KARADENİZ TEKNİK ÜNİVERSİTESİ
EĞİTİM BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR VE ÖĞRETİM TEKNOLOJİLERİ EĞİTİMİ
ANABİLİM DALI

BLOK TABANLI KODLAMA ORTAMINDA PROBLEM ÇÖZME SÜREÇLERİNİN İNCELENMESİ

YÜKSEK LİSANS TEZİ

Suheda YILDIZ

TRABZON
Ocak, 2018

KARADENİZ TEKNİK ÜNİVERSİTESİ

EĞİTİM BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR VE ÖĞRETİM TEKNOLOJİLERİ EĞİTİMİ ANABİLİM DALI

BLOK TABANLI KODLAMA ORTAMINDA PROBLEM ÇÖZME SÜREÇLERİNİN İNCELENMESİ

Suheda YILDIZ

Karadeniz Teknik Üniversitesi Eğitim Bilimleri Enstitüsü'nce Yüksek Lisans Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Danışmanı
Doç. Dr. Ünal ÇAKIROĞLU

TRABZON
Ocak, 2018

KTÜ Eğitim Bilimleri Enstitüsü Müdürlüğü'ne

Bu çalışma jürimiz tarafından Bilgisayar ve Öğretim Teknolojileri Eğitimi Anabilim Dalı'nda YÜKSEK LİSANS tezi olarak kabul edilmiştir. 10 / 01 / 2018

Tez Danışmanı	**: Doç. Dr. Ünal ÇAKIROĞLU**	**.....**
Üye	**: Doç. Dr. Mehmet Barış HORZUM**	**.....**
Üye	**: Yrd. Doç. Dr. Zeynep TATLI**	**.....**

Onay
Yukarıda imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım.

Prof. Dr. Nevzat YİĞİT
Enstitü Müdürü V.

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Tezimin içerdiği yenilik ve sonuçları başka bir yerden almadığımı; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalardan bilimsel etik ilke ve kurallara uygun davrandığımı, tez yazım kurallarına uygun olarak hazırlanan bu çalışmada kullanılan her türlü kaynağa eksiksiz atıf yaptığımı ve bu kaynaklara kaynakçada yer verdiğimi, ayrıca bu çalışmanın Karadeniz Teknik Üniversitesi tarafından kullanılan "bilimsel intihal tespit programı"yla tarandığını ve hiçbir şekilde "intihal içermediğini" beyan ederim. Herhangi bir zamanda aksinin ortaya çıkması durumunda her türlü yasal sonuca razı olduğumu bildiririm.

Suheda YILDIZ

10 / 01 / 2018

ÖN SÖZ

Blok tabanlı kodlama ortamında problem çözme süreçlerinin incelenmesi konusundaki bu çalışma, Karadeniz Teknik Üniversitesi Eğitim Bilimleri Enstitüsü Bilgisayar ve Öğretim Teknolojileri Eğitimi Anabilim Dalında Yüksek Lisans Tezi olarak hazırlanmıştır.

Bu çalışma süresince danışmanlığımı üstlenerek, hiçbir zaman yardımını esirgemeyen her durumda ve her ortamda bana destek veren, gerek konunun belirlenmesinde gerekse çalışmanın yürütülmesi sırasında engin bilgi ve deneyimlerinden sürekli yararlandığım değerli hocam, Doç. Dr. Ünal ÇAKIROĞLU'na sonsuz teşekkürlerimi sunarım.

Ayrıca çalışmam sırasında benden yardımlarını esirgemeyen manevi kardeşim Canan DOST'a, tüm İngilizce çevirilerde fazlasıyla destek olan Kübra KOLAYLI BİRİNCİ'ye, sevgili arkadaşlarım Fidan BUDAK'a ve Berkay MUMCU'ya sonsuz teşekkürlerimi sunarım.

Çalışmanın yürütüldüğü ortaokulda görev yapan ve çalışma süresince yardımlarını esirgemeyen başta kıymetli meslektaşım Ufuk GÜNAYDIN olmak üzere tüm öğretmenlere, öğrencilere ve anlayışlı okul müdürüm Ergün TAŞ'a teşekkür eder, saygılarımı sunarım.

Ayrıca tüm eğitim hayatım boyunca yanımda olan en büyük destekçim ve hakkını asla ödeyemeyeceğim değerli halam, Hatice ÇOLAK'a, eşi Ömer ÇOLAK'a, yüksek lisans eğitimime devam etmem noktasında her zaman ısrarla beni motive eden dedem Ziya YILDIZ ve babaannem Hayriye YILDIZ'a, en kıymetlim; kardeşim Umut YILDIZ'a, son olarak anne ve babama teşekkürlerimi sunarım.

Suheda YILDIZ

Trabzon 2018

İÇİNDEKİLER

ÖN SÖZ ..... **IV**

İÇİNDEKİLER ..... **V**

ÖZET ..... **VII**

ABSTRACT ..... **VIII**

TABLOLAR LİSTESİ ..... **IX**

ŞEKİLLER LİSTESİ ..... **X**

KISALTMALAR LİSTESİ ..... **XII**

1. GİRİŞ **1**

1.1. Araştırmanın Amacı 2

1.2. Araştırmanın Gerekçesi ve Önemi 2

1.3. Araştırmanın Sınırlılıkları 5

1.4. Araştırmanın Varsayımları 6

1.5. Tanımlar 6

2. LİTERATÜR TARAMASI **7**

2.1. Araştırmanın Kuramsal Çerçevesi 7

2.1.1. Bilgi – İşlemsel Düşünme 8

2.1.2. Bilgi – İşlemsel Düşünme ve Problem Çözme Becerisi 11

2.1.3. Programlama ve Problem Çözme 14

2.1.4. Çocuklarda Bilgi – İşlemsel Düşünme Becerisinin Geliştirilmesi ve Programlama Öğretimi 17

2.1.4.1. Blok Tabanlı Kodlama Ortamları 19

2.1.4.2. Neden Code.org? 20

2.1.4.3. Blok Tabanlı Kodlama Ortamlarında Problem Çözme Süreçleri 22

2.1.5. İlgili Çalışmalar 24

2.1.5.1. BTKO İle Farklı Becerilerin Geliştirilmesine Yönelik Çalışmalar 24

2.1.5.2. BTKO Olarak Code.org Kullanan Çalışmalar 26

2.2. Literatür Taramasının Sonucu 26

3. YÖNTEM **28**

3.1. Araştırma Modeli 28

3.2. Araştırma Grubu / Evren ve Örneklem / Denek - Denekler 29

3.3. Verilerin Toplanması 29

3.3.1. Veri Toplama Araçları/Teknikleri 30
3.3.1.1. Ekran Video Kaydı 30
3.3.1.2. Gözlemler 31
3.3.1.3. Yarı Yapılandırılmış Mülakat 32
3.3.2. Veri Toplama Süreci / Uygulama Akışı 32
3.3.3. Araştırmanın Geçerliliği ve Güvenilirliği 33
3.4. Verilerin Analizi 34

4. BULGULAR 37
4.1. Temel Taşlar (Tüm Sahneler) 37
4.2. Döngüler (Sahne 1-2-6-7-8-9) 42
4.3. Koşul Yapıları (Sahne 5-6-8) 47
4.4. Fonksiyonlar (Sahne 3-4) 52
4.5. Hata Ayıklama (Sahne 3-4) 61
4.6. Code.Org BTKO'da Ortaya Çıkan Problem Çözme Davranışlarına Bütüncül Bakış 64

5. TARTIŞMA 69
5.1. Code.org'da Problem Çözme Süreci 69
5.2. Code.org Özelliklerinin Problem Çözmedeki Rolü 74
5.3. Araştırmayı Farklılaştıran Bazı Durumlar ve Sınırlılıklar 76

6. SONUÇ VE ÖNERİLER 78
6.1. Sonuçlar 78
6.2. Öneriler 79
6.2.1. Araştırma Sonuçlarına Dayalı Öneriler 79
6.2.2. İleride Yapılabilecek Araştırmalara Yönelik Öneriler 79

7. KAYNAKLAR 81

8. EKLER 88

9. ÖZGEÇMİŞ VE İLETİŞİM BİLGİLERİ 93

ÖZET

Blok Tabanlı Kodlama Ortamında Problem Çözme Süreçlerinin İncelenmesi

Ortaokul bilişim teknolojileri ve yazılım dersi öğretim programında önemli bir yere sahip olan kodlama eğitimi, bilgi-işlemsel düşünme becerileri çerçevesinde problem çözme becerilerinin geliştirilmesine yönelik kazanımları içermektedir. Bu çerçevede kullanılması önerilen blok tabanlı kodlama ortamları, sağladıkları birçok araçla problem çözme sürecini kolaylaştırmaktadır. Araştırmalarda BTKO üzerinde çalışan öğrencilerin süreç sonundaki bilgi-işlemsel düşünmelerine ilişkin değerlendirmelere yer verilse de öğrencilerin süreçte problem çözme süreçlerinin nasıl geliştiğine, öğrencilerin nasıl davrandıklarına yönelik tanımlamalara ihtiyaç söz konusudur. Bu düşünceden hareketle bu çalışmada BTKO ile problem çözme etkinliklerinde öğrencilerin problem çözme süreçlerinde gösterdikleri davranışlar belirlenerek, BTKO'da problem çözme ile ilgili sürecin tanımlanmasına ilişkin bir çerçeve oluşturulmaya çalışılmaktadır. Çalışma 15 altıncı sınıf öğrencisiyle bir dönem boyunca devam eden Bilişim Teknolojileri ve Yazılım Dersinde Code.org ortamındaki problem çözme etkinlikleri üzerinden gerçekleştirilmiştir. Çalışmada problem çözme süreçleri betimlenerek elde edilen veriler ile sürecin temel aşamaları ortaya çıkarılmaya çalışılmaktadır. Nitel verilerin yorumlanmasıyla sonuca ulaşılan bu çalışmada bulgular ekran kayıtları mülakatlar ve gözlemlerden elde edilmiştir. Elde edilen bulgular nitel veri analizi programı Nvivo11 ile analiz edilmiştir. Sonuç olarak, Code.org üzerinde problem çözme süreçlerinin temel olarak Odaklanma, Sınama ve Sonuçlandırma şeklinde 3 aşamada gerçekleştiği belirlenmiştir. Bu aşamalar farklı faktörlerin etkisiyle bazı durumlarda birbirini takip eden sırada doğrusal biçimde gerçekleşirken, bazı durumlarda ise aşamalar arası döngüsel geçişler söz konusu olabilmektedir. Elde edilen sonuçların blok tabanlı kodlama ortamlarında yapılacak olan uygulamalarda, ders tasarımcılarına derslerin planlanması ve yürütülmesin noktasında rehber olacağı düşünülmektedir.

Anahtar Kelimeler: Blok tabanlı kodlama ortamları, Code.org, Bilgi-işlemsel düşünme, Problem çözme süreçleri

ABSTRACT

Investigation of Problem Solving Processes in Block Based Coding Environment

Considering computational thinking skills; coding instruction, which is of great importance in the middle school information technology and software lesson curriculum, consist of the functions that are designed for developing problem solving skills. The block-based coding environments suggested to be used to facilitate the process of problem-solving with the instruments that they provide. Even though there are evaluations related to computational skills of students working on this kind of environments in other studies, there is a need for definitions about how problem solving process of the students develops, how the students behave in the process. Based on this notion, in this study a framework is generated to determine the process of problem solving by defining behaviors of students displayed on BTKO problem solving activities. The study was performed with 15 6th grade students on Code.org problem solving environment activities during one term long ITC and Research Methods and Techniques lessons. In the study it was tried to reveal the main stages of the process with the data acquired from the description of problem solving processes. The results suggested that, it was stated that the problem solving processes are basically recognized as focusing, evaluating and finalisation. Some implementations andsuggestions for successful applications are also included.

Key Words: Blok Based Coding Platform, Code.org, Computational thinking, Problem solving process

TABLOLAR LİSTESİ

Tablo No	Tablo Adı	Sayfa No
1.	Wing(2008) ve NRC (2008)'e Göre Bilgi – İşlemsel Düşünme Alt Becerileri	12
2.	Farklı Araştırmalarda Bilgi – İşlemsel Düşünmeye İlişkin Alt Becerilerin Karşılaştırılması (Bacconi ve diğ., 2016)	13
3.	Çetin (2012)'e Göre Problem Çözme ve Programlama Süreci	16
4.	Geleneksel Programlama Ortamlarındaki Problem Çözme Süreçleri	22
5.	Araştırmanın Şekillendirilmesine Literatürün Katkısı	27
6.	Temel Taşlara İlişkin Kodlar	38
7.	Döngü Yapılarına İlişkin Kodlar	42
8.	Koşul Yapılarına İlişkin Kodlar	47
9.	Fonksiyon Yapılarına İlişkin Kodlar	53
10.	Hata Ayıklama Yapılarına İlişkin Kodlar	61
11.	Tüm Yapılara Yönelik Davranışları Yansıtan Kodlar	65

ŞEKİLLER LİSTESİ

Şekil No	Şekil Adı	Sayfa No
1.	ISTE 21.yy bireylerden beklenen roller	7
2.	Bilgi – İşlemsel düşünme modeli (Sengupta ve diğ., 2013)	9
3.	Bilgi – İşlemsel düşünme kavramlar ve yaklaşımlar (Berry, 2015)	10
4.	Problem çözme ve programlama arasındaki ilişkiye yönelik kavramsal çerçeve (Nance, 2016)	17
5.	Çalışmanın teorik çerçevesi	24
6.	Araştırma sürecinin gösterimi	29
7.	Veri toplama araçları	30
8.	Ekran videosu kayıt örnek görüntüsü	31
9.	Code.org öğretmen ara yüzü	33
10.	Farklı kaynaklardan gelen verilerin analiz süreci	36
11.	Sahnelerde blokların bulunduğu alan	37
12.	Kadar tekrarla-yap bloğunun görüntüsü	39
13.	İpucu alanı	40
14.	Sahne8 Ö4'e ait ekran görüntüsü	45
15.	Sahne8 Ö11'e ait ekran görüntüsü	46
16.	Sahne6 Ö3'e ait ekran görüntüsü	50
17.	Sahne5 Ö4'e ait ekran görüntüsü	50
18.	Sahne3, Ö10'a ait ekran görüntüsü	54
19.	Sahne3, Ö11'e ait rastgele blokları karıştıran ekran görüntüleri	56
20.	Sahne4, Ö13'e ait ekran görüntüsü	57
21.	Sahne7, Ö6'ya ait ekran görüntüsü	59
22.	Sahne5, Ö12'nin izlediği işlem adımları örneği	60
23.	Sahne3, Ö3'e ait ekran görüntüsü	63

24. Sahne2, Ö5'e ait ekran görüntüsü 64

25. Code.org BTKO'da ortaya çıkan problem çözme süreçleri temel aşamaları 72

KISALTMALAR LİSTESİ

BTKO : Blok Tabanlı Kodlama Ortamları

MEB : Milli Eğitim Bakanlığı

ISTE : International Society for Technology in Educational [Eğitimde Teknoloji Uluslararası Topluluğu]

1. GİRİŞ

Gelişen ve değişen dünyada, ihtiyaçlar da değişmiş ve bu ihtiyaçları karşılamak için geçmişte daha çok insanlardan beklentiler kas gücü şeklinde iken, günümüzde bu daha çok düşünceye dayalı becerilere dönüşmüştür. Bu becerilerden birisi olan bilgi-işlemsel düşünme günümüzde bireylerin sahip olması beklenen en önemli beceriler arasında değerlendirilmektedir (Wing, 2008). Bu becerinin geliştirilmesi yönünde eğitim politikalarında da çeşitli düzenlemeler yapılmakta ve güncellenen öğretim programlarına bilgi-işlemsel düşünme ile ilişkisi bilinen programlama öğretimi dahil edilmektedir (Milli Eğitim Bakanlığı [MEB], 2017). Türkiye'de başlangıçta liselere getirilen kodlama ile ilgili düzenleme 2017 yılında yayınlanan güncel öğretim programı ile Bilişim Teknolojileri ve Yazılım dersi kapsamında ortaokul öğretim programında da önemli bir yer kaplamıştır. Hazırlanan güncel öğretim programlarında küçük yaşlara yönelik ders içinde programlama öğretimi için kullanılabilecek farklı ortamlar önerilmektedir. Örneğin, Scratch, Alice, Code.org ortamları bunların başlıcaları arasında sayılabilir.

