşmesi istenmektedir. Bulut kuklası hareket ettikçe bulut kuklasından aşağı doğru hareket edecektir. Damla sayısı örnekte görüldüğü üzere döngü ile 10 defa yukarıdan aşağı düşecek ve sonra gizlenecektir.



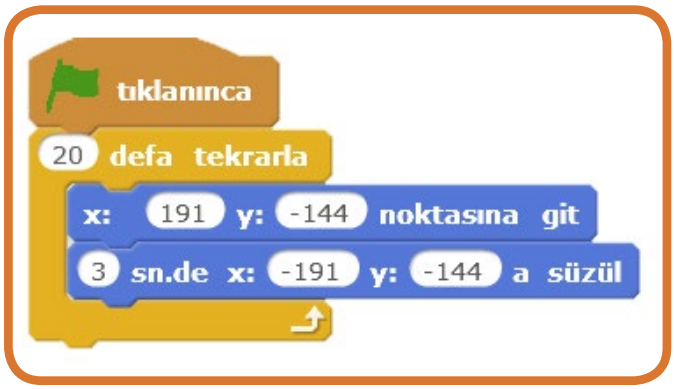
Şekil 107: Damla1 kuklasının komutları
Bulut kuklası x ekseninde sağa ve sola hareket etmektedir. Damla1 kuklası da X eksenindeki değerini bulut2'nin X ekse¬nindeki değerinden alarak harekete başlamaktadır. Y eksenindeki konumu 131 noktasından -156 noktasına gelmektedir.

Damla2 Kuklasının Komutları:
Damla2 kuklası da damla1 kuklası gibi hareket etmektedir. Damla2 kuklası bulut1 kuklasının X eksenindeki değerini almaktadır. Damla2 kuklasının komutları şekil108'de gösterilmektedir.



Şekil 108: Damla2 kuklasının komutları

Bomba Kuklasının Komutları:



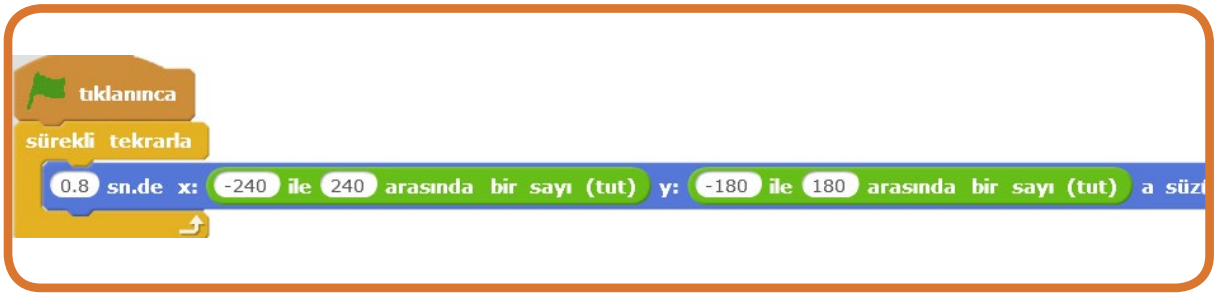
Şekil 109: Bomba kukla komutları

Şekil 109'da bomba kuklasının komutları gösterilmektedir. Bomba kuklası sahnenin sağ tarafında sabit noktaya gitmekte ve ardından sahnenin sol tarafındaki sabit bir noktaya 3 saniyede hareket etmektedir. Bu görevi 20 defa tekrarlayacaktır.

Fare Kuklasının Komutları:
Sahnede 3 adet fare kuklası bulunmaktadır. Bu kuklaların görevi sahnenin rastgele konumlarına hareket etmektir. Şekil 110'da Fare kuklasının komutları gösterilmektedir.

Şekil 110: Fare kuklasının komutları

Burada fare kuklasının uygulama başlarken görünmesi sağlanmıştır. Ardından sabit bir başlangıç noktasına gitmesi sağlanır. Daha sonra sürekli tekrarla bloğu içerisinde kedi kuklasına değip değmedi sorgulanmaktadır. Kedi kuklası nın değmesi durumunda, fare kuklası gizlenmekte ve puan değişkeninin değeri 10 arttırılmaktadır.
Şekil 111'de fare kuklasının sahne üzerinde rastgele hareket etmesi gösterilmektedir.

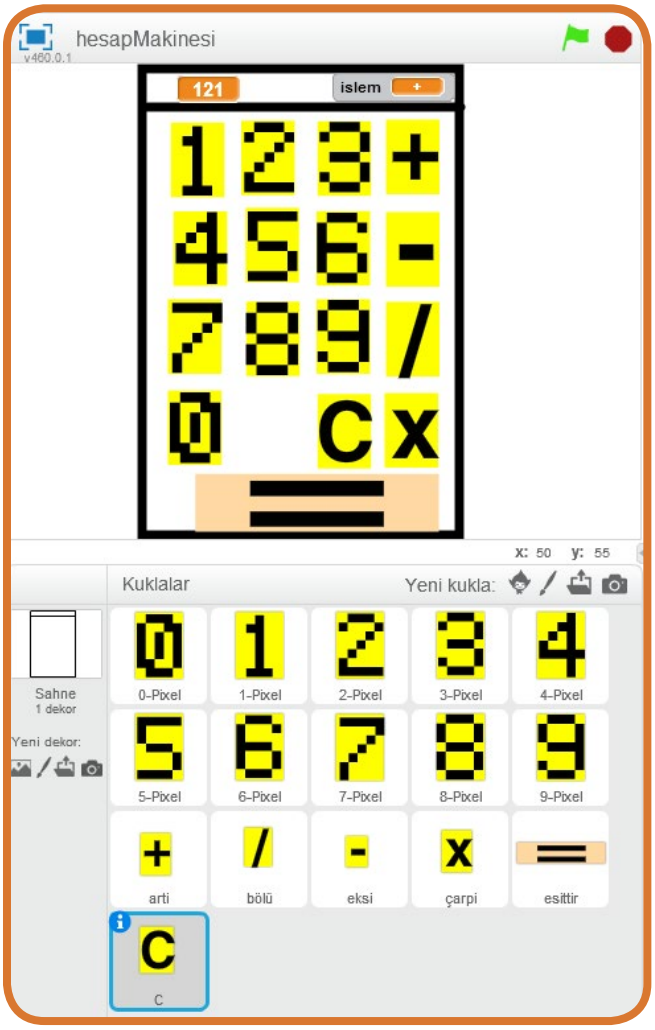


Şekil 111: Fare kuklasının sahne üzerinde rastgele hareket etmesi

Not: Fare2 ve Fare3 kuklalarının komutları da fare1 kuklası ile aynıdır. Sadece başlangıç noktaları farklı konumlar olarak belirlenmiştir.

8.8: Hesap Makinesi

Sahne tasarımı şekil 112'de gösterilen uygulama bir hesap makinesidir. Şekilde görülen 0-9 arasındaki rakamlar scratch kütüphanesinde bulunmaktadır ve oradan hazır alınmıştır. Toplama, çıkarma, çarpma ve bölme işlemleri için ayrı birer buton kullanılmış olup işlem sonucunu görmek için eşittir, mevcut işlem sonucunu ve ekranı sıfırlamak için C butonu kullanılmıştır.

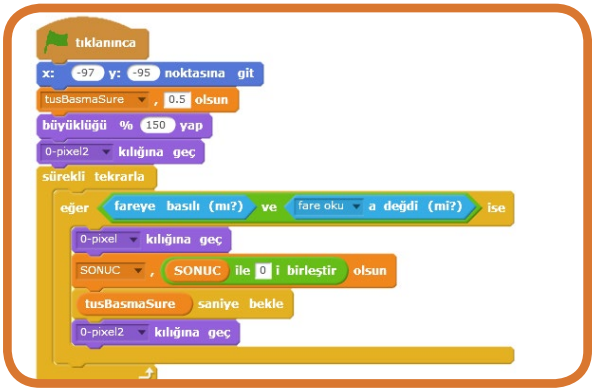


Şekil 112: Hesap makinesi sahne tasarımı

Hesap makinesinin çalışması için sahnedeki tuşlara basılarak hesap makinesinin ekranına istenilen sayı yazılır. Ardından yapılmak istenen işlem için buton (toplama, çıkarma, çarpma veya bölme) seçilir. Bu esnada ekran yeni sayı için temizlenir. Yeni sayı da yazıldıktan sonra eşittir butonuna basılarak sonuç ekranda görüntülenir. Ekran herhangi bir aşamada temizlemek istenirse C butonuna basılır.

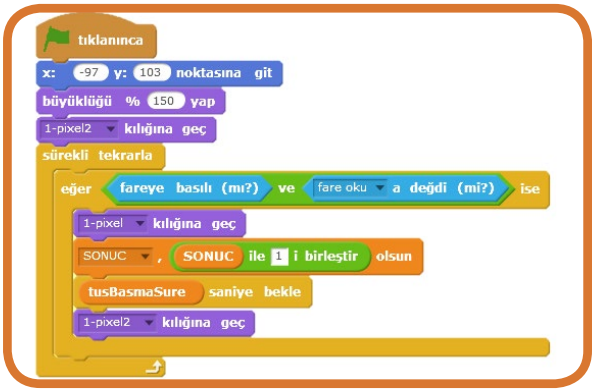
Rakamların Kodları:

0 Tuşunun Kodları: Şekil 113'de 0 tuşunun komutları gös¬terilmektedir. Öncelikle 0 tuşu sahnede sabit bir noktaya taşınır. Ardından "tusBasmaSure" değişkeninin değeri 0.5 saniye yapılır. 0 tuşunun büyüklüğü%150 yapılır, ardından 0 tuşunun hazırlanan ikinci kılığa geçmesi sağlanır. Uygulamada rakamların görevi; ekranda görüntülenen sonuç değişkenine eklenmektir. Bunun için fare ile her butona basıldığında önce 0 tuşunun ilk kılığına geçiş yapılır ardından sonuç değişkeninin mevcut değeri 0 değeri ile birleştirilir. "tusBasmaSure" değişkeninin değeri kadar bir süre beklenir. Bu bekleme süresinin amacı 0 tuşuna fare ile her tıkladığımızda sadece 1 kere basılan tuşun sonuç değişkenine eklenmesidir. Daha sonra başlangıç kılığına geçiş yapılır.