Bu ortamlardaki eğitimler çoğunlukla öğrencilere sunulan problemlerin veya kendi oluşturdukları problemlerin programlamanın temel yapıları kullanılarak çözülmesi şeklinde gerçekleşir. Bu süreçte öğrenciler bir taraftan programlamanın temel yapılarını öğrenirken, diğer taraftan problem çözme becerileri kazanırlar. Bu ortamların önemli bir yönü de programlama sürecinde yapılması gerekenleri kolaylaştırmalarıdır. Nitekim bu ortamlar, programlama sürecinde genellikle gerekli olan sözdizimsel bilgi, mantıksal bilgi ve kavramsal bilgi gibi bilgilerin tümüne hâkim olmadan da problem çözme sürecini gerçekleştirebilmelerini sağlayabilmektedir. Bu ortamlarda görselleştirme unsurları kullanılarak, lego mantığı ile öğrencilerin herhangi bir dile ait sözdizimsel yapısını bilme zorunlulukları ortadan kaldırılır. Diğer yandan bu eğitimlerin birincil amacı tümüyle programlama öğretmek olmadığından, programlamaya ilişkin kavramsal bilgilerin tamamını bilmelerine de gerek kalmaz. Dolayısıyla bu ortamlar daha çok programlamanın mantıksal bilgi gerektirmesi çerçevesinde bu bilgi üzerine daha çok yoğunlaşır. Böylelikle problem çözme süreci çoğu zaman kısalır. Programlamaya ilişkin gerekli tüm bilgiler kullanılmadığından ve belki de gerçek anlamda programlama yapılmadığından bu eğitimler programlama eğitiminden çok kodlama eğitimi olarak anılmaktadır. Bununla birlikte daha çok ekrandaki blok şeklindeki nesnelerin programlama yapılarını temsil ettiği fikrinden hareketle bu ortamlar Blok Tabanlı Kodlama Ortamları (BTKO) şeklinde isimlendirilmektedir.

BTKO'lar, kendilerine özgü ara yüzler ve programlama yapılarıyla ilişkilendirilme durumlarına göre problem çözme sürecini farklı şekillerde ele alabilmektedirler. Bu ortamlarda verilen eğitimlere ilişkin değerlendirme çalışmaları son yıllarda artış göstermiş olsa da, sürecin betimlenmesi ile ilgili bulgulara hala ihtiyaç söz konusudur. Nitekim bu ortamlarda öğrencilerin problem çözme süreçleri ortamın sunduğu imkânlar ve çocukların bunları algılama şekilleri çerçevesinde gelişecektir. Bu çerçevede bu ortamlarda çözümlemek üzere hazırlanan etkinliklerin öğretim programına giriyor olması; öğrencilerin bu ortamlarda nasıl problem çözdüklerinin, nasıl düşündüklerinin, hangi davranışları sergilediklerinin belirlenmesini ilgili öğretmenler ve ders tasarımcıları için önemli kılmaktadır. Bu düşünceden hareketle, bu çalışmada BTKO olarak Code.org seçilmiş ve süreç derinlemesine incelenmeye çalışılmıştır.

1. 1. Araştırmanın Amacı

Bu çalışma ortaokul öğrencilerinde BTKO'lar kullanılarak verilen kodlama eğitimi ile öğrencilerin problem çözme süreçlerinin ne şekilde gerçekleştiğini belirlemeyi amaçlamaktadır.

Bu amaç doğrultusunda araştırmanın problemi: "Blok tabanlı kodlama ortamlarında ortaokul öğrencileri problemleri çözerken nasıl bir süreç izlemektedirler?" şeklinde ifade edilebilir.

Araştırma problem çözme sürecini tanımlayan aşamaların belirlenmesi ve bu aşamalardaki davranışların ortaya konulması şeklinde ele alınacaktır. Bu doğrultuda araştırmanın alt problemleri;

1. Blok tabanlı kodlama ortamları ile gerçekleştirilen programlama eğitiminde problem çözme becerileri temelde hangi aşamaları içermektedir?

2. Belirlenen aşamalar doğrultusunda öğrenciler ne gibi davranışlar sergilemektedirler? şeklinde ifade edilebilir.

1. 2. Araştırmanın Gerekçesi ve Önemi

21.yüzyılda toplumsal hayatta değişen ihtiyaçlara cevap verebilmek için ülkeler nitelikli insan yetiştirme arayışlarına girmişlerdir. Bu çerçevede eğitim sisteminde bilginin üretilmesi kullanılması, aktarılması ile ilgili yeni becerilerin kazandırılması yönünde değişimler yaşanmaktadır. Bu beceriler arasında yaratıcı düşünme, bilgi-işlemsel düşünme, problem çözme, iletişim ve işbirlikli çalışma, eleştirel düşünme gibi beceriler öne çıkmaktadır (Ananiadou ve Claro, 2009; Eğitimi Araştırma ve Geliştirme Dairesi Başkanlığı [EARGED], 2011; Giordano ve Maiorana, 2015; Pinto ve Escudeiro, 2014;

Trilling ve Fadel, 2009). Özellikle bilgi-işlemsel düşünme günümüz bilişim çağı toplumlarının sahip olması gereken en önemli becerilerden birisi olarak gösterilmektedir (Kert ve Uğraş, 2009). Daha çok bilgisayar bilimleri ile ilişkili olan bu becerinin geliştirilmesi çerçevesinde yapılan çalışmalar, son 10 yıl sürecinde bilgisayar bilimlerine karşı toplumsal bakışı da etkilemiştir.

Bu noktada Zhou (2007), bilgi-işlemsel düşünmeyi "Bilgisayar biliminin temel kavramlarını kullanarak problemleri çözmeyi, sistemleri tasarlamayı ve insan davranışlarını kavrayabilmeyi içeren bilgisayar bilimini kapsayan bir dizi düşünme eylemi" olarak ele almaktadır. Zhou (2007) bu beceriyi, daha çok bilgisayar bilimi çerçevesinde ele alsa da bilgi-işlemsel düşünme sadece bilgisayar alanında çalışanların sahip olması gereken bir beceri olmayıp temel okuma yazma ve hesaplama süreçleri gibi herkesin sahip olması gereken bir beceri şeklinde değerlendirmektedir. Benzer biçimde ISTE (2016) de bu beceriyi bilgisayar bilimi çerçevesindeki bilgilerin kazanılması ve kullanılmasından çok, bilgisayar bilimlerindeki düşünme yapılarının diğer derslere veya bağlamlara da aktarılabilmesi olarak değerlendirmektedir.

Bilgi işlemsel düşünme becerilerinin gelişiminde programlamanın önemli bir yeri söz konusudur (Güneş ve Karabak, 2013; Resnick, Maloney, Monroy-Hernández ve Rusk, 2009; Shin, Park ve Bae, 2013). Bu bağlamda, programlama sürecinde öğrenciler algoritmik düşünme, mantıksal muhakeme, bilgisayar bilimi temel unsurlarıyla ilişkiler kurma, soyutlama gibi birçok alt beceriler ile birlikte problem çözme becerileri geliştirirler. Bu nedenle programlama özellikle küçük yaştaki öğrencilerin problem çözme becerilerinin gelişimi için önerilmektedir (Brown ve diğ., 2013; Du, Wimmer ve Rada, 2016). Bu noktadan hareketle, problem çözme becerilerinin geliştirilmesi amacıyla birçok ülkede küçük yaşlardan başlayarak programlama eğitimleri verilmeye başlanmıştır. Bu eğitimlerde özellikle küçük yaştaki öğrenciler için başlangıç seviyesinde blok kodlama ortamları kullanılmaktadır (Grover ve Pea, 2013).

Bu durum dikkate alınarak Türkiye'de de Güzel Sanatlar ve Spor Liselerinde 2016 - 2017 (Talim Terbiye Kurulu Başkanlığı [TTKB], 2017), Hazırlık Sınıfı Bulunan Özel Anadolu Lisesi, Özel Anadolu Lisesi, Hazırlık Sınıfı Bulunan Özel Fen Lisesi ve Özel Fen Lisesi ise 2017-2018 eğitim öğretim yılından itibaren uygulanacak şekilde içeriğinde programlamanın önemli bir yeri olan bilişim dersi öğretim programlarında güncellenerek ilgili düzenlemeler yapılmıştır (TTKB, 2017). Bununla birlikte 2017'de güncellenen ortaokul seviyesindeki Bilişim Teknolojileri ve Yazılım dersi öğretim programında da bilgi-işlemsel düşünme becerileri çerçevesinde problem çözmeye özellikle önem verilmektedir. Bu durum öğretim programının genel amaçlarının sıralandığı bölümde, *"Bu program; …5. Problem çözme ve bilgi-işlemsel düşünme becerileri edinmelerini ve geliştirmelerini, … 10.*

Problemleri çözmek için uygun programlama yaklaşımını seçerek uygulayabilmelerini… amaçlamaktadır." şeklinde yer almaktadır. Bu şekilde öğrencileri sorumluluk sahibi ve eleştirel düşünebilen bireyler haline getirebilmek için problem çözme, karar verme becerileri geliştirilmesi öngörülen temel beceriler arasında sayılarak özellikle problem çözmeye vurgu yapılmaktadır (MEB, 2017). Ünite temelli yaklaşımı benimseyen güncel öğretim programında bir ünite bilgi-işlemsel düşünme becerisine ayrılmış ve bu ünitenin amaçları aşağıdaki şekilde ifade edilmiştir:

> "Problem Çözme ve Programlama ünite başlığı altında; algoritma tasarımına ilişkin (arama, sıralama vb.) anlayış geliştirme; sözel ve görsel olarak ifade etme, problem çözmek için değişken, atama, sıralı mantık, karar yapısı, döngü ve fonksiyon yapılarını kullanma, problemleri çözmek için uygun programlama yaklaşımını seçme ve uygulama konusunda beceriler kazandırılması amaçlanmıştır. 5 ve 6. sınıflarda haftada iki saat zorunlu olarak okutulacak derste öğrencilerin temel bilgisayar kullanımı ve programlama becerileri kazanmaları hedeflenmiştir." (MEB, 2017).

Programlamanın öğretim programında yer alması, bu alanda yapılacak olan öğretim tasarımları ve uygulamalarına yönelik çalışmalara da hız kazandırmıştır. Bu çerçevede farklı ortamlar kullanılmakta ve farklı yöntemler uygulanabilmektedir. Bu tasarım ve uygulamaların olumlu sonuçlar oluşturmasında öğrencilerin programlama sırasında problemleri nasıl çözdüklerine yönelik tanımlamaların yapılması önemlidir. Bu çerçevede geleneksel programlama dilleri editörleri ile çalışan öğrencilere ilişkin gerek problem çözme gerekse bilgi-işlemsel düşünmenin diğer alt becerilerine yönelik öğrencilerin geçirdikleri süreçleri belirleme, aşamalandırma ve bu sürece ilişkin tanımlamaları ele alan birçok çalışma mevcuttur (Fessakis, Gouli ve Mavroudi, 2013; Jonassen, 2000; Palumbo, 1990; Winslow, 1996). Ancak özellikle son 10 yılda giderek yaygınlaşmaya başlayan BTKO üzerinde kodlama eğitimleriyle problem çözümlerinde, süreç tipik programlama yapısından farklılaşmaktadır. Bu noktada BTKO üzerindeki problem çözme süreçlerini tanımlayan veya aşamalandıran çerçeveler bu alandaki araştırmalar için katkı sağlayıcı olabilir. Her ne kadar bazı çalışmalarda (Çetin, 2012; Kalelioğlu ve Gülbahar, 2014; Olgun, 2014) BTKO üzerinde çalışan öğrencilerin süreç sonundaki bilgi-işlemsel düşünmelerine ilişkin değerlendirmelere yer verilse de öğrencilerin süreçte problem çözme süreçlerinin nasıl geliştiğine, öğrencilerin nasıl davrandıklarına yönelik tanımlamalara ihtiyaç söz konusudur. Nitekim bu tanımlamalar içerisinde BTKO özelliklerinin öğrencilerin problem çözme süreçlerine yansımaları da ele alınabilecektir. Dolayısıyla bu çalışma ile BTKO ile problem çözme etkinliklerinde öğrencilerin problem çözme süreçlerinde gösterdikleri davranışlar belirlenerek, BTKO'da problem çözme ile ilgili sürecin tanımlanmasına ilişkin bir çerçeve oluşturulmaya çalışılmaktadır.

Diğer yandan, Türkiye'de yenilenen Bilişim Teknolojileri ve Yazılım dersi programı ile ortaokul öğrencilerine yönelik gerçekleştirilecek olan BTKO temelli kodlama eğitiminde

öğrencilerin süreçte problem çözme becerilerindeki gelişimi belirleyecek ölçme araçlarına ihtiyaç söz konusu olacaktır. Bu çalışmada geliştirilecek olan öğrencilerin problem çözme yollarına ilişkin sürece yönelik tanımlamaların ve aşamalandırmanın, BTKO süreçlerinde gelişen problem çözme becerilerini ölçebilecek ölçme araçları için araştırmacılara yol gösterici olabileceği düşünülmektedir.