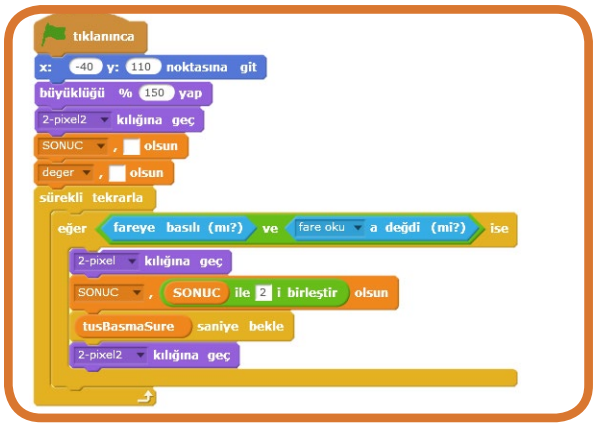


Şekil 113: 0 tuşunun komutları

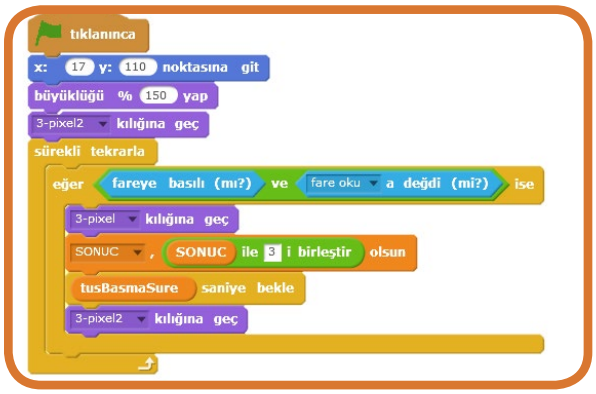
Sırası ile 1-2-3-4-5-6-7-8 ve 9 nolu tuşların da komutları aşağıda belirtilmiştir.



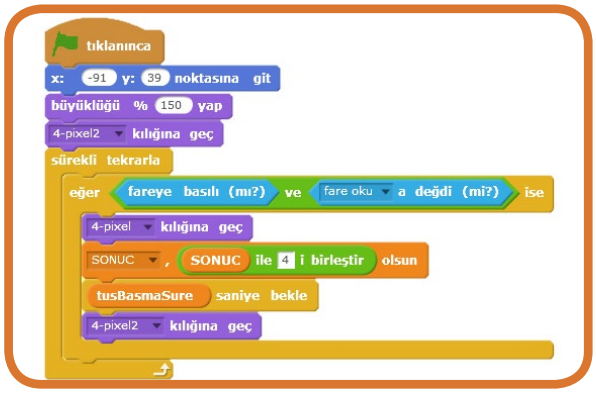
Şekil 114: 1 nolu tuşun komutları



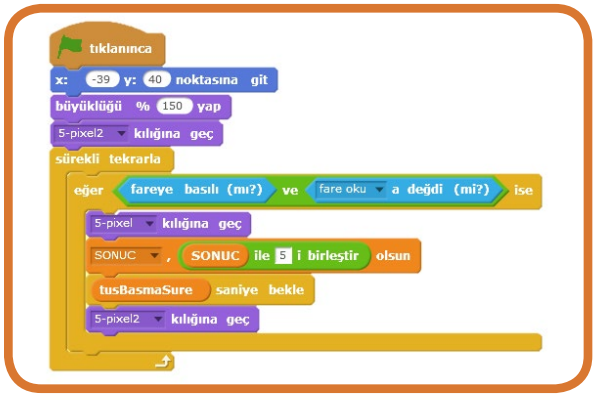
Şekil 115: 2 nolu tuşun komutları



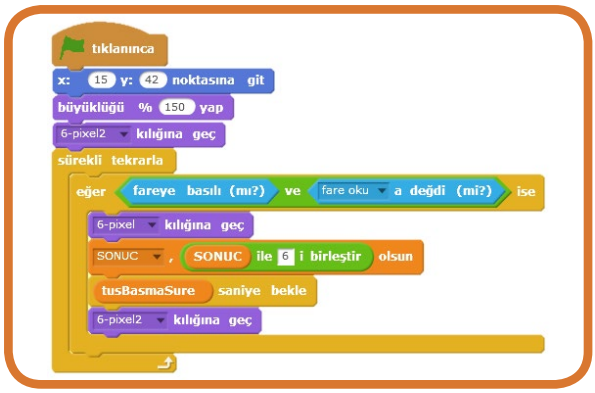
Şekil 116: 3 nolu tuşun komutları



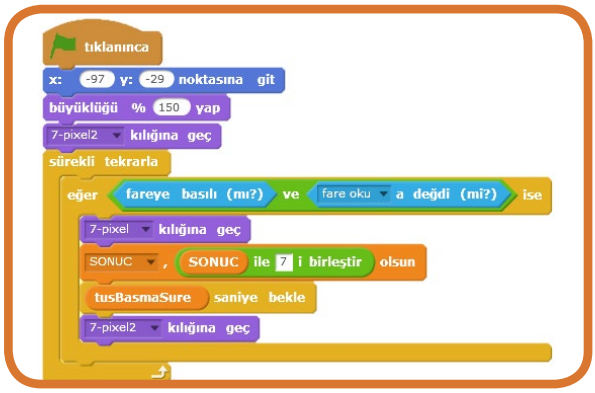
Şekil 117: 4 nolu tuşun komutları



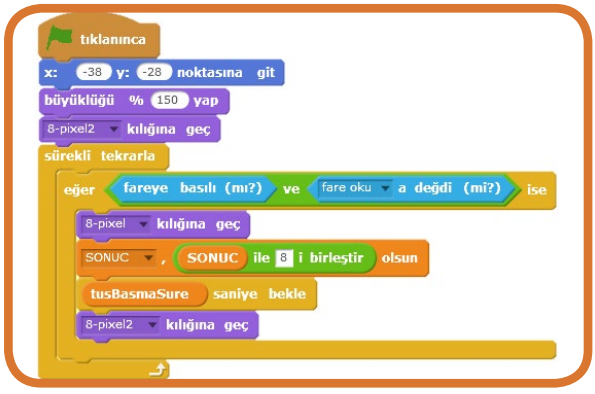
Şekil 118: 5 nolu tuşun komutları



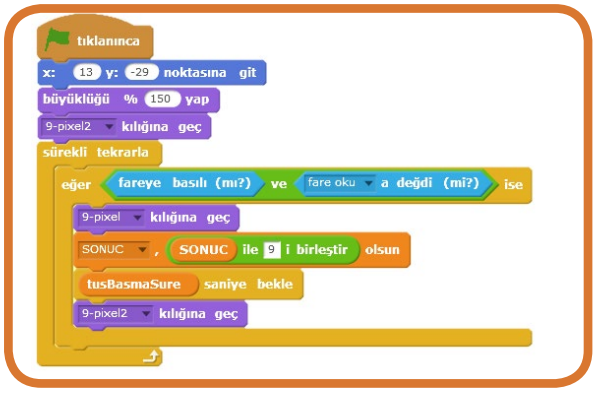
Şekil 119: 6 nolu tuşun komutları



Şekil 120: 7 nolu tuşun komutları

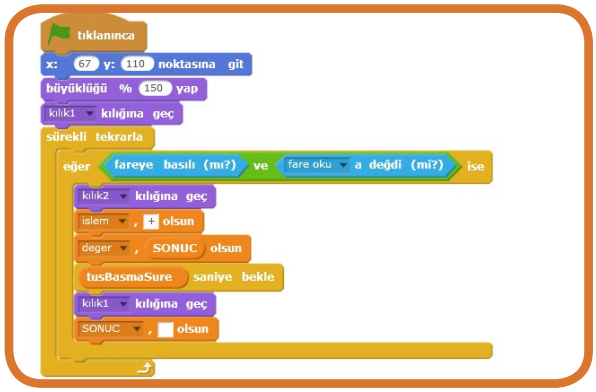


Şekil 121: 8 nolu tuşun komutları



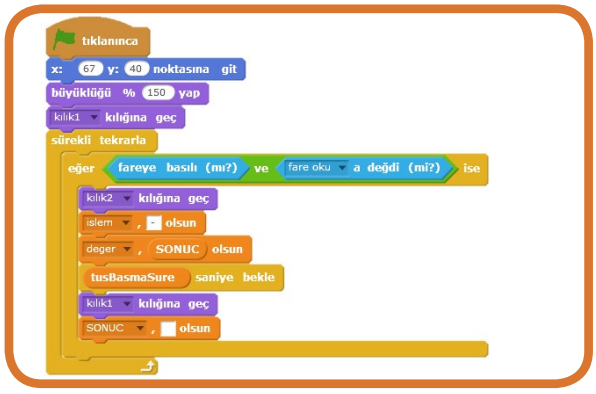
Şekil 122: 9 nolu tuşun komutları

Şekil 123'de artı tuşunun komutları gösterilmiştir. Dört işlem yapılmak istendiğinde "işlem" adında bir değişkene seçilen işlemin karakteri atanır. + tuşuna basıldığında işlem değişkeninin değeri + olacaktır. Ardından mevcut sonuç değişkeni, değer değişkenine aktarılacaktır. Daha sonra sonuç değişkeni temizlenir. Rakam tuşlarında olduğu gibi dört işlem tuşlarında da her tuşa basıldığında tuş için hazırlanmış diğer kılığa geçiş yapılır. "tusBasmaSure" kadar beklenerek birden fazla aynı işlemin gerçekleşmesi engellenir.



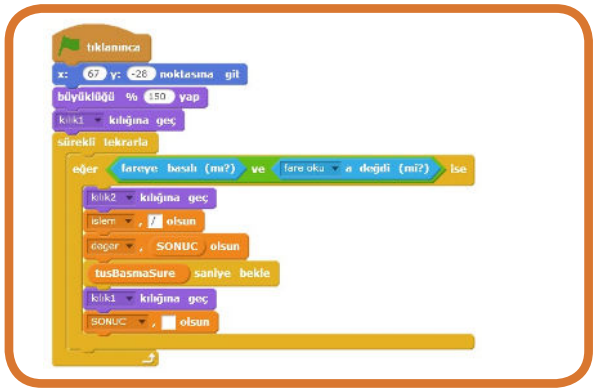
Şekil 123: artı tuşunun komutları

Şekil124'de eksi tuşunun komutları gösterilmiştir.