Çalışmada verilecek olan BTKO temelli kodlama eğitimleri mevcut örnekler üzerinden ilerlediğinden öğretim sürecinde geliştirilen problem çözme için düşünme yollarına ilişkin öğrenci davranışları kayıt altına alınıp değerlendirilmesi mümkün olabilmektedir. Ayrıca çalışmada verilecek olan eğitim için oluşturulan ders tasarımının farklı programlama yapıları için problem çözümüne yönelik hazırlanmıştır. Bu noktada çalışmada ortaya çıkan ders tasarımının ve süreçte kullanılan öğretim yöntemlerinin işe yararlığı hakkında öğretmenlere ve BTKO için ders tasarımcılarına fikir vereceği düşünülmektedir.

Günümüzde popüler olarak Scratch, Code.org, Alice, SmallBasic, KoduGameLab gibi BTKO ortamları kodlama eğitimleri için sıklıkla kullanılmaktadır. Code.org'un, BTKO içerisinde önemli bir yeri olsa da halen Code.org üzerinden yürütülen öğretim programlarının problem çözmedeki etkileri ile ilgili belirgin sonuçlara ulaşılmış değildir. Bu durumun nedenlerinden birisi olarak öğrencilerin problem çözme sürecindeki davranışlarını modelleyen tanımlamaları esas alan, kesin sonuçlar üreten değerlendirme araçlarının olmayışı düşünülebilir. Bu çalışma Code.org ortamında öğrencilerin problem çözme becerilerinin ne şekilde geliştiğini, sürecin nasıl ilerlediğini ortaya koyarak; hazırlanacak değerlendirme araçları için de katkı sağlayıcı olacaktır. Diğer yandan Johnson, Adams Becker, Estrada ve Freeman (2014) verilerin mantıksal analizi, modelleme ve soyutlama, problem çözme gibi becerilerin geliştirilmesiyle öğrencilerin dünyalarının nasıl çalıştığını anlamalarını ve karmaşık sorunları çözmede gerekli olan becerilerle donatılabileceğini ifade etmektedir. Dolayısıyla öğrencilerin programlama sürecinde problem çözme gelişimleri sürecine yönelik belirlenecek tanımlamalar aslında kendi hayatlarındaki diğer karmaşık problemleri de nasıl çözebilecekleri, bu çözümlerde temelde ilerleyememe nedenleri ve bunlara bulunabilecek çözüm önerileri noktasında da öğretim tasarımcılarına yol gösterebilir.

1. 3. Araştırmanın Sınırlılıkları

1. Araştırmada Code.org ortamı sahnelerindeki etkinlikler ele alınmış olup, bulgular bu ortama özgü özellikler çerçevesinde de ele alınarak yorumlanmıştır. Bu

noktada çalışma bulguları BTKO için genellenmemiş olsa da, sürecin takibi ve ele alınış şeklinin diğerleri için de yol gösterici olabileceği düşünülmektedir.

2. Araştırmada BTKO'da gerçekleşen problem çözme süreçleri sınırlı sayıdaki katılımcının deneyim ve değerlendirmeleri çerçevesinde ele alınmıştır.

1. 4. Araştırmanın Varsayımları

1. Katılımcıların kendilerine verilen zaman dilimleri içerisinde BTKO üzerindeki etkinlikleri yapmaları istenmiş olup bu durumun dışında kullanmadıkları varsayılmaktadır.
2. Öğretim süreci ile ilgili yapılan mülakatlarda katılımcıların yansız ve samimi cevaplar verdikleri varsayılmaktadır.

1. 5. Tanımlar

BTKO: Blok Tabanlı Kodlama Ortamları

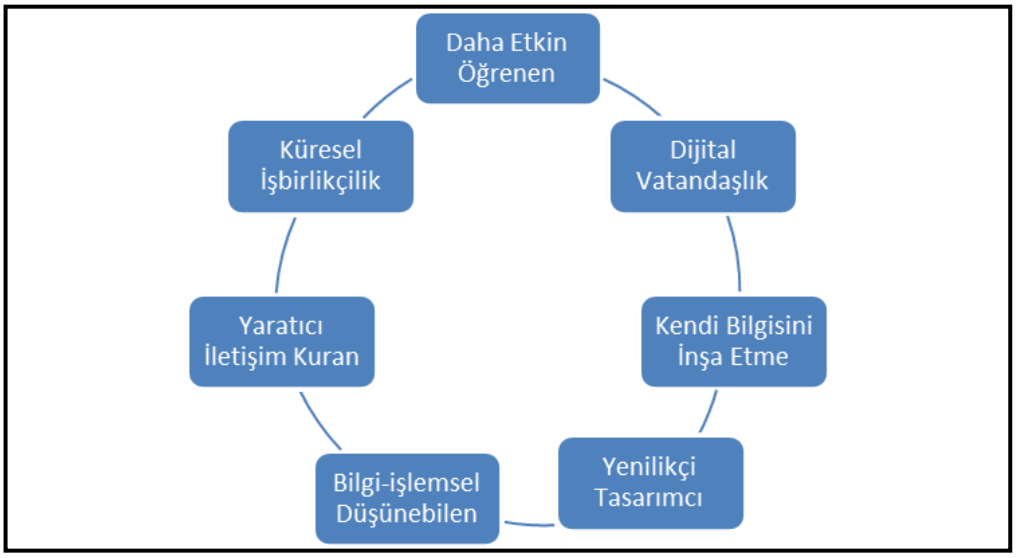
Code.org: Kodlama Ortamları için Kullanılan Ücretsiz Online Platform

HyperCam: Ekran Video Kaydı Alan Yazılım

2. LİTERATÜR TARAMASI

2. 1. Araştırmanın Kuramsal Çerçevesi

Değişen dünya ile birlikte 21.yy öğrenenlerinden beklenen roller de her geçen gün güncellenmektedir (Trilling ve Fadel, 2009). ISTE (2016) bu rolleri; daha etkin öğrenen rolü, dijital vatandaşlık, kendi bilgisini inşa edebilme, yenilikçi tasarımcı, Bilgi-işlemsel düşünebilme, yaratıcı iletişim kurabilme, küresel işbirlikçilik şeklinde ifade etmektedir. Bu roller Şekil 1'de özetlenmektedir.



Şekil 1. ISTE 21.yy bireylerden beklenen roller

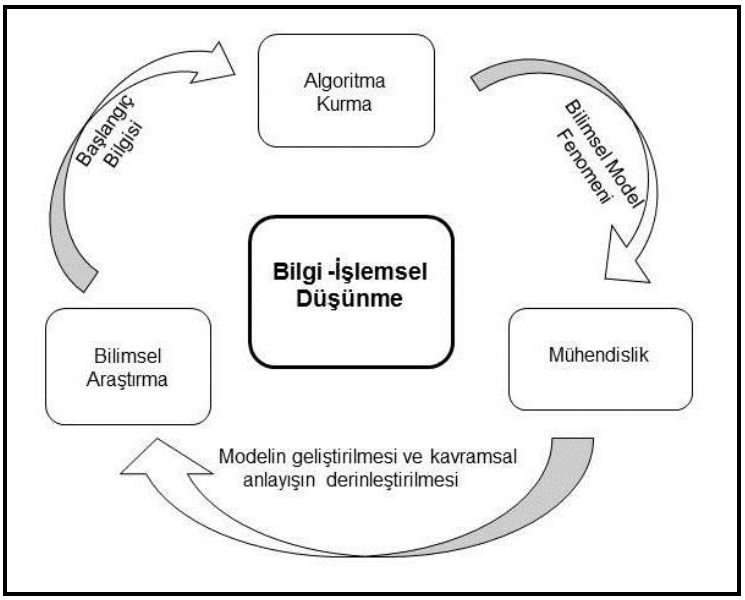
Bu roller içerisinde bilgi-işlemsel düşünme becerisine sahip olma ve bu beceriyi kullanabilme önemli bir yer tutmaktadır. Bilgi-işlemsel düşünme ile ilişkili alt beceriler; yaratıcı düşünme, algoritmik düşünme, eleştirel düşünme, problem çözme, işbirlilik ve iletişim becerileri olarak düşünülmektedir. Bireylerin okul çağında bu becerilere sahip olmaları onların hem kendilerini geliştirmeleri hem de dijital çağ öğrenme kültürüne sahip olmaları açısından önemli görülmektedir (Brown, 2015; Grover ve Pea, 2013; Günüç, Odabaşı ve Kuzu, 2013; ISTE, 2016; Lye ve Koh, 2014). Bu çerçevede aşağıdaki bölümde, bilgi-işlemsel düşünme becerisi ve problem çözme becerisi ile ilişkileri tartışılmaktadır.

2. 1. 1. Bilgi – İşlemsel Düşünme

Bilgi-işlemsel düşünme becerisi ifadesini ilk kullanan araştırmacılardan birisi olan Wing (2006) bu beceriyi; sadece bilgisayar bilimcileri için değil okuma, yazma ve aritmetik gibi her çocuğun analitik kabiliyetine eklenmesi gereken temel bir beceri şeklinde tanımlamaktadır. Bu tanımlamanın içerisinde karmaşık sistemleri tasarlarken veya karmaşık görevleri yerine getirirken bilgi-işlemsel düşünmede soyutlama ve ayrışma gibi işlemler öne çıkmaktadır. Bilgi-işlemsel düşünme becerisine yönelik çalışmalar arttıkça araştırmacılar bu becerinin gelişebileceği ortam ve bağlamlar için farklı bulgulara dayalı öneriler ortaya koymuşlar, bu durum yaptıkları tanımlamalarına da yansımıştır. Bu çerçevede bilgi-işlemsel düşünmeye ilişkin tanımlamalardan bazıları aşağıda ifade edilmektedir.

Bundy (2007) bilgi-işlemsel düşünmeyi oldukça geniş bir çerçevede değerlendirmiş; beşeri bilimlerlerden doğal bilimlere neredeyse tüm disiplinlerdeki araştırmalarda yer alabilecek bir beceri olarak ifade etmiştir. Curzon (2015) ise bilgi-işlemsel düşünmeyi problem çözme olarak değerlendirmiştir. Bu süreçte belli bir bakış açısına göre bir problemi çözerken çözümleri düşünmeden önce problemin ne olduğunu anlamak gerekmektedir. Bir diğer çalışmada Patan (2016) bilgi-işlemsel düşünmeyi teknolojinin kullanımı ile çözülebilecek bir problemi tanımlayarak çözüm yollarını algoritmik düşünme ile belirleyip, en etkili ve verimli çözümü uygulayarak değerlendirmeyi, benzer durumlara bu süreci transfer etmeyi içeren bir beceri kümesi olarak ele alır.

Daha geniş bir perspektifle Sengupta, Kinnebrew, Basu, Biswas ve Clark (2013) bilgi-işlemsel düşünme becerisini öğrenme ortamlarında çok disiplinli olarak ele almaktadır. Bu çerçevede özellikle fen ve mühendislik konuları ile ilgili becerileri bazı tanımlamalar öne çıkarmaktadır. Bu tanımlama Şekil 2'de özetlenmektedir.

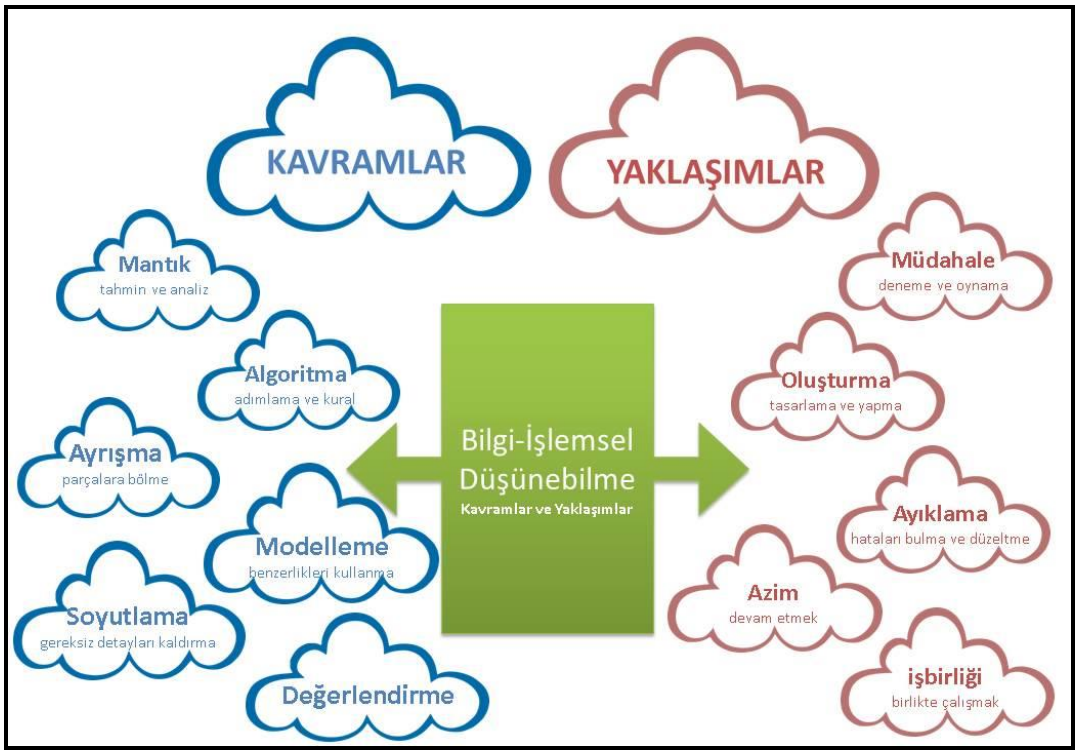


Şekil 2. Bilgi – İşlemsel düşünme modeli (Sengupta ve diğ., 2013)

Wing (2006) yukarıdaki tanımlamaların ortak yönlerinden hareketle bilgi-işlemsel düşünme becerisi için aşağıda sıralanan bazı temel karakteristikleri ortaya koymaktadır. Araştırmacı bilgi-işlemsel düşünme becerisinin birçok beceriyle ilişkisi olduğundan hareketle bu becerinin ne olduğunun yanında ne olmadığına ilişkin de değerlendirmeler yapmaktadır. Bu çerçevede bilgi-işlemsel düşünme için:

- Kavramsallaştırmalar içerdiği ancak tümüyle programlama becerisi olmadığı;
- Temel bir beceri olduğu ancak alışılmış bir beceri olmadığı;
- İnsanlar için bir düşünme şekli olduğu ancak bilgisayarların düşünme şekli olmadığı;
- Daha çok fikirler ve düşünme yapılarını içerdiği, insan eliyle yapılan ürünler olmadığı;
- Matematiksel ve mühendislik düşünme biçimini bütünleştirilme ve birleştirme olduğu
- Herkes için, her yerde gerekli olan bir beceri olduğunu ortaya koymaktadır.

Literatür incelendiğinde bilgi-işlemsel düşünmenin farklı kavramları içerdiği görülmektedir. Bu kavramlar bazı durumlarda farklı becerilere karşılık gelirken, bazı durumlarda problem çözme becerilerini işaret etmektedir. Bu çerçevede bilgi-işlemsel düşünme (Şekil 3) aşağıdaki kavram ve yaklaşımlar temelinde özetlenebilmektedir (Berry, 2015).



Şekil 3. Bilgi – İşlemsel düşünme kavramlar ve yaklaşımlar (Berry, 2015)

Bilgi-işlemsel düşünme becerisini kazandırmaya yönelik uygulamalar ve araçlar geliştikçe tanımların çerçevesi de yeniden ele alınmaktadır. Bu çerçevede NSF (National Science Foundation) lise düzeyinde yürütülecek bilgi-işlemsel düşünmeyi geliştirecek eğitimler için 7 temel fikri öne çıkarmaktadır (URL-1).

- Bilgi-işlem yaratıcı bir insan faaliyetidir.
- Soyutlama, sorunları anlama-çözme ile ilgili kavramlara odaklanmak için bilgi ve ayrıntıları azaltmaktır.
- Veri ve enformasyon, bilginin oluşumunu kolaylaştırır.
- Algoritmalar, bilgi-işlemsel problemlere hızlıca çözümler üretmek ve geliştirmek için araçtırlar.
- Programlama, bilgisayarca ürünler üreten yaratıcı bir süreçtir.
- Dijital cihazlar, sistemler ve onları birbirine bağlayan ağlar, problemleri çözmede sayısal yaklaşımlar geliştirir ve destekler.
- Bilgi İşlem; bilim, sosyal bilimler, beşeri bilimler, sanat, tıp, mühendislik ve ticaret gibi diğer alanlarda yenilikler sağlar.

CCEA (2017) ise Bilgi – işlemsel düşünme alt becerilerini;

Mantıksal akıl yürütme: Bir problem durumu üzerinde tahmin ve analiz ederek düşünme,

Algoritma: Adım adım ve sıralı bir şekilde çözüme ulaşacak şekilde düşünme.

Ayrıştırma: Bir problemi küçük parçalara ayırarak düşünme.

Soyutlama: Bir problemin çözümüne giderken gereksiz bilgileri dışarda bırakarak düşünme.

Modelleme ve genelleme: Benzerlikleri fark ederek yeni durumlarda kullanma,

Değerlendirme: Yargı oluşturma şeklinde ifade etmektedir.

Tanımlamalara bakıldığında bilgi-işlemsel düşünme becerisinin çoğunlukla bir problemin çözümü sırasında gerekli olduğu veya gözlenebilir olduğu düşünülebilir. Dolayısıyla birçok araştırmacı bilgi-işlemsel düşünme süreçlerini doğrudan veya dolaylı olarak problem çözme ile ilişkilendirmektedir. Bu çerçevede bu ilişkinin ortaya konulmasına yönelik değerlendirmeler yer almaktadır.

2. 1. 2. Bilgi – İşlemsel Düşünme ve Problem Çözme Becerisi

Problem çözme becerisi, çoğunlukla bilgi-işlemsel düşünme içerisinde ayrı bir beceri olarak ele alınırken bazı durumlarda diğer alt beceriler ile ilişkili olarak değerlendirilmektedir. Nitekim yapılan tanımlamalarda bilgi-işlemsel düşünme ile problem çözme sürekli olarak birlikte anılmaktadır. Örneğin Wing (2008) bilgi-işlemsel düşünmeyi gerçekte bir çeşit analitik düşünme şekli olduğunu belirtmekte ve bilgisayar biliminin temel kavramlarına dikkat çekerek problemi anlama, çözme, tasarlama ve insan davranışlarına uyarlama yöntemi olarak değerlendirmektedir. Yapılan tanımlamalarda problem çözme becerisi bilgi-işlemsel düşünme içerisinde bir yandan ayrı bir beceri olarak ele alınırken diğer yandan bilgi-işlemsel düşünme becerisinin alt becerileri ile ilişkili olarak da değerlendirilmektedir. Bu çerçevede; Barr, Harrison ve Conery (2011) bilgi-işlemsel düşünme becerilerinin genel özelliklerini şöyle ifade etmektedir:

- Problemleri, çözmeye yardımcı olmak için, bilgisayar ve diğer araçları kullanmamızı sağlayacak bir biçimde formüle etme,
- Verileri mantıksal olarak organize ve analiz etme,
- Soyutlama yolu ile verileri temsil etme (modeller ve benzetimler gibi),
- Çözümleri algoritmik düşünme yoluyla otomatik hale getirme,
- Adımların ve kaynakların en etkili kombinasyonunu elde etmek amacıyla olası çözümleri belirleme, analiz etme ve uygulama,
- Problem çözme sürecini genelleme ve diğer çeşitli durumlara aktarma.

Diğer taraftan National Research Council -NRC (2008) bilgi-işlemsel düşünmenin alt becerilerini; problem gösterimi, soyutlama, ayrıştırma, simülasyon, doğrulama ve tahmin

etme şekline ifade ederken; alt becerilerin tanımlanmasında problem çözme ile ilgili alt becerilere yer vermektedir. Wing (2008) de bilgi-işlemsel düşünmeyi problem çözmeden bağımsız ele almamıştır. Tanımlamalarında bilgi-işlemsel düşünmeyi sistem tasarlama ve insan davranışlarını anlama için genel analitik bir yaklaşım olarak tanımlamışlardır. Bilgi-işlemsel düşünme, bilgi işlem ve bilgisayar bilimi için temel kavramları çizer; problem gösterimi, soyutlama, ayrışma, simülasyon, doğrulama ve tahmini gibi uygulamaları da içerir. Bu uygulamalar aynı zamanda, matematik disiplininde modelleme, akıl yürütme ve problem çözmede merkezi bir noktadır. Wing (2008) ve NRC(2008) tanımlamaları, bilgi-işlemsel düşünce ifadesinin yeni yaygınlaşmaya başladığı yıllarda bu beceriyi Tablo 1'deki alt beceriler çerçevesinde ele alındığını göstermektedir.

Tablo 1. Wing(2008) ve NRC (2008)'e Göre Bilgi – İşlemsel Düşünme Alt Becerileri

Wing(2008) tanımlaması	National Research Council (2008) tanımlaması	Açıklama
	Mantıksal akıl yürütme	Bir problem durumu üzerinde tahmin ve analiz ederek düşünme.
Soyutlama	Soyutlama	Gereksiz ayrıntıları dışarda bırakarak düşünme.
Algoritmik düşünme	Algoritmik düşünme	Çözüme ulaşmak için adım adım düşünme.
Otomasyon		İş gücünde tasarruf süreci.
Ayrıştırma	Ayrıştırma	Problemi anlaşılır küçük parçalara ayırarak çözümünü düşünme.
Ayıklama		Mantıksal analiz ve değerlendirmeler göre hataları tespit edip giderme süreci.
Genelleme	Modelleme ve genelleme	Benzerlikleri yeni durumlarda fark ederek çözümleri genelleyebilme.
	Değerlendirme	Yargı oluşturma.

Bilgi-işlemsel düşünme ile ilgili yapılan tanımlama ve uygulamalar incelendiğinde, bu kavramın bir problemi çözebilmek için gerekli olan düşünme sürecini yönetmeye dönük bir strateji olduğu görülmektedir (Barr, Harrison ve Conery, 2011).

Benzer şekilde bilgi-işlemsel düşünme ile ilişkili kavram ve becerilere yönelik tanımlamaları karşılaştıran bir çalışmada birçok alt beceride ortaklaşmalar söz konusu olduğu görülmektedir. Bilgi-işlemsel becerilerinin kazandırılması sırasında öğrencilerin ilgili alt becerileri farklı aşamalarda kazandıklarından hareketle (Bocconi, Chioccariello, Dettori, Ferrari ve Engelhardt, 2016) bilgi-işlemsel düşünme kavramlarını ve alt becerilerini işleyen farklı çalışmaları analiz ederek, bu düşünme becerisine ilişkin modellemeleri Tablo 2'de ortaya koymuştur.

Tablo 2. Farklı Araştırmalarda Bilgi – İşlemsel Düşünmeye İlişkin Alt Becerilerin Karşılaştırılması (Bacconi ve diğ., 2016)

Barr ve Stephenson, 2011	Lee ve diğerleri, 2011	Grover ve Pea, 2013	Selby ve Woollard, 2013	Angeli ve diğerleri, 2016
Soyutlama	Soyutlama	Soyutlama ve örüntü genelleme	Soyutlama	Soyutlama
Algoritmalar ve prosedürler		Algoritmik yaklaşım ve akış kontrolü	Algoritmik düşünme	Algoritmalar (Sıralama ve akış kontrolü)
Otomasyon	Otomasyon			
	Analiz			
		Koşul mantığı		
Problemi ayrıştırma		Yapısal olarak problem ayrıştırma (modülleme)	Ayrıştırma	Ayrıştırma
		Test ve sistematik hata denetimi		Test
		Etkililik ve performans kısıtları	Değerlendirme	
			Genelleme	Genelleme
		Yinelemeli ve paralel düşünme		
Paralelleştirme				
Simülasyon				
		Sembol sistemleri ve gösterimler		
		Sistematik bilgi işleme		

Elbette bilgi-işlemsel düşünmenin gelişimi sürecinde öğrenciler alt becerilerin tümünü her zaman gerçekleştiremeyebilirler. Öğrencilerin bu alt becerileri sergileme durumları, ilgilenilen problemlerin doğasına göre farklı sırada gerçekleşebilir veya problemin özelliklerine bağlı olarak hiç gerçekleşmeyebilir.

Yukarıda özetlenmeye çalışılan farklı araştırmacıların değerlendirmeleri; bilgi-işlemsel düşünmenin programlama sürecindeki birçok zihinsel işlem ile ilişkili olduğuna ve uygun etkinliklerle geliştirilebileceğine işaret etmektedir (Wing, 2006). Bu noktada da programlama, bilgi-işlemsel düşünme becerilerinin gelişiminde önemli bir araç olarak ele alınmaktadır. Nitekim programlama ile problemler çözülürken doğrudan veya dolaylı olarak bilgi-işlemsel becerilerin gelişimine de katkıda bulunulmaktadır. Nitekim araştırmacıların birçoğu bilgi-işlemsel düşünme becerisinin bilgisayar bilimleriyle ilişkili olarak eğitim-öğretim süreçleri içinde özellikle programlama dersleri içerisinde problem çözme etkinlikleriyle öğrencilere kazandırılabileceğini ortaya koymaktadırlar (Barr, Harrison ve Conery, 2011; ISTE, 2016; Wing, 2006).

Bu noktada bilgi-işlemsel düşünme ile ilişkili kavram ve becerilere yönelik tanımlamaları karşılaştıran bir çalışmada, birçok alt beceride değerlendirmelerin ortak

olduğu görülmektedir. Bilgi-işlemsel düşünme becerilerinin kazandırılması sırasında öğrenciler alt becerileri farklı aşamalarda kazanırlar. Bu aşamalarda kazanılan becerilere ilişkin farklı modellemeler söz konusudur.

Wing (2008) bilgi-işlemsel düşünmeyi daha kapsamlı olarak ele alarak; problem çözme, sistem tasarlama ve insan davranışlarını anlama için genel analitik bir yaklaşım olarak tanımlamıştır. Araştırmacıya göre, bilgi-işlemsel düşünme, bilgi işlem ve bilgisayar bilimi için temel kavramları çizer; problem gösterimi, soyutlama, ayrışma, simülasyon, doğrulama ve tahmini gibi uygulamaları da içerir. Bu uygulamalar, aynı zamanda, çok sayıda bilimsel ve matematik disiplininde modelleme, akıl yürütme ve problem çözmede merkezi bir noktadır. Bilgi-işlemsel düşünme ile ilgili yapılan tanımlama ve uygulamalara bakıldığında, bu kavramın bir problemi çözebilmek için gerekli olan düşünme sürecini yönetmeye dönük bir strateji olduğu görülmektedir (Barr, Harrison ve Conery, 2011). Bu anlamda araştırmacılar bilgi-işlemsel düşünme becerisinin genel özelliklerini aşağıdaki gibi sıralamaktadır:

- Problemi formülleştirme,
- Verileri mantıklı hale getirme ve analiz etme,
- Verileri sunma (modeller ve simülasyonlar gibi),
- Çözümleri otomatikleştirme (algoritmik düşünme yardımıyla),
- En etkili basamaklar ve materyaller ile olası çözümler üretme ve uygulama,
- Problem çözme sürecini yaygınlaştırma

Yapılan birçok çalışmada bilgi-işlemsel düşünme içerisinde yaratıcılık, kritik düşünme ve problem çözme gibi becerilerin öne çıktığı düşünüldüğünde, bu durum programlamanın problem çözme becerisi geliştirmede önemli rol oynayabileceğine işaret etmektedir (Ananiadou ve Claro, 2009; Binkley ve diğ., 2012). Bu doğrultuda aşağıda programlama ve problem çözme ilişkisine yer veren değerlendirmeler ele alınmaktadır.

2. 1. 3. Programlama ve Problem Çözme

Programlama sürecinde öğrencilerin yaşamış oldukları bilişsel süreçlere ilişkin çalışmalar değerlendirildiğinde matematiksel problemlerin çözme aşamalarının bu çalışmalara önemli katkılar yapmış olduğu görülebilir. Bu çerçevede öğrencilerin bilişsel süreçlerine yönelik çalışmalardan birisi de programlama sürecinin aşamalandırılması veya süreçteki görevleri yerine getirirken gerçekleştirdikleri davranışların sınıflandırılmasıdır. Yapılan tanımlamalar incelendiğinde yaşanan bilişsel sürece ilişkin aşamaların temel olarak programlama sürecinde öğretilen bilgiler temelinde şekillendiği söylenebilir. Genel

olarak programlama öğrenimi süresince birbirleriyle ilişkili üç tip programlama bilgisi söz konusudur (Bayman ve Mayer, 1988)

- Sözdizimsel (Syntactic) Bilgi: Belirli bir programlama diline ait kullanım kurallarının bilgisi,
- Kavramsal (Conceptual) Bilgi: Programlama kavramlarının ve prensiplerinin bilgisi,
- Stratejik (Strategic) Bilgi: Programlama ile ilgili problem çözme becerisi Bu bilgiler seçilen programlama öğretimi yolu, programlama dilinin yapısı, programlama yapılan editörün arayüzü gibi hususlar çerçevesinde şekillenebilmektedir.