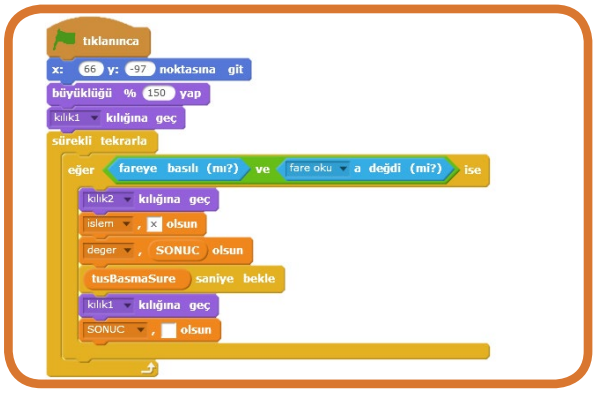
Şekil 124: eksi tuşunun komutları

Şekil 125'de bölü tuşunun komutları gösterilmektedir.



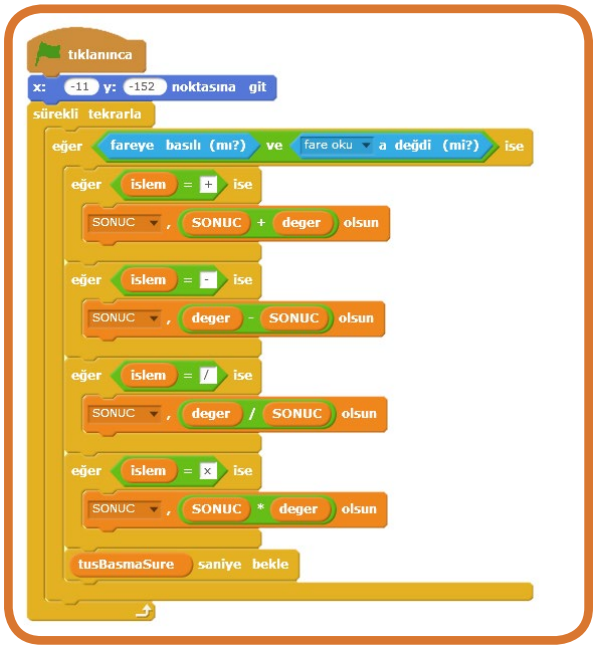
Şekil 125: bölü tuşunun komutları

Şekil 126'da çarpı tuşunun komutları gösterilmektedir.



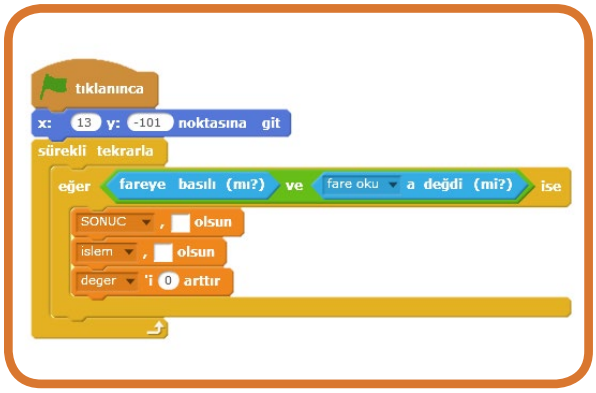
Şekil 126: çarpı tuşunun komutları

Eşittir tuşunun komutları şekil 127'de gösterilmiştir. Eşittir tuşunun görevi işlem değişkeninin değerine bakarak kullanıcının yapmak istediği işlemi yerine getirmektir. Eşittir tuşunun komutlarını incelediğimizde eğer şart ifadesi ile islem değişkeninin değeri kontrol edilmektedir. İşlem değişkenin değeri + ise değer ve sonuç değişkeni¬nin değeri toplanır. İşlem değişkeninin değeri – ise değer değişkeninin değerinden sonuç değişkeninin değeri çıkarılır. İşlem değişkeninin değeri / ise değer değişkeninin değeri sonuç değişkeninin değerine bölünür. İşlem değiş¬keninin değeri x ise değer değişkenin değeri ile sonuç değişkeninin değeri çarpılır. Her halükarda yapılan işlemin sonucu sonuç değişkeninde gösterilir.



Şekil 127: Eşittir tuşunun komutları

Şekil 128'de C tuşunun komutları gösterilmektedir.



Şekil 128: C tuşunun komutları

BÖLÜM

PROGRAMLANABİLİR ELEKTRONİK DEVRE KARTI

BÖLÜM 9: PROGRAMLANABİLİR ELEKTRONİK DEVRE KARTI

Bu bölümde scratch kullanılarak yapılan uygulamaların, bilgisayar ortamından gerçek dünyaya uyarlanabilmesini sağlayan elektronik devre kartları incelenecek ve programlanacaktır. İnteraktif uygulamalar geliştirilerek sahnede bulunan kuklaların gerçek dünyadaki nesneleri kontrol edebildiği görülecektir. Dış dünyadaki sensörlerden okunan bilgilerin sahne üzerindeki kuklalar tarafından görüntülenmesi sağlanacaktır.

9.1: Programlanabilir Elektronik Devre Kartı

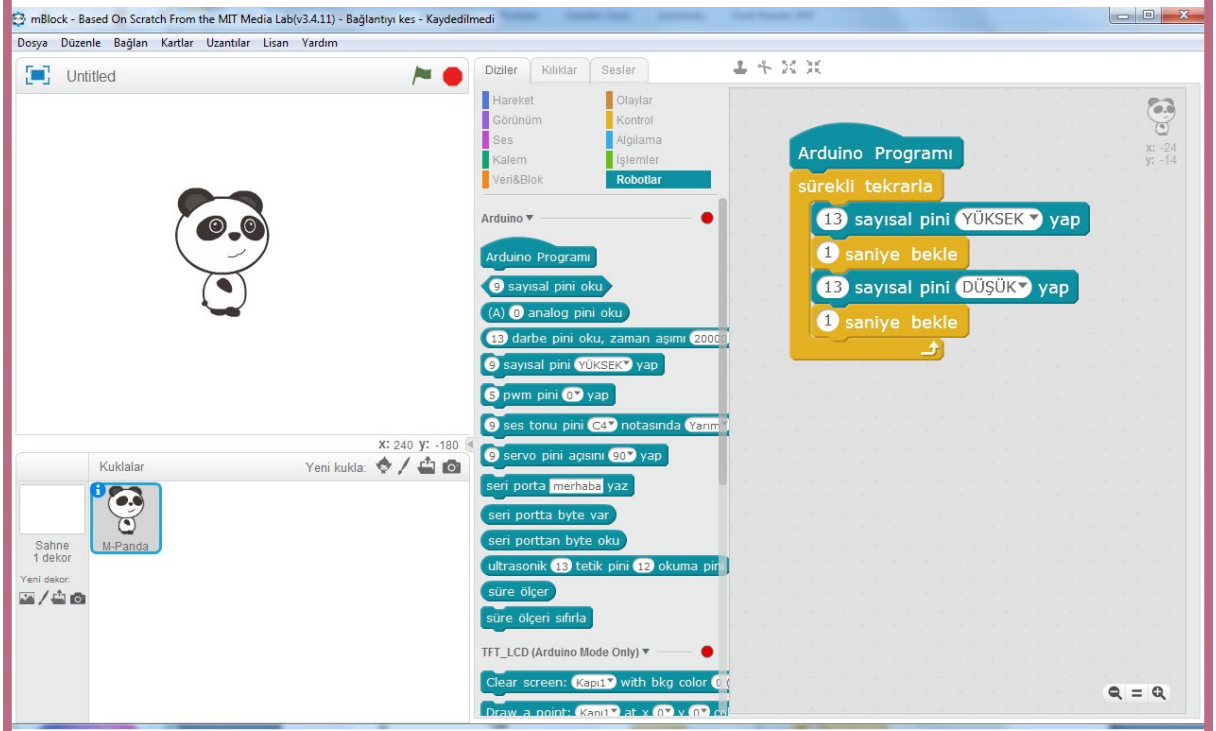
Arduino kartları bu bölümde incelenecektir. Arduino; açık kaynak olarak üretilen programlanabilir elektronik devre kartlarıdır. www.Arduino.cc adresinden detaylarını inceleyebilirsiniz. Birçok çeşidi bulunmakla birlikte Arduino Uno kartını bu bölümde incelenecektir. Şekil 129'da Arduino Uno kartı gösterilmektedir.

Şekil 129: Arduino Uno kartı

Programlanabilir elektronik devre kartları dijital giriş çıkış pinleri ve analog giriş pinlerine sahiptir. Bu pinler kullanılarak elektronik devre kartının dış dünya ile iletişimi sağlanabilir. Elektronik devre kartına yazılacak kodlar ile sensörlerden bilgi okuyabilir, elektronik devre elemanlarını kontrol edebiliriz.
Şekil 127'de gösterilen elektronik devre kartı 14 adet dijital giriş/çıkış pinine, 6 adet de analog giriş pinine sahiptir. Bu pinler 0 numarası ile başlar.

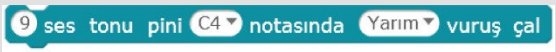
9.2: Elektronik Devre Kartının Programlanması

Arduino kartlarını programlamak için, metin tabanlı ve blok tabanlı programlama araçları kullanılabilir. Metin taban¬lı programlama aracı olarak C/C++ dilini kullanan Arduino ide yazılımını www.Arduino.cc adresinden; blok tabanlı programlama için olan mBlock yazılımını https://www.makeblock.com/software/mBlock3/downloads adresinden temin edebilirsiniz. Yazılım scratch tabanlı ve ücretsiz ve açık kaynak kodludur. Scratch arayüzüne ek olarak Robotlar adında bir blok kategorisine sahiptir. Bu yazılım ile kod blokları kullanılarak Arduino kartınıza hazırladığınız uygulamaları yükleyebilir ve çalıştırabilirsiniz. Şekil 130'da mBlock yazılımı ara yüzü gösterilmektedir.