Programlama araçlarını kullanarak okul çağındaki çocukların problem çözme etkinlikleri gerçekleştirmeleri Papert'in (1980) Logo programlama dilini geliştirmesiyle belirgin olarak gündeme gelmeye başlamıştır. Logo ile daha çok matematik dersindeki birçok problemin çözümü için araştırmalar yapılmıştır. Bu ortamda öğrencilere gerekli kodlama yapıları sunulmakta, öğrenciler bu kodları yazarak kaplumbağa karakterinin ekranda problemi çözmesini izleyebilmektedirler. Programlama ile problem çözme aşamalarına yönelik yapılan ilişkilendirmelerin bazılarında öğrencilerin programlama sürecinde geçirdikleri aşamalar ile tipik problem çözme aşamaları eşleştirilmeye çalışılırken, bazılarında problem çözme süreci programlama içerisinde bir alt beceri kazanma süreci olarak ele alınmaktadır. Tipik olarak daha çok matematiksel problemlerin çözümü için Polya (1980) aşağıdaki adımları önermektedir.

1. Problemin anlaşılması,
2. Çözüm ile ilgili stratejinin seçilmesi,
3. Seçilen stratejinin uygulanması,
4. Çözümün değerlendirilmesi

Bilgisayar programlamada özel öğretim stratejileri ile ilgili çalışmalar yıllar öncesinden beri bu yöntemlerin öğrencilerin problem çözme becerilerini arttırabileceğine yönelik değerlendirmelere sebep olmuştur (Deek, Turoff ve McHugh,1999; Hung, 2008) Bu noktada bilgisayar programlama ile desteklenen problem çözme çalışmalarının en az 4 farklı adımı içermesi beklenir. 1) problemi anlama ve tanımlama, 2) çözümü planlama ve gerekli bilgiyi toplama, 3) çözümü tasarlama ve uygulama, ve 4) sonuçları doğrulama ve sunma. PISA 2003 raporları çerçevesinde bu adımları daha da derinleştirilerek tipik problem çözme aşamalarını aşağıdaki gibi ifade edilmiştir.

Problemin tanımlanması

- İlgili bilgi ya da sınırlılıkların tanımlanması,
- Olası seçenek ya da çözüm yollarının sunulması,
- Çözüm stratejilerinin seçilmesi,

- Problemlerin çözülmesi
- Çözümün ifade edilmesi veya kontrol edilmesi
- Sonuçların paylaşılması (OECD, 2004).

Programlama sürecine yönelik aşamalar zaman zaman programlamanın temel öğeleri, algoritma mantığı, programlama ara yüzlerinin sürece etkisi de değerlendirilerek ortaya konuşmuştur. Bu çerçevede bir başka araştırmada, problem çözmeyi programlama çerçevesinde ele alan bir aşamalandırma (McCain, 2007). İngilizce 4 adet "D" harfi ile başlayan problem çözme aşamalarını "Problem çözmenin 4D'si" şeklinde ifade etmektedir. İngilizce kelimelerine bakıldığında problem çözme sürecinin her bir adımını ifade ederken cümlelere ait kelimelerin D harfi ile başlamasından ötürü "problem çözmenin 4D'si" şeklinde isimlendiren süreç tanımı aşağıdaki gibidir:

1. Çalışmaya başlamadan önce problemin tespiti. (Define)
2. Problemin çözümü için bir plan tasarlanması. (Design)
3. Problemin ele alınması. (Do tackle)
4. Süreç ile ilgili bilgilendirme, değerlendirme. (Debrief)

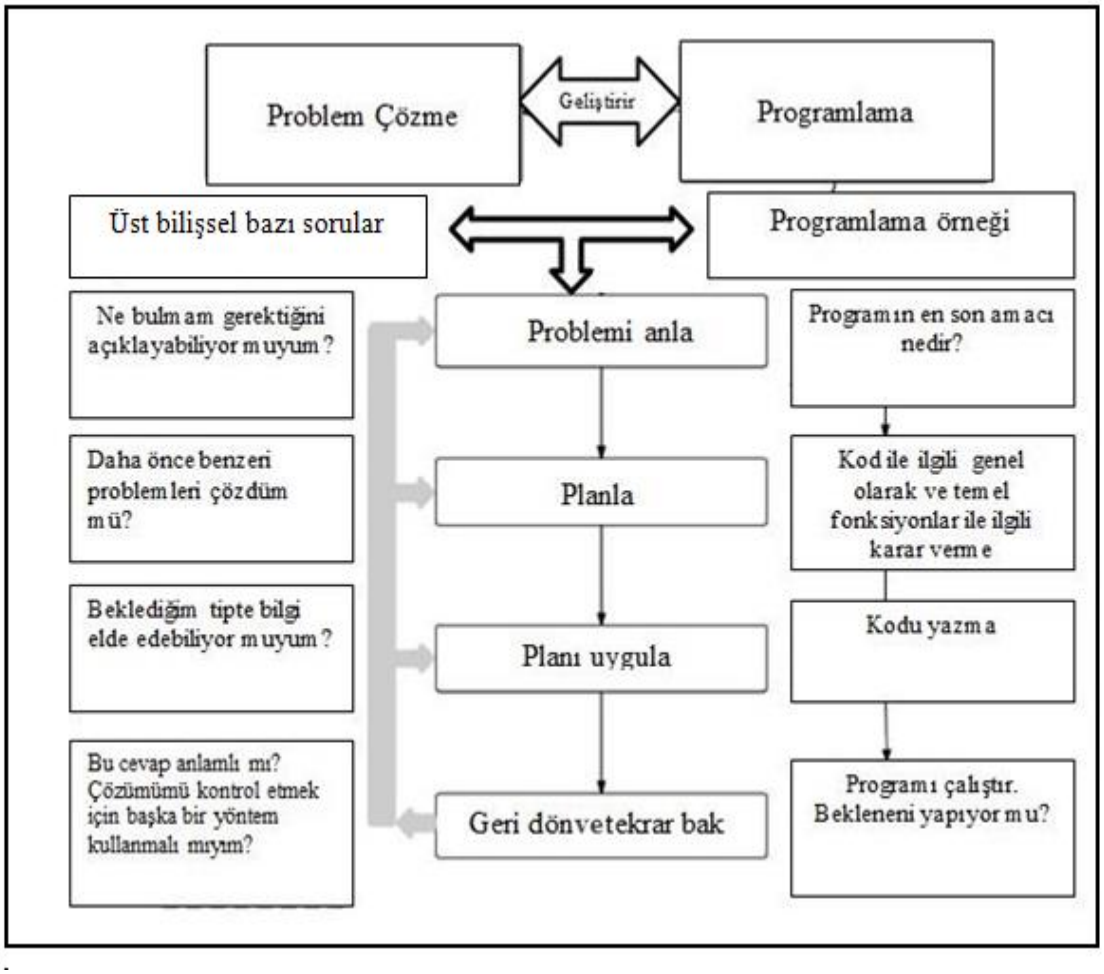
Garner (2003) ise programlama sürecinin tanımlanmasına ilişkin olarak; daha çok algoritma geliştirmeyi öne çıkararak bu adımları biraz daha farklılaştırarak programlama sürecinde öğrencilerin problemin analizi, algoritma geliştirme, geliştirilen algoritmayı uygulama şeklinde programlama süreçlerini aşamalandırmıştır.

Yukarıda gerek problem çözme aşamaları ile gerekse programlama sürecine ilişkin aşamalar arasındaki benzerlikler birçok araştırmacının dikkatini çekmiş olup, programlamanın doğası bu ilişkiler üzerinden açıklanmaya çalışılmaktadır. Bu çerçevede (Çetin, 2012) aşamalar arasındaki bu ilişkiyi Tablo 3'teki gibi ortaya koymaktadır.

Tablo 3. Çetin (2012)'e Göre Problem Çözme ve Programlama Süreci

Problem Çözme Süreci	Programlama Süreci
Problemi anlama	Analiz
Çözüm yollarını bulma	Tasarım
Çözümün uygulanması	Geliştirme
Problemin çözülmesi ve değerlendirilmesi	Test

Nance (2016) ise problem çözme ve programlama süreçlerinin aslında birebir aynı olabileceği fikrinden hareketle aralarındaki ilişkiyi Şekil 4'teki gibi örneklendirerek özetlemiştir.



Şekil 4. Problem çözme ve programlama arasındaki ilişkiye yönelik kavramsal çerçeve (Nance, 2016)

Problem çözmenin programlama ile ilişkisi küçük yaştan itibaren çocuklara programlamanın öğretilmesi düşüncesini doğurmuş ve bu düşünce çerçevesinde farklı araçlar oluşturularak uygulanmaya başlanmıştır. Bu noktada aşağıda kısaca bu alanda yapılan araştırmalar ele alınmaktadır.

2. 1. 4. Çocuklarda Bilgi – İşlemsel Düşünme Becerisinin Geliştirilmesi ve Programlama Öğretimi

Prensky (2001) günümüze işaret ederek, gelecekteki toplumdan donanım, yazılım, nano teknoloji gibi dijital bilgilere sahip ve teknolojiyi üretim için kullanabilen bireylerin beklendiğini ifade etmektedir. Bu çerçevede dijital yerliler olarak değerlendirilen günümüz çocuklarının sadece teknoloji okuyucusu olması değil, aynı zamanda üreterek de bu alana katılmaları gerektiğini vurgulamaktadır. Teknoloji alanına üretici olarak katılmanın önemli

bir yolu programlama becerisinin geliştirilmesidir. Nitekim Olgun (2014) küçük yaşlarda öğrencilerin programlama ile tanışmasının gelecekte hayatlarında karşılaşacakları problemleri çözme konusunda olumlu etkisinin olabileceğini ifade etmiştir.

Günümüzde özellikle programlama ile yeni tanışan kullanıcılar için ortaya çıkan en büyük engelin, geleneksel programlama dillerinin yapılarındaki karmaşıklıktan dolayı öğrenilmelerinin zor olması görülmektedir. Nitekim programlama öğretimi sürecinde

- Bilgisayar okur-yazarlık bilgisi,
- Donanım bilgisi,
- Programlamaya ilişkin temel kavramlar bilgisi,
- Programlama dilinin "dil yapısı" bilgisi,
- Problem çözme becerisi gibi bilgi ve becerilerin öğrenilmesi gerekmektedir.

Geleneksel programlama dillerinin veya araçlarının çoğu insanlar için çok karmaşık ve üst düzey düşünme becerisi gerektirdiği bilinmektedir. Bununla birlikte programlama öğretilirken sıklıkla programlamanın, öğrenenlerin geçmiş bilgilerinden ve gerçek yaşam uygulamalarından bağımsız olduğu düşünülmektedir (Gomes ve Mendes, 2007). Bu doğrultuda literatürde birçok çalışma öğrencinin programlama eğitimini zor bulduklarına işaret etmektedir (Yağcı, 2016). Bu çerçevede Papert (1980) yapılandırmacı programlama ortamlarının, çocukların düşüncelerini sorgulamalarını cazip hale getiren, kendi fikirlerini daha fazla somutlaştırabildikleri ve dolayısıyla derinlemesine düşünme ile karşı karşıya kalabilecekleri şekilde geliştirilmesi gerektiğini ifade etmektedir. Bu doğrultuda tasarlanan ilk uygulamalardan biri olarak örnek gösterebileceğimiz Logo, uzun yıllar problem çözmede kullanılan önemli bir araç olmuştur. Programlama sürecini doğrudan problem çözme üzerine inşa eden Hung (2008) ise var olan problem çözme tabanlı öğretim yöntemlerinin programlama çalışmaları açısından okul çocukları ile uyumlu olması gerektiğini iddia etmiştir.

Programlama öğretiminin faydaları, öğretim yöntemleri ve ortamlarındaki gelişmeler son yıllarda özellikle ilköğretim düzeyindeki programlama öğretimine ilgiyi arttırmıştır. Önceleri küçük yaşta çocuklar yazım anlamında zorlandığı için bu seviyede programlama oldukça zor bir kabiliyet olarak görülmüştür (Grover ve Pea, 2013; Resnick ve diğ., 2009). Küçük yaştaki öğrencilerin çektikleri bu zorluk, zamanla gelişen ve benimsenen blok tabanlı platformlar ile ortadan kalkmaya başlamıştır. Bu amaçla Scratch, Toontalk, Stagecast creator, Alice, Blockly, Code.org gibi kullanması kolay olan çocukların kullanımına uygun ortamlar geliştirilmiş ve kullanılmaya başlanmıştır (Burke, 2012; Denner Werner ve Ortiz, 2012; Lee, 2010). Bu yeni ortamların en önemli özelliği öğrencilerin temel bir eğitimle problem çözme süreçlerini kendilerinin yapılandırabileceği ortama imkân tanımalarıdır. Scratch, Alice, Code.org vb. gibi bu yeni ortamların Logo'nun

ortaya koyduğu öğrencinin kendi çözüm yolunu oluşturabilmesi yaklaşımlarının farklı biçimlerde ele alınışıyla ortaya çıktığı değerlendirilmektedir (Utting, Cooper, Kölling, Maloney ve Resnick, 2010).

2. 1. 4. 1. Blok Tabanlı Kodlama Ortamları

Programlama dillerinin arayüzlerinin genel anlamda ikiye ayrıldığı görülmektedir (Chang, 2014; Grover ve Pea, 2013; Sáez-López, Román-González ve Vázquez-Cano, 2016). Birinci tür arayüzlerde Java veya C++ gibi geleneksel programlama dilleri bilgisayarların düşünme biçimini çok yakın temsil eder. Delphi, Visual Basic, Visual C# gibi grafiksel programlama dilleri insan diline daha yakın bir temsil kullanır. Grafiksel programlama ortamlarının kullanımı, geleneksel metin tabanlı dillere göre nispeten daha kolaydır ve bu ortamlar programlamanın sözdizimi sorunlarından kaçınarak tasarım ile üretme üzerine odaklanmalarına izin verir. Son yıllarda ise küçük yaştaki çocukların da programlama yapabilmesi fikrinden yola çıkılarak görsel programlama ortamlarının da daha basitleştirilmesi gerektiği düşüncesi ortaya çıkmıştır. Bu düşünceden hareketle; muhtemel sözdizimi hatalarını azaltarak ve çoğu metinsel programlama dilinden daha küçük bir komut seti sağlayarak programlamaya yönelik bilişsel engelleri azaltan blok tabanlı ortamlar geliştirilmeye başlanılmıştır. Bu tür ortamlar tipik uygulama programları yazma amaçlı olmayıp daha çok problem çözme düşüncenin geliştirilmesi ve yerleşmesi için oluşturulduğundan bu ortamlar programlama ortamlarından çok kodlama ortamları olarak anılmaktadır. Bu çerçevede BTKO'nun okul çağındaki çocuklar için bilgi-işlemsel düşünme becerisini geliştirmek için zengin özelliklere sahip, düşük eşik ve yüksek tavana sahip olup, her seviyeden öğrenciye hitap edebilir olması beklenir. Bu ortamların özellikle öğrencilerin takıldıkları yerde düşünme süreçlerini destekleyebilen ve farklı durumlara transfer desteği sunan, sistematik ve sürdürülebilir biçimde tasarlanması önerilmektedir (Repenning, Webb ve Ioannidou, 2010).

BTKO, kullanıcıların komut dosyaları oluşturmak için bulmaca parçaları gibi sürükleyecekleri Lego benzeri bloklarda bir dizi programlama komutları sağlar. Bu komut dosyaları genellikle bir oyun oluşturmak, senaryolar temelinde hikaye anlatmak veya daha fazlasını yapmak için kullanılabilecek karakterleri veya diğer görüntüleri kontrol eder. Birçok BTKO'da sınırlı yazıma imkân veren renkli tasarlanmış etkileşimli arabirimler bulunur ve çocuklar kolaylıkla bu ortama uyum sağlarlar. Bununla birlikte, çocuklar için BTKO geliştiricileri hedef yaş grubunun gelişim düzeylerini ve sınırlamalarını dikkate alır (Meerbaum-Salant, Armoni ve Ben-Ari, 2013). Bu çerçevede öğrenilmesi kolay ve görselliği ön planda tutan Alice, Scratch, Microsoft Small Basic, Toontalk, SmallBasic,

Blockly, Mıt App Inventor, KoduGameLab ve Stagecast Creator gibi programlama ortamları geliştirilmiştir (Demirer ve Sak, 2016). Bu ortamlar, gerek sınıf içi etkinliklerde gerekse okul dışında verilen ödevlerde farklı özellikleriyle kodlama eğitimine kolaylık sağlayabildiklerinden öğretmenler öğrenci ihtiyaçlarına göre farklı BTKO'ları tercih edebilmektedirler. Bu ortamlardan önde gelenlerinden birisi olan Scratch, kullanıcıların bir program oluşturmak için görsel blokları sürükleyip bıraktığı bir programlama yazılımıdır (Meerbaum-Salant, Armoni ve Ben-Ari, 2013). Fakat Scratch arayüzünde öğrencilerden boş bir sahnede kendi fikirleri doğrultusunda ürün ortaya koymaları beklenmektedir, yani öğrenci hem problemi belirlemeli hemde çözümünü düşünmelidir. Bu ortamlardan yaygın kullanılan diğer bir arayüz Code.org olup, bu çalışmada öğrencilere her sahnesinde sunduğu farklı problem durumları göz önünde bulundurularak BTKO olarak Code.org tercih edilmiştir.

2. 1. 4. 2. Neden Code.org?

Code.org, bilgisayar bilgisinin yaygınlaştırılması ve bilgi-işlemsel düşüncenin geliştirilmesi için "eğitmek, savunmak ve kutlamak" şeklinde ifade edilen üç faktörlü bir yaklaşım geliştirmiştir (Wilson, 2013). Bu çerçevede Code.org bir organizasyon olarak bilgisayar bilimleri dersleriyle ilgili yeni düşünceler geliştirilmesini ve değişiklikler yapılmasını savunmakta ve katılımcıları cesaretlendirmektedir (Wilson, 2013). Bu çerçevede Code.org ortamında motive edici özellikler, öğretim materyalleri ve ders planları vardır, dolayısıyla bu ortamın yeni başlayanlara programlama yapmak ve öğrencilere bilgisayar bilimi temelini kazandırmak isteyenler için iyi bir seçenek olabileceği düşünülmektedir. Benzer biçimde bu ortamın bilgi-işlemsel düşünme becerisi ve ilgili alt becerileri geliştirme ve takip etmede önemli katkı sağlayabildiği değerlendirilmektedir (Gülbahar ve Kalelioğlu, 2014; Lu ve Fletcher, 2009; Ozoran, Cagiltay ve Topalli, 2012; Shin ve Park, 2014; Su, Yang, Huang ve Hwang, 2014).

Literatürde bu ortamların programlama öğretimindeki etkisini belirlemek için çok sayıda çalışma yapılmıştır (Gülbahar ve Kalelioğlu, 2014; Lu ve Fletcher, 2009; Ozoran, Cagiltay ve Topalli, 2012; Shin ve Park, 2014; Su, Yang, Huang ve Hwang, 2014). Sistem üzerinde kayıtlı olan öğrenci sayısı anlık değişmekle birlikte araştırma esnasında 24.709.631 ile oldukça dikkat çekicidir. Bu ortamda farklı yaş gruplarına uygun alternatif uygulamalar bulunmaktadır. Her bölüme ait bir sahne içeren sistemde öğrenciler sürükle bırak hareketleriyle akış oluşturmakta ve çalıştır komutu ile hedefe ulaştıkları zaman bölümleri tamamlamaktadır. Her yıl kodlama etkinliklerine dikkat çekmek için "Hour of Code" etkinliği düzenlemekte ve tüm dünyadan katılımcıların bu etkinliği tamamlayarak

sertifika alması sağlanmaktadır. Code.org ortamının bu kadar yaygın kullanımının nedenleri arasında, destekçilerinin Microsoft ve Google gibi büyük markalar olması da gösterilebilir. Ayrıca Code.org bünyesinden diğer BTKO'nun bir çoğuna aktivite partnerleri bölümünden bağlantı sağlanabilmektedir. Code.org ile benzer mantıkta ve programlama yapılarında bloklara sahip olan diğer ortamlar genellikle kullanıcılara tamamlaması istenilen bölümler sunulmamaktadır. Diğer BTKO'larda bazen tüm bloklar ilgili bölümlerin altında bulunmakta ve kullanıcı istediği senaryoya uygun karakter ve dekorları tasarlayıp kodlayarak ürününü oluşturabilmektedir. Diğer taraftan bu çalışmada ele alınacak olan 6. sınıf öğrencileri göz önüne alındığında programlama dillerinde aranacak en önemli özelliklerden birisinin ortamın sunduğu dil desteği olduğu düşünülebilir. Ayrıca Code.org ortamındaki öğretmen ipuçları da yapılacak ders tasarımları için değerlendirilebilir.

Bütün bu özelliklerin yanında, Code.org kendi web sitesinde aşağıdaki amaçlar çerçevesinde kullanılabileceğini özetlemektedir.

- Yaygın kullanım ve kurulum gerektirmeyen çevrimiçi platforma sahip olması,
- Ortam içerisindeki sahnelerin bilimsel dayanakları ve kaynak ders planlarının olması,
- Öğretmen arayüzü sayesinde kolay öğrenci takibi,
- Öğrencilerin sisteme kayıt edilmesi esnasında sınıf oluşturma sistemi ve atanan görseller ile kullanıcıların giriş yapabilmesi,
- Sistem tarafından öğrenci ilerlemelerine yönelik istatistiki bilgilerin tutulması,
- Ders-sahne eşleşmesi çerçevesinde programlama yapıları temelinde hazırlanmış oluşu,
- Öğrenciler için ilgi çekici ve renkli bir ortam oluşu,
- Öğrencilerin kendi ilerlemelerini görebilmesi.

Özetle, okullarda bulunan bilgisayar laboratuvarlarının sistem özellikleri düşünüldüğünde; kurulum gerektirmeden çevrimiçi çalışan bir ortam olması ve internete girebilen her cihazda çalışabiliyor olması Code.org seçimindeki en önemli etkenlerden biri olmuştur. Code.org'un Türkçe dil desteği, yaş gruplarına özel bilimsel dayanaklara göre hazırlanmış ve programlama yapıları üzerine inşa edilmiş gündelik yaşam problemleri sunan yapısı diğer BTKO'lara kıyasla seçilmesine sebep olmuştur. Sürecin yürütülmesi noktasında ise öğretmen hesabı ve öğrenci takibi özelliklerinin oldukça kolaylık sağlayacağı düşünülmüştür.

2. 1. 4. 3. Blok Tabanlı Kodlama Ortamlarında Problem Çözme Süreçleri

Bazı araştırmacılar problem çözme süreçlerini farklı programlama ortamları için süreci değerlendirme ve aşamalandırma çalışmaları gerçekleştirmektedirler. Örneğin Volet ve Lund (1994) öğrencilerin program planlama sürecine rehberlik etmek için beş adımlı biliş ötesi strateji adını verdikleri sıralamayı ortaya koymuş ve bunları; 1) problemin tanımlanması; 2) algoritma geliştirilmesi; 3) akış şeması veya sahte kod uygulaması; 4) kodlama ve 5) kodun uygulanması, hata ayıklanması ve programın iyileştirilmesi olarak açıklamıştır. Benzer biçimde Lister ve diğ. (2004) ise problem çözme süreçlerine benzeterek geleneksel programlama ortamlarındaki programlama süreçlerini Tablo 4'teki gibi özetlemektedir.

Tablo 4. Geleneksel Programlama Ortamlarındaki Problem Çözme Süreçleri

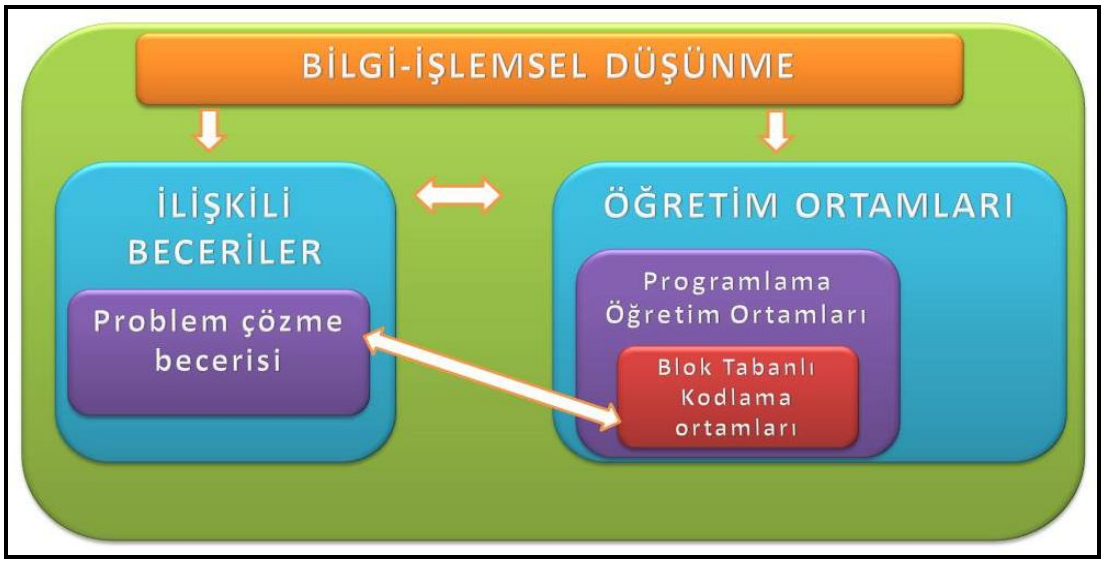
AŞAMA (SAHNE)	Herhangi bir alan (sahne) (Polya, 1980)	Herhangi bir alan (sahne) (Weiss, 1995)	Programlama alanı (sahnesi) (Lister, ve diğ., 2004)
1	Problemi Anlama • Tüm kelimeleri anla • İsteneni bul veya göster • Problemi yeniden ifade et • Problem şeması oluştur • Yeterli bilgi var mı?	Problem veya durumu seç veya seçilmesine izin ver	Açıklamadan problemi özetle
2	Planlama • Tahmin ve kontrol • Olasılıkları listeleme • Örüntüleri/istisnaları bulma • Modellemek/akıl yürütme (muhakeme etme) • Kullanılabilir/tekrar kullanılabilir algoritmalar • Daha basit bir problemi çözme • Geriye doğru çalışma • Kafada canlandırma (hayal etme)	Gözlemle, Düzenle ve Problem veya Durumu Tanımla	Problemi alt problemlere ayır
3	Planı gerçekleştirme (uygulama) • Tekrarlamak/devam etmek(sürdürme) • Deneme/hata/değişiklik	Problemin tüm bölümlerini sorgulayarak öğren	Alt problemleri alt çözümlere dönüştür
4		Olağan çözümleri görselleştir/Birini seç/ve rafine et	Özetlenen problem çözülene kadar alt çözümleri üst düzey çözümlere dönüştür (sentez)

Tablo 4'ün devamı

5	Geriye dönme • Yeniden göz geçirme/ değerlendirme • Sürekli geliştirme	Çözümleri çalıştır/ ve sonuçları izle	Çözümleri değerlendir ve problemin tüm sınırları doğrulanana kadar düzenle (yinele)

Programlama sürecini doğrudan problem çözme üzerine inşa eden Hung (2008) var olan problem çözme tabanlı öğretim yöntemlerinin programlama ekinlikleri açısından okul çağındaki çocuklar ile uyumlu olması gerektiğini iddia etmiştir. Bu uyumun gerçekleşebilmesi için çocukların problem çözdükleri ortamların onlara nasıl bir süreç yaşattığına ilişkin bilgi sahibi olunmalıdır. Bu bakımdan Code.org ve benzeri BTKO'larda öğrencilerin bilgi-işlemsel düşünmelerinin gelişmeleri çerçevesinde gerçekleşen davranışlarının belirlenmesi, problem çözme süreçlerinin değerlendirilmesi yapılacak öğretimlerin tasarımı ve uygulanması için önemlidir. Ayrıca elde edilecek aşamalara ilişkin göstergeler bu ortamların değerlendirilmesi noktasında da katkı sağlayabilir. Bu çerçevede BTKO için problem çözme süreçlerinin aşamalandırılmasına ilişkin az sayıda çalışma gerçekleştirilmiştir. Bu çalışmalardan birisinde Scratch gibi ortamları kullanarak mevcut problem çözme temelli öğretim yaklaşımlarını programlama hedeflerine ulaşmak için *"Bir örneğin amacını açıklamak, örneğin gereksinimlerini anlamak, örnek tasarım için bir plan önermek, örneği uygulama ve hataları ayıklamak*" adımlarını önerilmiştir (Ismail, Ngah ve Umar, 2010).