Şekil 130: mBlock yazılımı ara yüzü

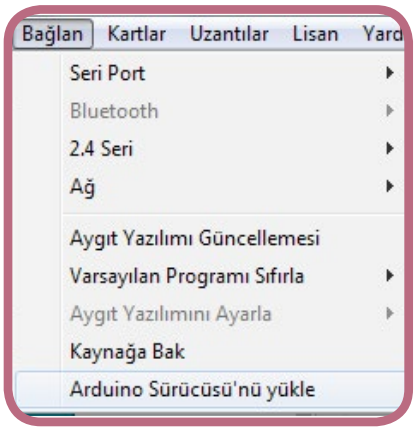
Tablo 15'de mBlock yazılımı Robotlar kategorisinde bulunan kodlar ve açıklamaları gösterilmektedir.

Blok	Açıklama
	Arduino kartına kod yazmak için kullanılan başlangıç bloğudur. Offline mod da kullanılır.
	Belirtilen dijital pini okur. Geri dönüş değeri 1 ya da 0 'dır
	Belirtilen analog pini okur. Geri dönüş değeri 1 ya da 0 'dır
	13 nolu pin lojik düşük seviyede tutuluyorken, bu pine gelen lojik yüksek seviyedeki sinyal sayını 20000 mikrosaniye (20 saniye) boyunca sayar.
	Belirtilen dijital pin Değerini yüksek/düşük yapar.
	Belirtilen pwm pinine 0-255 arasında değer verebilir.
	Belirtilen pine bağlı buzzer'dan nota çalmasını sağlar.
	Belirtilen pine bağlı servo motor açısını0-180 arasında değiştirir.
	Seri porta belirtilen ifadeyi yazar.
	Seri porttan veri gelip gelmediğini sorgular.
	Seri porttan byte karakter okunmasını sağlar.
	Ultrasonik sensörün Echo(okuma) ve Trig(tetik) pinlerinin, belirtilen pinlere bağlanarak sensörün ölçtüğü mesafeyi geri döndürmesi sağlanır.
	Süre ölçer; kod çalışmaya başladığında bir sayaç da çalışmaya başlar. Bu sayacın görevi uygulamanın çalışmaya başladığı andan süre ölçer komutunun kullanıldığı ana kadar geçen süreyi hesaplamaktır. Hesaplanan değer milisaniye birimindendir.
	Süre ölçeri sıfırlar, böylece istediğimiz yerde süre ölçer değerini 0'dan başlatarak kullanabiliriz.

Tablo 15: Robotlar kategorisi kodları

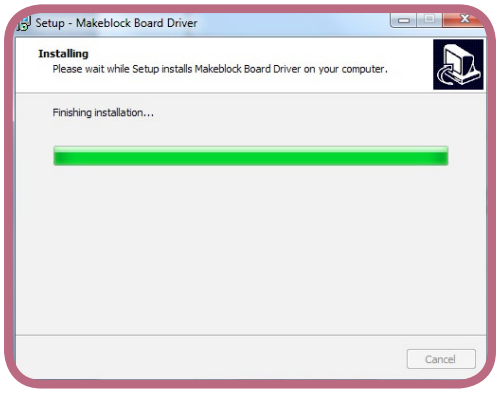
9.3: Bilgisayara Bağlantı ve Kod yükleme

Arduino kartı bilgisayarın usb portuna takıldığında öncelikle sürücü yüklemeniz beklenecektir. Sürücüyü yüklemek için Şekil 131'de görüldüğü gibi Bağlan menüsünde bulunan "Aygıt sürücüsünü yükle" sekmesine tıklanır.



Şekil 131: Arduino sürücüsü'nü yükle ile sürücü yükleme

Açılan pencerede istenilen şartlar kabul edilerek ileri butonu ile Arduino kartı sürücülerinin bilgisayara kurulması sağlanır.

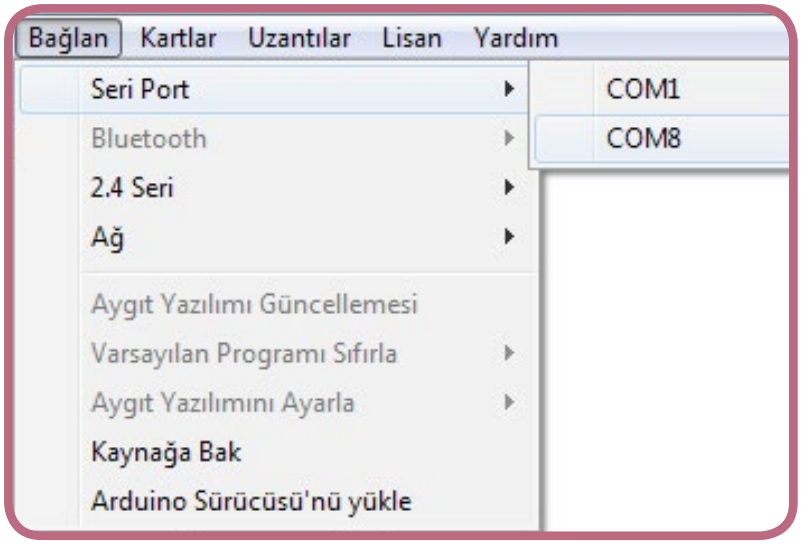


Şekil 132: Arduino sürücülerinin yükleme penceresi

Arduino kartının sürücüleri bilgisayara yüklendikten sonra, mBlock programı ile program yazabilmek için;
1:Kartlar menüsünden Arduino kartı seçilir.

Şekil 133: Kartlar menüsünden Arduino kartı seçimi

2:Bağlan menüsünden Arduino kartının iletişim kuracağı com port seçilir.



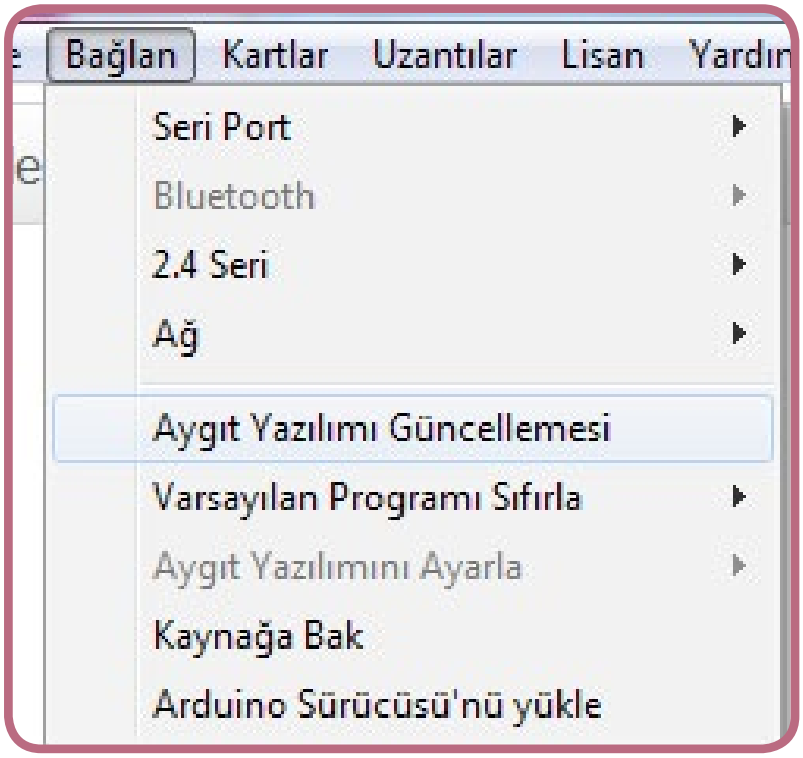
Şekil 134: Bağlan menüsünden Arduino kartının bağlı olduğu port seçimi

Arduino ile mBlock'u interaktif mod ve offline mod olarak iki farklı çalışma modunda kullanabiliriz.

9.3.1: İnteraktif Mod

İnteraktif modda mBlock yazılımını kullanarak Arduino ile iletişim sağlanabilir. Arduino kartına bağlı bir sensörden alınan veriler, sahnede bulunan karakterler tarafından görüntülenebilir. İnteraktif çalışma modunda sahnede oluşturulan her eylem Arduino kartı ile etkileşim sağlayabilir.

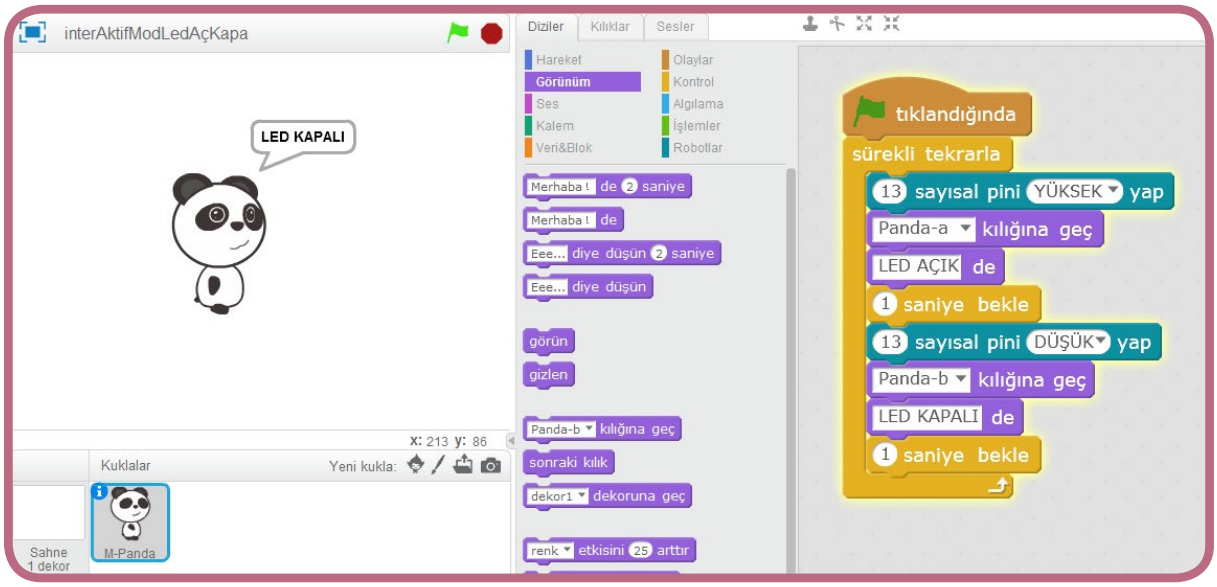
İnteraktif modda çalışmak için Arduino kartına bir iletişim programının yüklenmesi gerekmektedir. Böylelikle Arduino kartı bilgisayarınıza bağlı olduğu sürece, mBlock sahne-sindeki kuklalar ile etkileşim sağlayabilirsiniz. Bunun için bağlan menüsünden Aygıt yazılımı Güncellemesine tıklanır.



Şekil 135: İnteraktif mod da çalışmak için Aygıt Yazılımı Güncellemesinin yapılması

Bu işlem tamamlandığında interaktif modda çalışmaya başlayabilirsiniz.

İnteraktif Mod Etkinliği:
Bu etkinlikte Arduino kartı interaktif modda çalıştırılacaktır. Böylelikle scratch kodları ile arduino kartının birlikte çalışması sağlanmış olacaktır. Etkinliğin sahne tasarımı şekil 136'da gösterilmiştir.



Şekil 136: İnteraktifMod led aç kapa uygulaması

Etkinlikte Arduino kartların 13 nolu dijital pinine bağlı olan bir led diyotun, sahnedeki Panda kuklası ile etkileşimli olarak açılıp kapanması sağlanacaktır. Bunun için 13 nolu dijital pin YÜKSEK olduğunda, bu pinden 5V gerilim; 13 nolu dijital pin Düşük olduğunda ise bu pinden 0V gerilim elde edilecektir.
Şekil 137'de kodlar gösterilmektedir.

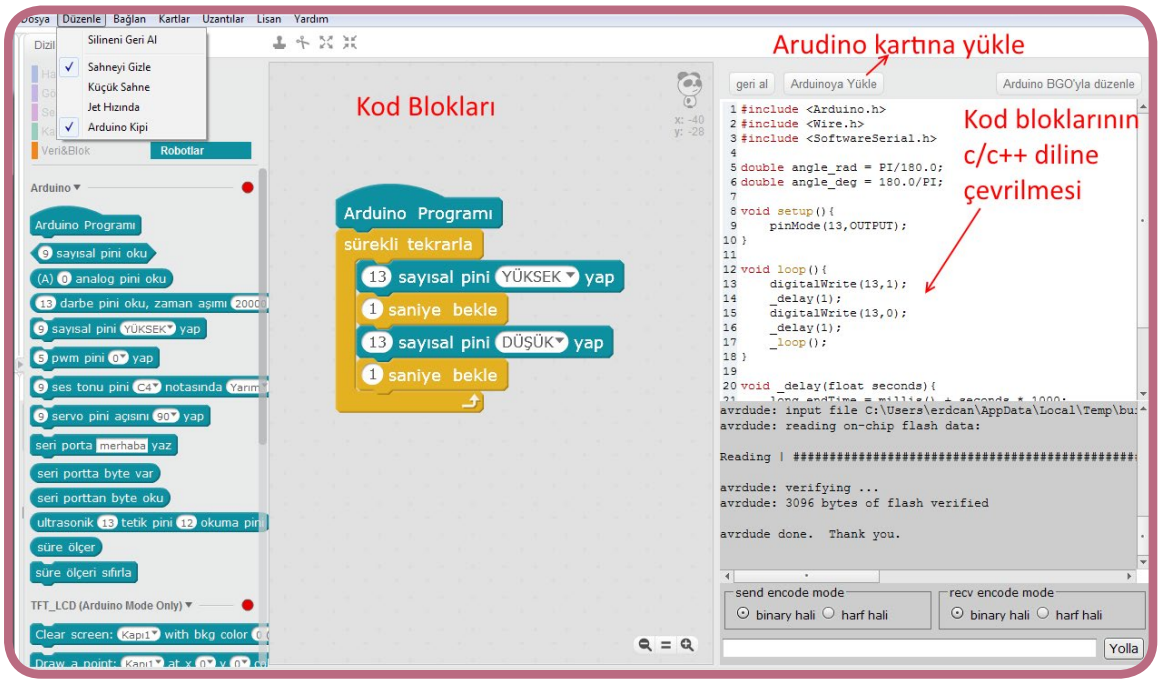
Şekil 137: İnteraktif mod led aç kapa uygulaması

Scratch uygulamalarında daha önce de yapıldığı üzere yeşil bayrak bloğu ile uygulama başlatılıyor. Şekil 137'de gösterildiği üzere "sürekli tekrarla" bloğu içerisinde yer alan "13 sayısal pini Yüksek yap" kod bloğu, Arduino kartının 13 no lu pinine bağlı olan ledin yanmasını sağlayacaktır. Ardından pandanın kılığı değişecek ve panda "LED AÇIK" diyecektir. 1 saniye bekle komutundan sonra bu sefer "13 no lu pini Düşük yap" komutu çalışacak ve led sönecektir. Panda kılığı değişecek ve panda "LED KAPALI" diyecektir. 1 saniyelik beklemeden sonra süreç tekrar tekrar devam edecektir.

İnteraktif modda uygulama yapılabilmesi için;
1- Arduino kartın bilgisayara usb üzerinden bağlı olması,
2- Kartlar menüsünden Arduino Uno kartın seçili olması,
3- Bağlan menüsünden Arduino kartının bağlı olduğu Seri Portun seçilmesi,
4- Uygulamayı çalıştıracak kod bloklarının yazılması gerekmektedir.

9.3.2: Offline Mod (Arduino Kipi) Çalışma

Ardunio kartları ile çalışırken her zaman bilgisayara bağlı olmak gerekmemektedir. Çizgi izleyen robot veya sumo robot yapmak istediğinizde bilgisayara bağlı olarak çalışmak robotun şartlarını sınırlandırabilir. Bunun için offline mod veya Arduino kipi olarak adlandırılan bir diğer çalışma yöntemi tercih edilmelidir. Bu kısımda bu yöntemden sizlere bahsedilecektir. Ardunio kipine geçmek için Düzenle menüsünden Arduino kipi sekmesi seçilir ve sahneyi gizle sekmesi ile de sahne gizlenir.
Offline çalışma moduna geçildiğinde mBlock programında kullanılan sahne gizlenir. Bu mod da çalışırken hazırlanan kod blokları C/C++ diline çevrilir. **"Arduino Kartına Yükle"** butonu kullanılarak Arduino kartının hafızasına, derlenen hex dosyası kaydedilmiş olur. Böylelikle Arduino kartının bilgisayara bağlı olarak çalışmasına gerek kalmaz. Enerjisini sağlandığı sürece Arduino kartı, hafızasındaki kodları sürekli olarak çalıştıracaktır. Şekil 138'de Arduino kipi gösterilmektedir.



Şekil 138: Arduino kipi

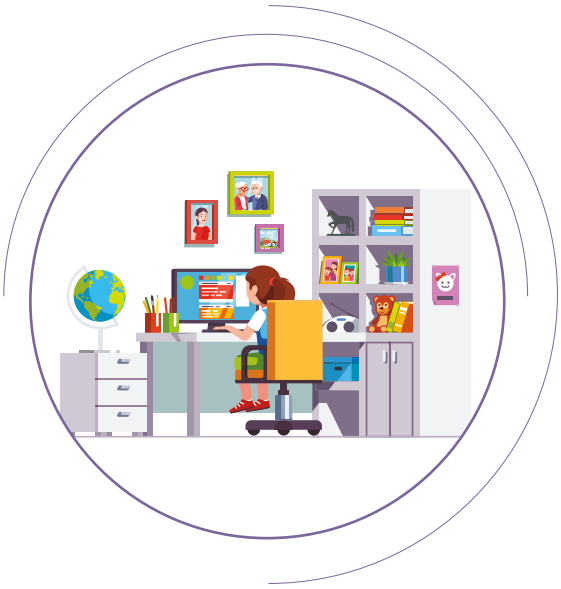
Arduino kipini incelediğimizde, dikkat çeken husus; Arduino kartıyla ilgili donanımlara ulaşmak istendiğinde Robotlar kategorisindeki kod bloklarını kullanılmasıdır. Uygulamanın başlangıç olayı "Arduino programı" kod bloğudur. Bu bloğun altına yazılacak kodlar, sahnenin sağ tarafında görüleceği üzere C/C++ diline çevrilmektedir. Kodların Arduino kartına yüklenmesi için;
1- Kartlar menüsünden Arduino Uno kartı seçilir
2- Bağlan menüsünden Arduino kartının bağlı olduğu seri port seçilir.
3- Kodlar hazırlandıktan sonra Arduino'ya Yükle butonuna basılarak kodların yüklenmesi sağlanır.
Oluşturulan C/C++ dili üzerinde bir değişiklik yapılmak istenirse Arduino BGO ile düzenle butonuna tıklanır, mBlock yazılımı ile yüklü gelen Arduino Ide açılır ve bu sayede kod üzerinde istenilen değişiklikler C/C++ dilinde yapılabilir.

Arduino Kipi Etkinliği:
Önceki bölümde yaptığımız etkinliği Arduino kipi etkinliği için de tekrarlayacağız ve
Arduino kartının 13 numaralı pinine bağlı olan led diyotun açılıp kapanmasını sağlayacağız. Kullanılacak kodlar şekil 139 da gösterilmiştir.

Şekil 139: Arduino kipi etkinliği

Uygulamada 13 numaralı dijital pinin 1 saniye aralıklar ile Yüksek ve Düşük yapılması sağlanmıştır. Bu sayede enerjisi olduğu sürece Arduino kartının 13 numaralı dijital pine bağlı ledin yanıp söndüğünü görürüz.
Arduino kipinde yer alan kod blokları kategorilerinden sadece; veri, kontrol, işlemler ve robotlar kategorileri aktif, diğer kod kategorileri pasif durumdadır.

BÖLÜM



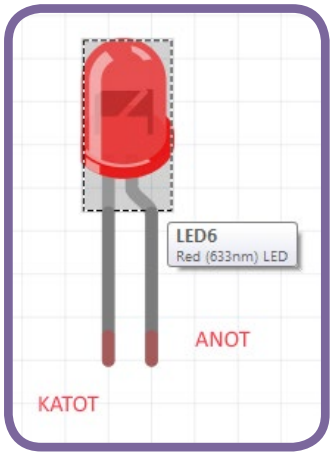
ELEKTRONİK DEVRE KARTI UYGULAMALARI

BÖLÜM10: ELEKTRONİK DEVRE KARTI UYGULAMALARI

Bu bölümde elektronik devre kartı uygulamaları incelenecektir. Konu içerisinde elektronik devre elemanlarının çalışmasından kısmen bahsedilecektir.