Özetle bu çalışma bilgi-işlemsel düşünme becerisi üst çerçevesinde problem çözme becerisi gibi becerileri, çocuklara programlama öğretimindeki yöntem ve uygulamaları, BTKO ve Code.org gibi araçları içeren bir yapıda ele almaktadır. Bu temel unsurların çalışma çerçevesindeki ilişkisi aşağıda özetlenmektedir (Şekil 5).



Şekil 5. Çalışmanın teorik çerçevesi

2. 1. 5. İlgili Çalışmalar

Son yıllarda BTKO giderek yaygınlaşan biçimde, farklı ortamlarda yapılan uygulamalarla gelişmektedir. Yapılan uygulamalar çerçevesinde kullanılan ortamlar, öğretim yöntemleri ve öğrenme çıktıları çerçevesinde gerçekleştirilen araştırmalar gün geçtikçe artmaktadır. Bu çalışmalar daha çok ortamların farklı değişkenler çerçevesinde değerlendirilmesi üzerine yoğunlaştığı görülmektedir. BTKO olarak Code.org'un kullanıldığı çalışma oldukça sınırlı olup, bu bölümde genel olarak BTKO çerçevesindeki çalışmaların problem çözme süreci ile ilişkili olanları üzerinden değerlendirmeler yapılmıştır. Bu noktada elde edilen sonuçlar ile bir BTKO olarak Code.org'da problem çözme süreçlerinin tanımlanmasında nasıl bir yol izlenmesi gerektiği belirlenmeye çalışılmıştır. Aşağıda öncelikle farklı BTKO'lar ile yürütülen çalışmalar, daha sonra Code.org kullanılarak gerçekleştirilen çalışmalar tartışılmaktadır.

2. 1. 5. 1. BTKO İle Farklı Becerilerin Geliştirilmesine Yönelik Çalışmalar

İlköğretim öğrencilerinin problem çözme becerilerindeki gelişimin belirlenmeye çalışıldığı bir çalışmada (Kalelioğlu ve Gülbahar, 2014) BTKO olarak Scratch kullanmışlar ve süreç sonunda bu ortamın problem çözme becerisi geliştirme noktasında önemli bir farklılığa neden olmadığı belirlenmiştir. Bir başka çalışmada Genç ve Karakuş (2012) bilgisayar oyunları tasarımı dersinde öğrencilerin tasarım sürecinde aktif katılımlarını,

matematiksel ve bilgi-işlemsel becerilerinin gelişimi sağlamak amacıyla grafiksel bir programlama dili olan Scratch ortamını kullanmışlardır. Çalışma sonucunda öğrencilerin özellikle Scratch hakkında olumlu görüşlere sahip oldukları, tasarımla öğrenmenin kalıcı bir öğrenme sağladığı ve blok destekli öğretim metodunu benimsedikleri sonucu ortaya çıkmıştır. Benzer biçimde bir başka araştırmada Second Life sanal dünyasında Scratch ile öğrencilerde bilgi–işlemsel düşünme, problem çözme ve programlama becerisi geliştirilmesi araştırılmıştır. Bu çalışmanın sonucunda öğrencilerin yapısalcı ortamda işbirlikçi bir şekilde problem çözme, bilgi-işlemsel düşünme ve programlama becerilerinde olumlu yönde gelişme olduğu belirlenmiştir (Pellas ve Peroutseas, 2016).

Chiu (2014) çalışmasında başlangıç seviyesindeki öğrencilerle Scratch programlama için problem çözme yaklaşımlarını incelemiştir. Öğrenciler problem çözme süreçlerini yönlendiren analiz, çözüm için tasarım, çözüm yolu, kodlama, test ve hata ayıklama adımlarını içeren çalışma kâğıtlarını kullanmışlardır. Bir dönem boyunca devam eden çalışma sonunda öğrencilerden kendi projelerini yapmaları istenmiş ve sonuç olarak öğrencilerin problem çözme yaklaşımlarıyla programlama yapmaya yönelik olumlu tutumlar gerçekleştirdikleri görülmüştür. Ayrıca bu yaklaşımın programlama için yararlı olduğu ve kendilerine etkili programlama projeleri yapmada katkı sağlayıcı olduğu sonucuna ulaşılmıştır.

Sakamoto, Takano, Washizaki ve Fukazawa (2013) ise çalışmalarında hazırladıkları ikon tabanlı android ara yüz ile bilgi-işlemsel düşünme becerisini öğrencilerde geliştirmek ve programlamaya yönelik motivasyonu arttırmak amacıyla geliştirilen uygulama sonucunda, uygulamanın programlama ve bilgi-işlemsel düşünme becerisini geliştirmede katkı sağlayıcı nitelikte olduğu görülmüştür.

BTKO'lar kullandıkları ara yüzlerin çekici özellikleri, etkinliklerin zaman zaman oyunlara benziyor olması gibi durumlarla genelde öğrencilerin bu ortamlarda geçirdikleri süreç ile ilgili olumlu değerlendirmeler yapmalarına sebep olmaktadır. Nitekim bu alanda farklı BTKO kullanılarak gerçekleştirilen çalışmalarındaki çocukların neredeyse tamamı bilgisayar öğrenmeyle ilgili olumlu bir tutum geliştirmiştir. (Bers ve diğ., 2014; Fessakis, Gouli ve Mavroudi, 2013; Kalelioğlu ve Gülbahar, 2014; Keren ve Fridin, 2014; Rogozhkina ve Kushnirenko, 2011). Benzer biçimde Denner, Werner ve Ortiz (2012) yaptıkları analiz çalışmalarıyla farklı BTKO kullanılarak yapılan programlama öğretimi çalışmalarının öğrencilerin üst düzey düşünme beceri gelişmeleri olan algoritmik düşünme, eleştirel düşünme, yaratıcı düşünme ve problem çözme becerilerine katkı sağlayabileceğini ortaya koymuştur.

2. 1. 5. 2. BTKO Olarak Code.org Kullanan Çalışmalar

Dünya üzerinde çok fazla kullanıcısı olmakla birlikte bilimsel araştırmalar noktasında Code.org ortamı kullanılan çalışmalar oldukça azdır. Bu çalışmalardan birisinde Kalelioğlu (2015) Code.org ortamının problem çözme için gerekli düşünme becerileri üzerine etkisini araştırmıştır. Araştırma deneysel desende yürütülmüş olup, t-test sonuçları, ilkokul öğrencilerine Code.org üzerinden programlama öğretiminin problem çözmeye yönelik yansıtıcı düşünme becerilerinde bir farklılığa sebep olmadığını göstermiştir.

Bir başka çalışmada Kod Saati ("Hour of Code") etkinliği çerçevesinde öğrencilerin bilgisayar programlarına yönelik tutumları ve programlama bilgisi üzerindeki etkileri araştırılmıştır. İki farklı üniversiteden öğrencilerin katıldığı uygulamada Code.org uygulamalarının öğrencilerin programlamaya yönelik tutumlarında olumlu katkıları olduğunu, ancak öğrencilerin programlama becerilerinde anlamlı bir değişim olmadığını ortaya koymuştur (Du, Wimmer ve Rada, 2016).

Mallios ve Vassilakopoulos (2015) çalışmalarında yine Kod Saati uygulamaları çerçevesinde öğrencilerin programlama sürecinde bireysel ihtiyaçlarını dikkate alan anketleri kullanmışlardır. Bu araçlar çerçevesinde geliştirdikleri öğretim yöntemleri ile öğrencilerin Code.org gibi ortamlar için öğrenme ihtiyaçlarının belirlemesinin öğrenme performanslarını arttırabileceği yönünde sonuçlara ulaşmışlardır.

2. 2. Literatür Taramasının Sonucu

BTKO'lar ile yapılan çalışmalar değerlendirildiğinde daha çok öğretim sürecinin sonucunda ortaya çıkan etkilere odaklı çalışmalar gerçekleştirildiği görülmektedir. Öğrenme performansı, programlama, problem çözme, bilgi-işlemsel düşünme vb. becerilerin gelişimi, tutum gelişimi gibi durumlar süreç sonunda odaklanılan noktalar olarak karşımıza çıkmaktadır. Literatür taraması sonucunda farklı ülkelerin küçük yaştaki gruplar için öğretim programlarına BTKO üzerinden problem çözme etkinliklerini dahil ettikleri görülmektedir. Benzer biçimde bu süreçteki uygulamalar için hangi ortamların kullanılması gerektiği, etkinliklerin hangi seviye için ne şekilde tasarlanması gerektiğine ilişkin değerlendirmeler bu çalışma için yol gösterici olmuştur.

BTKO'larda problem çözme sürecinin nasıl gerçekleştiğini inceleyen bu çalışmada, metin tabanlı ve görsel tabanlı olan geleneksel programlama sürecine yönelik tanımlamalardan da yararlanılmıştır. Bu noktada kullanılan ara yüz, programlama yaklaşımı ve öğretim yöntemlerinin geleneksel programlama ortamlarında nasıl ele alındığına ilişkin değerlendirmeler bu çalışmada da dikkate alınmıştır.

Özetle, BTKO'larda süreci belirlemeye olan ihtiyaç, süreci yaşatmaya yönelik öğretim yöntemlerine örnekler, Code.org ortamının seçimi ve sürecin tanımlanmasına yönelik çerçeve örneklerini elde edilmesi noktasında bu çalışma literatürden elde edilen sonuçları değerlendirmiştir. Çalışmanın şekillenmesinde literatürün katkısı, yukarıda ifade edilmeye çalışılan konu başlıkları ve yararlanılan araştırmaları gösterecek şekilde Tablo 5'te özetlenmektedir.

Tablo 5. Araştırmanın Şekillendirilmesine Literatürün Katkısı

Çalışma Bölümü	Kaynaklar
Bilgi-işlemsel düşünme ve problem çözmeye ilişkin çerçeveler	Wing (2008); Bocconi ve diğerleri (2016)
Araştırma problemlerinin belirlenmesi	Nance (2016); Resnick ve diğerleri (2009)
Veri toplama araçlarının seçilmesi Araştırma yönteminin belirlenmesi	(McMillan ve Schumacher, 2010)
Uygulamanın yapılması	Kalelioğlu ve Gülbahar (2014) ; (Wilson, 2013)

3. YÖNTEM

Bu araştırma, ortaokul öğrencilerinin BTKO yürütülen programlama eğitimi sırasında problem çözme süreçlerini incelemektedir. Bu doğrultuda bu bölümde; araştırma sürecinde süreç, araştırma grubu, verilerin toplanması, veri toplama araçları ve verilerin analizi bölümleri açıklanmıştır.

3. 1. Araştırma Modeli

Bu araştırmada BTKO'da problem çözme süreçlerinin betimlenerek elde edilen veriler ile sürecin temel aşamaları ortaya çıkarılmaya çalışmaktadır. Bu çerçevede araştırma betimleyici araştırmalar çerçevesinde ele almak mümkündür. Betimleyici araştırmalar, araştırılan durumların betimlemesini, tasvirini ortaya koyar (Lin, 1976). Bu araştırmalarda çalışılan olgu ya da süreç ile ilgili örneklem hakkında elde edilen veriler betimlenerek temel özellikleri tasvir edilir. Daha özel olarak, bu araştırma betimleme ile açıklamayı esas aldığı düşünüldüğünde, çalışmayı açıklayıcı/tanımlayıcı durum çalışması çerçevesinde değerlendirmek mümkündür. Açıklayıcı/tanımlayıcı durum çalışmalarında az sayıda özel durum için derinlemesine inceleme gerekmektedir ve özelde çalışma için seçilen durumlar anlaşılmaya çalışılan olayları en iyi şekilde temsil etmelidir (Davey, 1991). Bu çalışmada Code.org üzerine yoğunlaşılarak BTKO'larda gerçekleşen süreç derinlemesine incelenmeye çalışılmıştır, bu yönüyle açıklayıcı/tanımlayıcı durum çalışmasıyla örtüşmektedir. Nitel verilerin yorumlanarak sonuca ulaşıldığı bu çalışmada araştırma verileri; nitel veri toplama araçları olan gözlem, mülakat ve ekran kayıtlarıyla toplanmıştır. Nitel araştırmalarda genellikle amaca uygun şekilde seçilmiş küçük örneklemlerle çalışılmaktadır (Patton, 2014). Bu araştırma da öğrencilerin yaşadığı süreci derinlemesine incelemek amaçlandığı için küçük bir gruptaki öğrenci davranışlarını derinlemesine inceleyen bir süreç tasarlanmıştır. Patton (2014)'a göre nitel araştırmalarda örneklemin büyüklüğünden çok araştırmaya yönelik durumları içermesi önemlidir ve bu aynı zamanda araştırmacının gözlem yeteneği ile yakından ilişkilidir. Bu açıdan nitel araştırmalarda araştırmacının da sürece dahil olması elde edilenlerin geçerliliği üzerine olumlu bir etkiye sahiptir. Bütün bunlardan yola çıkarak nitel bir şekilde tasarlanan bu araştırma amacına yönelik uygulanan süreç Şekil 6'da özetlenmiştir.

Şekil 6. Araştırma sürecinin gösterimi

3. 2. Araştırma Grubu / Evren ve Örneklem / Denek - Denekler

Bu çalışma 2016-2017 eğitim öğretim yılında Rize ilinde bir ortaokulda 6.sınıfa devam eden, Bilişim Teknolojileri ve Yazılım dersini alan 15 öğrenci (11 kız, 4 erkek) ile gerçekleştirilmiştir. Öğrenciler ilk defa bu çalışma kapsamında programlama eğitimi almışlardır. Genel olarak 5. Sınıfta Bilişim Teknolojileri ve Yazılım dersinin kazanımlarında temel bilgisayar kullanımına yönelik becerileri edindikleri için öğrencilerin bilgisayar kullanım seviyeleri birbirine yakın ve çalışmayı yürütebilecek düzeyde olduğu varsayılmıştır. Öğrenciler daha önceki yıllarda matematik dersi kazanımları çerçevesinde karmaşık olmayan problem çözümlerine ilişkin deneyimlere sahiptir.