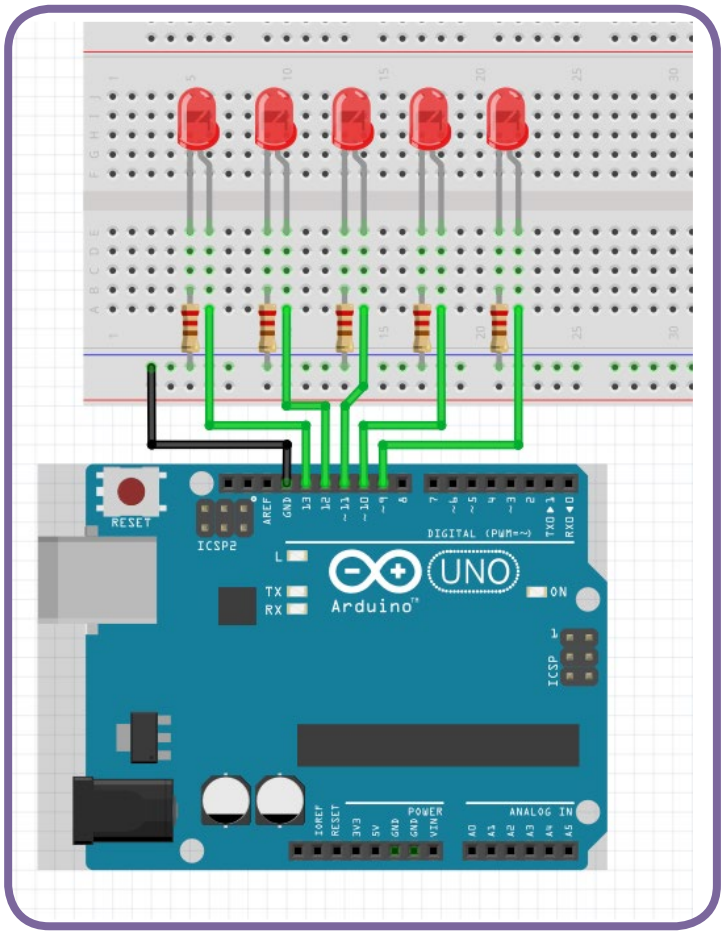
10.1: Yürüyen Işık Devresi

Yürüyen ışık devresi LED diyotlar ile yapılan basit bir ışık animasyon devresidir. Elektronik devre kartına gerekli bağlantılar yapıldıktan sonra kod yazılarak uygulama çalıştırılır. Öncelikle LED diyotun çalışmasını kısaca inceleyelim. Şekil 140'da LED diyot gösterilmektedir.



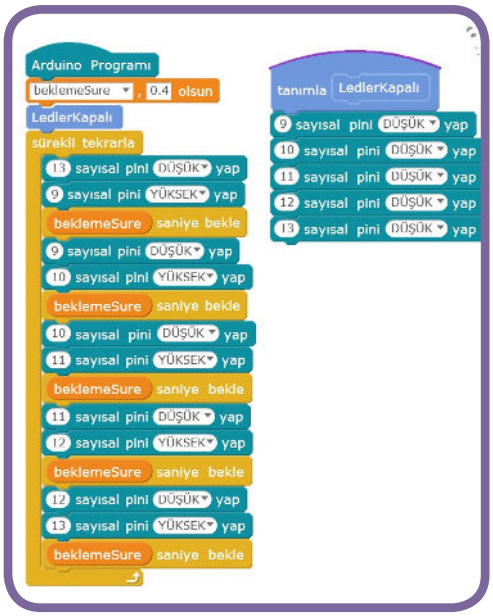
Şekil 140: LED diyot

LED diyot; ışık yayan diyot anlamındadır. Diyotlar en temel yarı iletken elektronik devre elemanlarıdır, şekil 140'da görüldüğü üzere iki adet bacağa sahiptir. Bu bacaklar anot ve katot olarak isimlendirilir. Tek yönde elektrik akımını iletir. Diyotların iletime geçmesi için gerilim kaynağının + ucu diyotun anoduna, gerilim kaynağının eksi ucu ise diyotun katoduna bağlanır. Bu şartlar sağlandığında diyot iletime geçer. LED diyot iletime geçtiğinde ışık yayar. İletime geçebilmesi için 0.7V'luk gerilim yeterlidir. Şekil 141'de devre şeması gösterilmektedir. Devrede 220 Ohm'luk direnç kullanılmıştır.



Şekil 141: Yürüyen ışık devre şeması

Şekil 141'de görüldüğü üzere 5 adet LED in anot uçları elektronik devre kartının 9, 10, 11, 12 ve 13 nolu pinlerine bağlanmıştır. LED diyotlarının katot uçları ise bir direnç aracılığıyla arduino kartının GND pinine bağlanmıştır. Devrenin çalışması için şekil 142'de gösterilen komutları arduino kipinde elektronik devre kartına yüklemeniz gerekmektedir. Kodları incelediğimizde Ledler kapalı adında bir kod bloğu tüm LEDlerin bağlı olduğu pinleri düşük yaparak LEDlerin sönmesini sağlar. Ardından sırasıyla 9, 10, 11, 12 ve 13 nolu pinler yüksek yapılarak LEDlerin sırasıyla yanıp sönmesi sağlanır. Bekleme süresi değişkeninin değeri ayarlanarak LEDlerin ne kadar süre yanık kalacağı belirlenir.



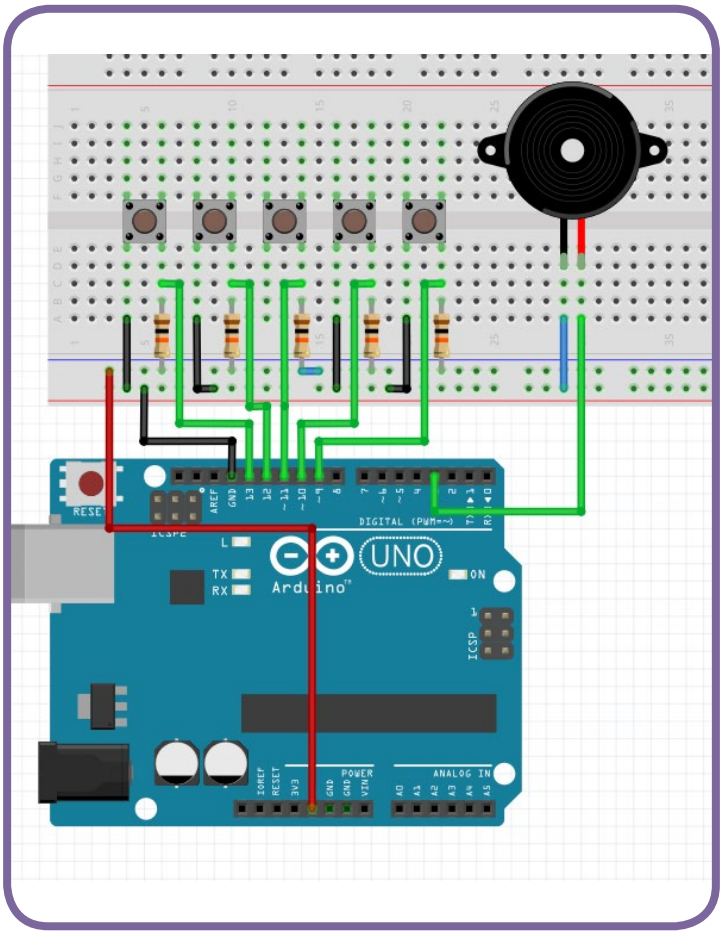
Şekil 142: Yürüyen ışık kodları

10.2: Piyano Yapımı

Bu uygulama da piyano yapımı incelenecektir. 5 adet buton kullanılarak devreye bağlı olan buzzerdan belirli notaların çıkması sağlanacaktır. Butonların bir ucu GND'ye bağlı iken diğer uçları elektronik devre kartının 9,10, 11, 12 ve 13 nolu pinlerine bağlı olacaktır. Ayrıca bu pinler 10K'lık direnç ile butona basılmadığında yüksek konuma çekilmiştir. Dirençlerin diğer uçları 5V'a bağlıdır.

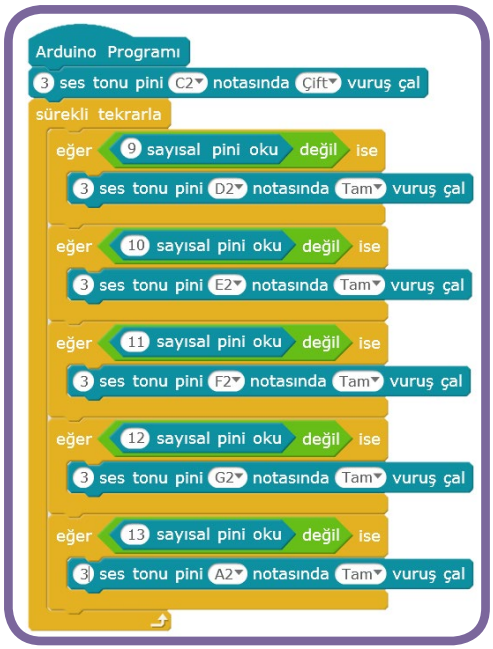
Şekil 144'de devre şeması gösterilmektedir.

Şekil 143: Buton devre elemanı



Şekil 144: Piyano devre şeması

Şekil 145'de piyano devresinin kodları gösterilmektedir. Arduino kipine geçip kodların elektronik devre kartına yük lenmesi gerekmektedir.

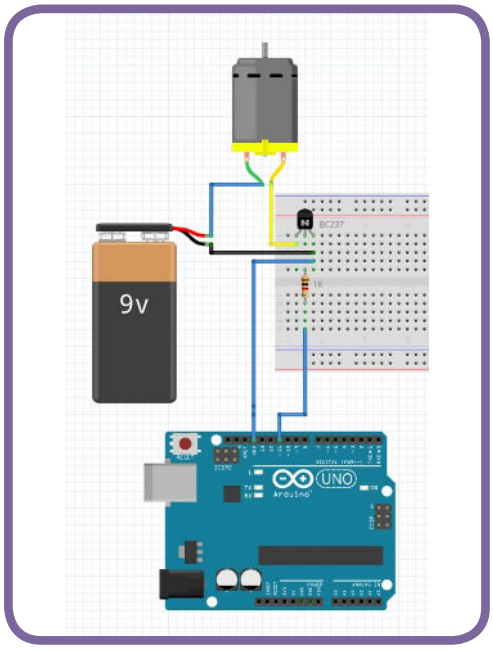


Şekil 145: Piyano devresi kodları

Piyano devresinde kodların çalışabilmesi için butonlara basılması gerekir. 9 nolu pine bağlı olan butona basıldığında buzzerdan D2 notasında tam vuruş, 10 nolu pine bağlı butona basıldığında E2 notasında tam vuruş, 11 nolu pine bağlı butona basıldığında F2 notasında tam vuruş, 12 nolu pine bağlı butona basıldığında G2 notasında tam vuruş, 13 nolu pine bağlı butona basıldığında ise A2 notasında tam vuruş çalınır.

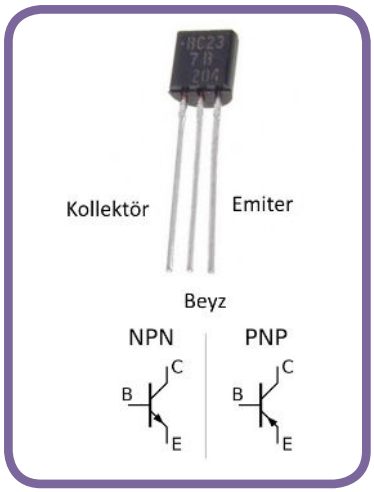
10.3: Motor Hız kontrolü

DC motorlar bilindiği üzere elektrik enerjisini hareket enerjisine çevirirler. Motor hızının ayarlanması için elektronik devre kartında bulunan PWM olarak isimlendirilen pinlerden faydalanılır. Şekil 146'da motor hız kontrol devresi görülmektedir.



Şekil 146: Motor hız kontrol devresi

Devrede transistör olarak adlandırılan yarı iletken elektronik devre elemanı kullanılmıştır. Transistörün 3 bacağı bulunur. Bunlar Emiter, Kollektör ve Beyz bacaklarıdır. Beyz bacağı transistörün kontrol bacağıdır ve beyz bacağına uygulanacak gerilim ile transistorün iletime geçmesi veya kesime gitmesi sağlanır. Transistör iletime geçtiğinde Kollektörden emiter ucuna doğru elektrik akımının geçişi sağlanır. Şekil 147'de transistorün bacak bağlantıları gösterilmiştir.



Şekil 147: BC 237 transistörü bacak bağlantıları

Elektronik devre kartının 3, 5, 6, 9, 10 ve 11 no lu pinlerinin PWM özelliği bulunmaktadır. Bu pinler üzerinde ~ işareti bulunur. Dijital pin çıkışı yüksek yapıldığında bu pinden 5V gerilim, düşük yapıldığında ise 0V ge¬rilim elde edilir. Bu şekilde motor hızı kontrol edilmek istendiğinde, dijital pin yüksek yapıldığında motor tam hızda çalışır, düşük yapıldığında ise motor çalışmaz.
PWM, pulse width modulation- palse genişlikli modülasyon olarak bilinen bir modülasyon çeşididir. PWM pinlerinin özelliği bu pin üzerindeki gerilimi dijital olarak kontrol edilmesini sağlamasıdır. Kontrol sistemlerinde sıklıkla kullanılır. Bu proje de pwm pinine verilecek 0-255 arasındaki değer ile motorun hızını kontrol edeceğiz.
Şekil 148'de devrenin çalışması için gerekli kodlar gösterilmektedir. Motorun hızı yavaş yavaş arttırılarak son hıza ulaşacak ardından hız azaltılarak durdurulacaktır.

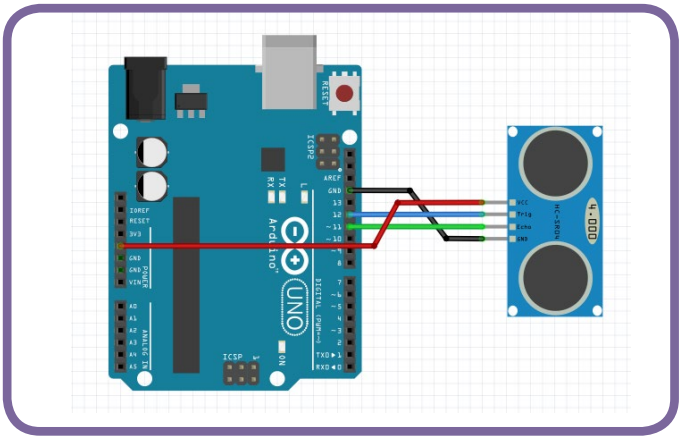
Bu amaçla hız adında bir değişken tanımlanarak hız değişkeninin değeri uygulama başladığında 0 olarak belirlenmiştir. Hız değişkeninin değeri 255 değerinden küçük olduğu sürece her 0.3 saniyede 10 arttırılmaktadır. Böylelikle 3 nolu pwm pinin değeri 255'e kadar arttırılacaktır. Bu sayede motorun hızı da arttırılmış olacaktır.

Motorun tam hıza ulaşmasından sonra hızı düşürmek için hız değişkeninin değeri 10 değerinden küçük olana kadar tekrarla bloğu içerisinde azaltılacaktır. Böylece motor hızı sürekli olarak arttırılıp azaltılacaktır.

Şekil 148: Motor hız kontrol devresi kodları

10.4: Ultrasonik Sensör ile Mesafe Ölçümü

Bu uygulamada HC-sr04 olarak isimlendirilen mesafe sensörünün çalışması incelenecektir. Sensör ultrasonik dalgaların gönderildiği verici ve ultrasonik dalgaları yakalayan alıcıdan oluşur. Böylelikle kendisinin önündeki engelle olan mesafesini ölçebilmektedir. Yarasaların hareket etme prensibine göre çalışır.
Şekil 149'da ultrasonik sensörün arduino kartına bağlantı şeması gösterilmektedir.



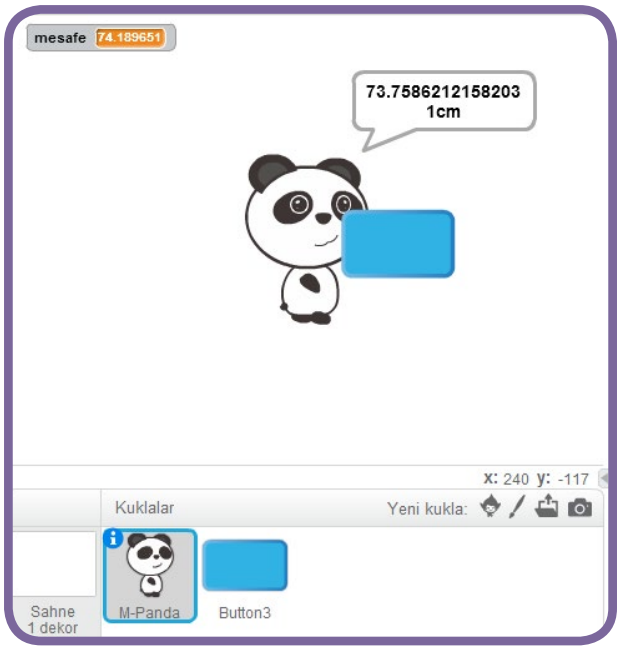
Şekil 149: Ultrasonik sensörün elektronik devre kartına bağlantı şeması

Ultrasonik sensörün VCC,GND Trig ve Echo pini olmak üzere 4 pini vardır. Bağlantı pinleri tablo16'da gösterilmiştir.

Pin adı	Arduino pini
VCC	5V
Trig	12
Echo	11
GND	GND

Tablo 16: Ultrasonik sensörün elektronik devre kartına bağlantı pinleri

Bu uygulama da elektronik devre kartı mBlock yazılımı ile online (interaktif) modda çalıştırılacaktır. Bu modun çalışma prensibi 9. Bölümde incelenmişti. Sahne tasarımı Şekil 150'de gösterilmektedir.



Şekil 150: Ultrasonik sensör mesafe ölçümü sahne tasarımı

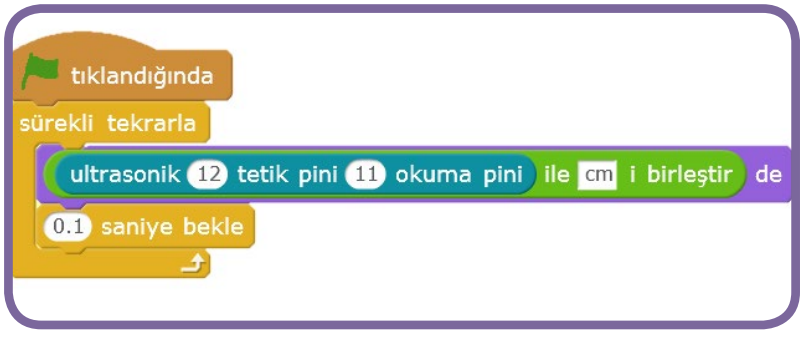
Panda Komutları:

Şekil 151: Mesafe ölçümü yapan pandanın komutları

Şekil 151'de mesafe ölçümü yapan pandanın komutları gösterilmektedir. Buna göre ultrasonik sensörden alınan mesafe bilgisi her 0.1 saniyede bir defa ölçülerek panda tarafından kullanıcıya gösterilir.

Buton3 Komutları:

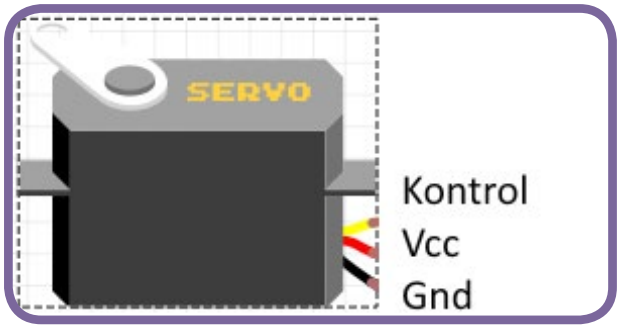
Şekil 152'de gösterilen komutlar, butonun sahne üzerinde yerleşimini göstermektedir. Buna göre ultrasonik sensörden alınan mesafe bilgisi buton3 kuklasının sahnedeki X eksenine göre yerini göstermektedir. Sahne üzerinde hem panda hem de buton 3 merkezi noktada bulunmaktadır. Uygulama çalıştırıldığında engele olan mesafeye göre buton 3 panda kuklasından uzaklaşmakta veya yakınlaşmaktadır.



Şekil 152: Buton3 kuklasının komutları

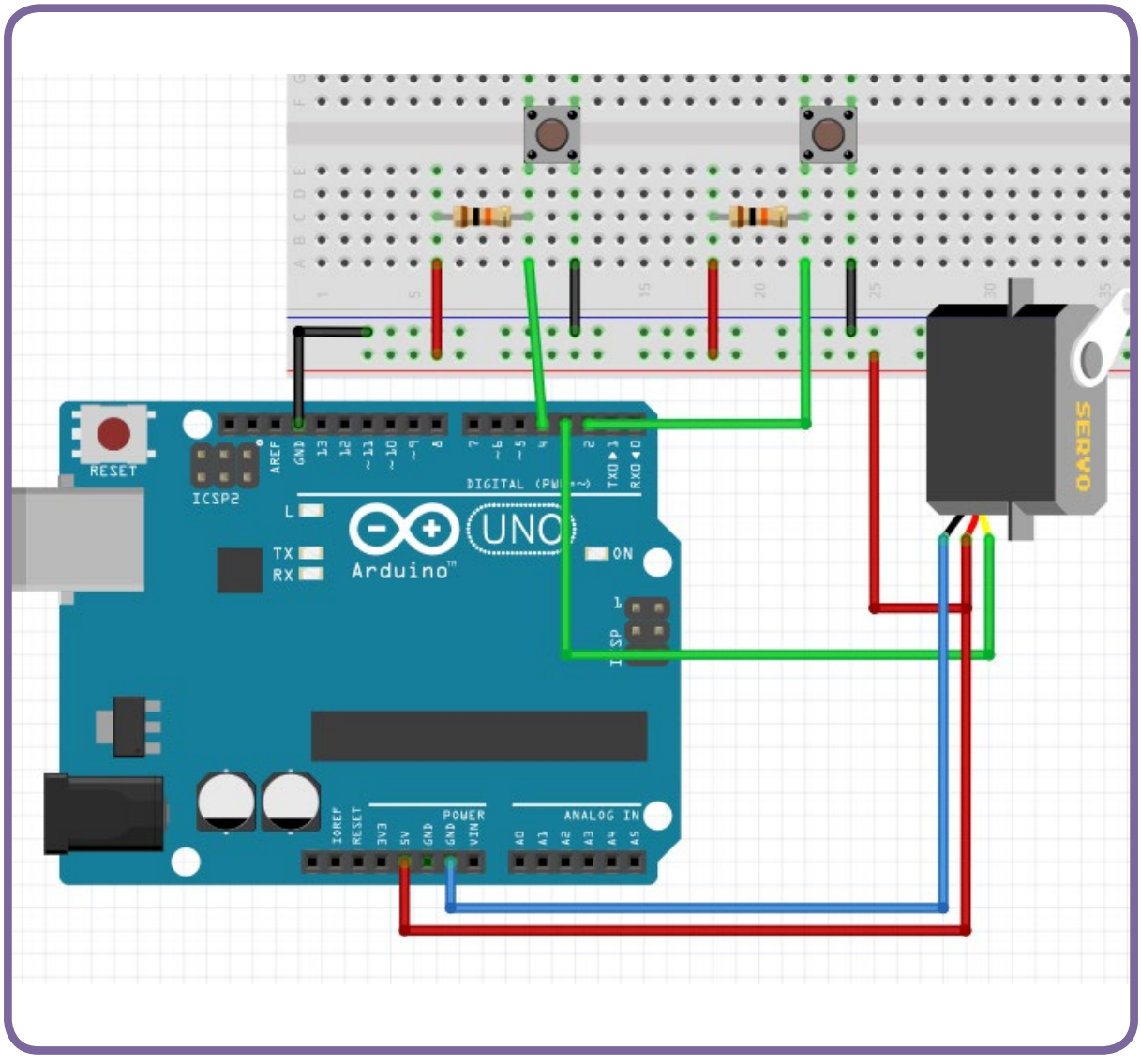
10.5: Servo Motor Kontrolü

Servo motorlar açı kontrollü motorlardır. Basit DC motorlar gibi sürekli dönebilen çeşitleri de bulunmaktadır. Bu bölümdeki uygulamada 0-180 derece arasında dönebilen servo motor kullanılacaktır. Motorun açı değerini elektronik olarak kontrol edebiliriz. Bu amaçla iki adet buton ile açı değerini arttırıp azaltacağız.



Şekil 153: Servo motor

Kontrol pini servo motorun açısını kontrol edebilmek için elektronik devre kartı pwm pinine bağlanır. VCC pini 5V, GND pini de GND'ye bağlanır. Şekil 154'de gösterilen devre şeması incelendiğinde iki adet buton kullanıldığı görülmektedir. Butonlar servo motorun açısını arttırıp azaltacaktır.



Şekil 154: Servo motor kontrolü Devre şeması

Şekil 155 deki kodlar incelendiğinde "aci" adında bir değişken oluşturulduğu ve değerinin 90 yapıldığı görülmektedir. Bu değerin arttırılıp azaltılması ise butonlar aracılığıyla olacaktır. 2 nolu dijital pine bağlı olan butona basıldığında eğerki "aci" değeri 180 'den düşük ise "aci" değeri 5 artırılacak; 4 nolu dijital pine bağlı butona basıldığında ise "aci" değeri 0'dan büyük ise 5 azaltılacaktır. Burada "aci" değerinin 0'dan büyük ve 180'den küçük olmasının sorgulanma nedeni servo motorun hareket edebileceği sınırların dışına çıkmamasıdır.

Şekil 155: Servo motor kontrolü uygulama kodları

SONUÇ

Kodlama konusunda öğrencilere ve öğretmenlere rehber olması amacıyla kodlama kılavuzu hazırlanmıştır. Basit, eğlenceli ve hayal edilen ürünlerin ortaya çıkmasına imkan sağlayan etkinlikler ile içerik desteklenmiştir.
Kılavuzun ilk bölümünde problemin tanımı yapılmıştır. Var olan bir sorunu çözebilmek için sorunun tanımlı hale getirilmesinin ne kadar önemli olduğu belirtilmiştir. Problem çözme becerisine sahip kişiler günlük hayatta sıklıkla karşılarına çıkan sorunların üstesinden gelebilmekte, sorunlara ilişkin akılcı ve pratik çözümler üretebilmektedir. Programcı açısından düşünüldüğünde problemin tanımlı hale gelmesi programcının yol haritasını belirlemesine yardımcı olmaktadır.

Problemi çözebilmek için algoritmalar hazırlanması ve akış şemalarının çizilmesinden bahsedilmiştir. Öğrencilerin herhangi bir konu hakkında algoritma hazırlaması; işe nasıl başlayacaklarını, karşılaşabilecekleri sorunları görebilmelerini ve yapmakta oldukları işin nasıl sonuçlanacağına ilişkin durumları zihinlerinde canlandırmalarına imkan sağlar. Sonraki bölümlerde programlama ve programlama dillerinden bahsedilmiştir. Blok tabanlı ve metin tabanlı programlama dillerinin kullanım kolaylıklarından; blok tabanlı programlamanın ortaokul öğrencileri ve programlama öğrenmeye yeni başlayanlar için etkili bir yöntem olduğuna değinilmiştir.

Scratch ile programlama ve programlama editörleri incelenmiştir. Scratch ara yüzü, kod blokları ve ilk uygulamanın nasıl çalıştırılacağı gösterilmiştir. Doğrusal mantık yapısının problem çözümünde kullanılmasına değinilmiştir. Bu mantığın temel amacı sabit adımların oluşturduğu bir çözümü probleme uyarlayarak sonuca ulaşmaktır.

Kılavuz içerisinde problem çözümünde karar mantık yapısının nasıl kullanılacağı açıklanmıştır. Bu yapıda programcı bir şarta bağlı olarak program akışının değişmesini sağlayabilir. Bu sayede farklı durumlara göre esnek sonuçlar üretebilen programlar ortaya çıkar. Bir şarta bağlı olarak belirli sayıda yapılması istenen görevleri içeren döngülere de değinilmiştir.

Bu açıklamalardan sonra Scratch ile hazırlanmış uygulamalar incelenmiştir. Öğrencilerin kendi tasarlayabileceği oyunlar ve uygulamalar için ön çalışma niteliğinde olan çalışmalara yer verilmiştir.

İlerleyen bölümlerde elektronik devre kartı ile blok tabanlı kodlamanın nasıl yapılacağı açıklanmıştır. Scratch kodları ile blok tabanlı uygulamaların sanal ortamda oluşturulduktan sonra gerçekte nasıl etkileşim sağladığı incelenmiştir. Programlanabilir elektronik devre kartının pinleri, donanımı, nasıl programlanacağı açıklanmıştır. Ayrıca mBlock yazılımı ve robotlar menüsüne de detaylıca yer verilmiştir. Son bölümde ise elektronik devre kartı ile yapılabilecek uygulamalar için basit örnekler gösterilmiştir.

Bu kılavuzda genel olarak öğretmen ve öğrencilere kodlama konusunda yol gösterecek açıklama ve etkinliklere yer verilmiştir. Böylece kodlama konusunda öğretmenlere, öğrencilere rehberlik edecek bir doküman olması hedeflenmiştir.

KAYNAKÇA:

1-https://www.tynker.com/blog/articles/ideas-and-tips/how-to-explain-algorithms-to-kids/
2-Computing Without Computers- Curzon P.
3-https://www.tynker.com/blog/articles/ideas-and-tips/understanding-the-basic-algorithms-that-power-your-digital-life/
4-https://teachingexcellence.mit.edu/
5-https://dhgm.meb.gov.tr/yayimlar/dergiler/Milli_Egitim_Dergisi/147/altun.htm
6-http://yozgat.meb.gov.tr/www/5-6siniflar-kodlama-dersi-icerikleri/icerik/1368
7-https://www.thecodingdelight.com/sharpen-problem-solving-skills-programmer/#tab-con-16

5. ve 6. Sınıf
Bilişim Teknolojileri ve Yazılım Dersi

KODLAMA KILAVUZU

Yenilik ve Eğitim Teknolojileri Genel Müdürlüğü

5. ve 6. Sınıf
Bilişim Teknolojileri ve Yazılım Dersi

KODLAMA KILAVUZU