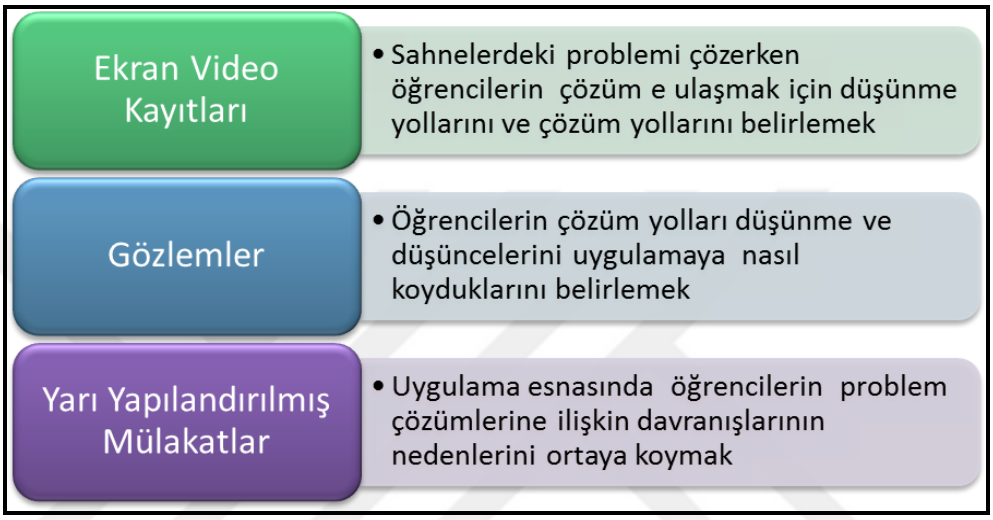
3. 3. Verilerin Toplanması

Nitel çalışmalarda esas olan insan davranışlarının en iyi şekilde gözlemlenerek yorumlanması, süreç boyunca farklı veri toplama araçları kullanılarak elde edilen verilerin birçok kanaldan doğrulanabilmesi özel durum çalışmalarında tercih edilen bir uygulama şeklidir. Bu çerçevede veriler uygulama süreci boyunca ekran kayıtları, gözlemler ve bunlara ilişkin mülakatlar ile toplanmıştır. Aşağıda bu araçlar tanıtılmaktadır.

3. 3. 1. Veri Toplama Araçları/Teknikleri

Verilerin toplanması hem süreç esnasında hem de süreç sonunda gerçekleştirilmiştir. Araştırmacının aynı zamanda uygulayıcı olması süreçte verilerin yakından elde edilmesi bakımından etkili olmuştur.

Çalışma esnasında kullanılan veri toplama araçları ve amaçları Şekil 7'de özetlenmiştir.



Şekil 7. Veri toplama araçları

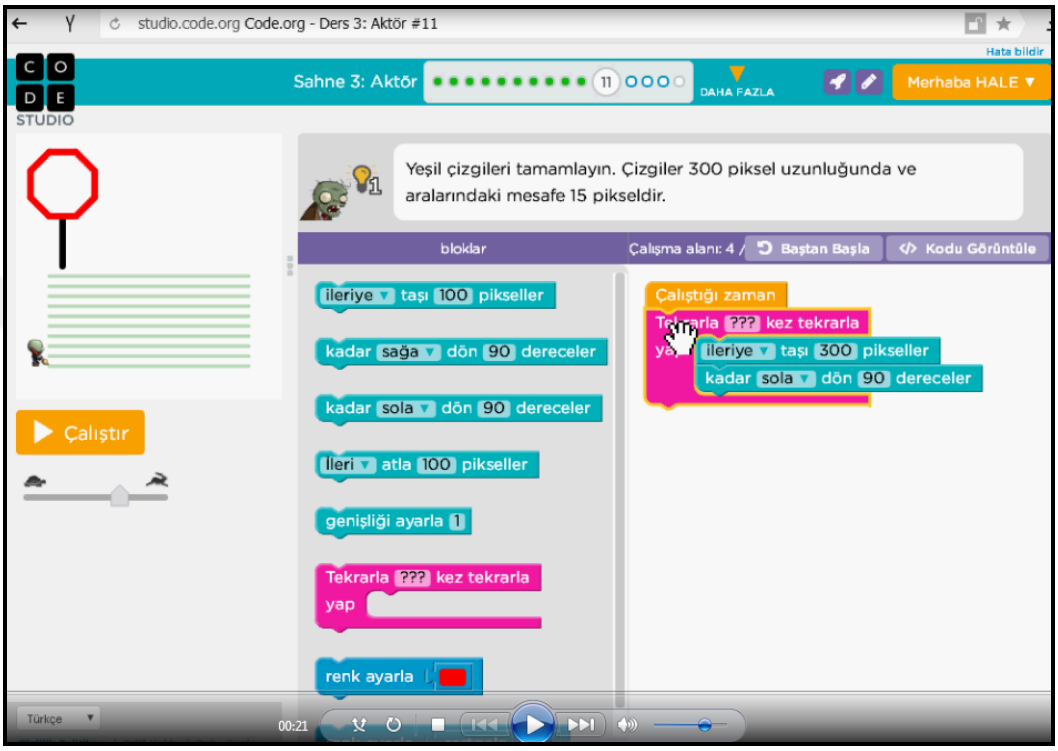
3. 3. 1. 1. Ekran Video Kaydı

Öğrencilerin süreç boyunca uyguladığı işlem adımlarını takip edebilmek için ücretsiz bir uygulama olan HyperCam yazılımı ile uygulama esnasında ekranlarını kayıt altına almaları sağlanmıştır. Çalışma hiç başlamadan yazılımın ara yüzü ve kısa yol tuşları öğrencilere tanıtılmış uygulama esnasında ekran kaydı almalarında yaşanabilecek olumsuzluklar en aza indirilmeye çalışılmıştır.

Derslerin işlenişinde belirlenen sahnelerin bazılarını öğrenciler kendileri çalışırken, bazı sahnelerde ekran kaydı alınmıştır. Ekran kaydı alınan sahneler ilgili dersteki genel programlama yapılarını en iyi temsil edebileceği düşünülen bölümlerdir. Kayıt alınan bölüme gelen öğrencilerin HyperCam ekran kayıt programını uygulamaya başlamadan önce mutlaka çalıştırmaları sağlanmıştır. Bölümü başarıyla tamamlayan öğrenciler bir sonraki kayıtlı bölüme kadar kendi bireysel hızında ilerlemek üzere serbest bırakılmıştır. Her kayıtlı bölümde öğrenciler sınıf içinde gözlemlenmiş ve eş zamanlı olarak öğretmen

ara yüzü üzerinden Code.org ortamında öğrencilerin bölümlerdeki ilerlemeleri takip edilmiştir.

Bir bölüme ait ekran video kaydı tamamlandıktan sonra kayıt öğrencilere izletilerek süreçteki hareketlerinin gerekçeleri araştırmacı tarafından sorulmuştur. Program ile kayıt alınan ekranlara dair örnek bir görüntü Şekil 8'de gösterilmiştir.



Şekil 8. Ekran videosu kayıt örnek görüntüsü

3. 3. 1. 2. Gözlemler

Araştırmacının sınıf içi uygulayıcı olduğu bu çalışmada süreç boyunca araştırmacı gözlemler yapmıştır. Gözlemler ders boyunca tek tek öğrenciler dolaşılarak düşünme şekilleri ve sahnelerdeki uygulamaları ile ilgili yapılandırılmamış notlar alınarak gerçekleştirilmiştir. Gözlem esnasında araştırmacı, öğrenci davranışlarına müdahale etmeden ortaya çıkan durumları incelemeye çalışmıştır. Sınıf ortamında tek uygulayıcı tarafından gözlem gerçekleştirilmesi bir sınırlılık oluştursa da öğrencilerin hepsinin aynı anda aynı sahnede olamama durumu bu sınırlılığı en aza indirmiştir. Her öğrencinin kendi hızında ilerlediği düşünüldüğünde seçili 10 farklı sahne için kayıt alacak öğrencilerin eş zamanlı olarak o sahnelerde bulunması ve öğrenci gözlemi gerektiren durum olasılığı

oldukça azalmaktadır. Gözlem ile elde edilen bulgular ekran kayıtlarında problem çözümü sırasındaki davranışlar ile birlikte ele alınarak analiz edilmiştir.

3. 3. 1. 3. Yarı Yapılandırılmış Mülakat

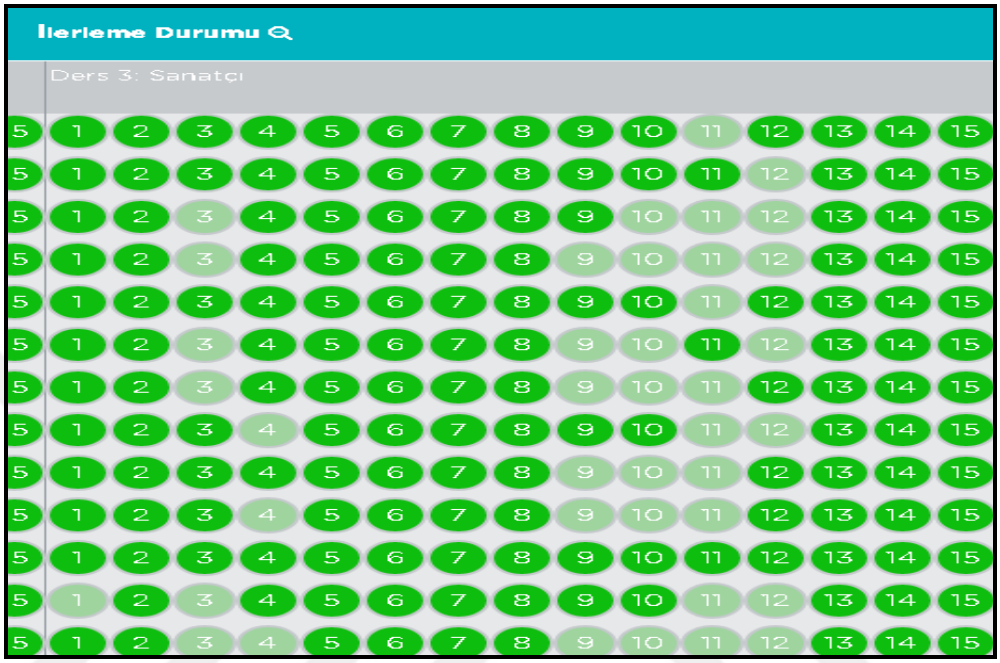
Yarı yapılandırılmış mülakatlar her sahnenin tamamlanmasının ardından öğrencilerin tümüyle birebir görüşmeler şeklinde gerçekleştirilmiştir. Mülakatlarda sürece yönelik sorulan temel sorular BTKO'da eğitim veren 2 uzman görüşüne sunularak bu sorulara son şekli verilmiştir. Mülakatlarda o dersteki süreç boyunca kaydedilen etkinlikleri içeren ekran kayıtları öğrenciye izletilmiş ve yaptığı hamlelerin gerekçesi ve nedenleriyle ilgili sorularak nasıl bir düşünme biçimiyle hareket ettiği tespit edilmeye çalışılmıştır. Mülakat soruları öncelikle BTKO süreçleri ve programlama yapılarına yönelik davranışları inceleyecek şekilde uzman görüşüne başvurularak hazırlanmıştır. Ekran kayıtları üzerinden gerçekleştirilen mülakatlar yaklaşık 10-15 dk sürmüş olup ses kaydedici ile kaydedilmiştir. Bu mülakatlar esnasında öğrencinin ekran üzerindeki hareketlerine yönelik "Bu sahnedeki problem durumu nedir?", "Bu sahne senden ne yapmanı istiyor?", "Genelde bir sahneyi ilk gördüğünde ne yaparsın?" gibi sorulara yer verilmiş olup, öğrencinin verdiği cevaba ve sahnedeki hamlelerine göre mülakat esnasında yeni sorulara yer verilmiştir. Sorulan soruları içeren form Ek-1'de sunulmaktadır.

3. 3. 2. Veri Toplama Süreci / Uygulama Akışı

Çalışmada Code.org BTKO'da belirlenen sahneler üzerinden haftada 4 saat olmak üzere toplam 10 hafta olacak şekilde bilişim teknolojileri laboratuvarında yürütülmüştür. Sınıf içerisinde yürütülen etkinlikler Code.org ortamında Ders-3 kapsamında olacak şekilde belirlendikten sonra kayıt alınacak sahnelerin öğrenciler tarafından kolayca takip edilebilmesi için öğrencilere ders takip çizelgesi dağıtılmıştır. Ders takip çizelgesi ortamdaki hangi sahnelerin çalışılacağını göstermekte olup Ek-2'de sunulmaktadır. Sahnelerin seçiminde, sahnelerin farklı programlama yapılarını içeriyor olmaları, bu yapıların kombinasyonlarını içeriyor olmaları, birbirine doğrudan benzer olmaları, öğrenciler için matematik vb. gibi derslere ilişkin mevcut bilgilerinden fazla bilgi gerektirmemeleri gibi durumlar BTKO öğretiminde bir uzman yardımıyla değerlendirilerek dikkate alınmıştır.

Öğrencilerin takibi ve kontrolü için Code.org'da bulunan Öğretmen Ara yüzü kullanılarak öncelikle sınıf oluşturulmuş ve öğrenciler sisteme tanımlanmıştır. Sürecin yönetilmesi bakımından böyle bir ara yüze sahip ortamın oluşu araştırmanın amacıyla

uyumlu görülmektedir. Öğrenciler tarafından sahnelerin tamamlanma şekli ve toplam yazılan kod blok sayısı gibi detayları saklayan öğretmen ara yüzü sağladığı veriler bakımından sürecin yürütülmesinde kolaylık sağlamıştır. Öğretmen ara yüzüne ilişkin ekran görüntüsü Şekil 9'da sunulmaktadır.



Şekil 9. Code.org öğretmen ara yüzü

Şekil 6'da özetlenen süreç adım adım uygulanmış ve çalışma tamamlanmıştır. Süreç boyunca öğrencilerin sınıf dışı ilerlemelerini kontrol edememenin bir sınırlılık doğuracağı öngörülmüş ve öğrencilere sınıf dışında ilerlememeleri gerektiği özellikle vurgulanmış, okul dışında Öğretmen ara yüzü kullanılarak öğrencilerin ilerleyip ilerlemediği sürekli araştırmacı tarafından kontrol edilmiştir. Böylece hedeflenen çalışma kapsamında tüm seçili sahnelerin sınıf ortamında uygulanması sağlanmıştır.

Süreçte seçili bölümlere gelen öğrenciler uygulamaya başlamadan önce ekran kaydına başlamıştır.

3. 3. 3. Araştırmanın Geçerliliği ve Güvenilirliği

Araştırma problemleri çerçevesinde nitel verilerin yorumlandığı bu çalışmada geçerliliği ve güvenilirliği sağlamada izlenen yol, verilerin toplanması, analiz ve sonuç çıkarılma süreçlerinin anlamlı olması için çalışmaya ilişkin detaylı bir tanımlama sunulmalıdır (Merriam, 1998). Bu çerçevede yapılanlar veya alınan önlemler bu bölümde

temel başlıklar halinde sunulmaktadır. Veri toplama araçlarının geçerliliğinde, mülakat sorularının etkinliklerde sergilenen davranışları kapsaması, açık ve anlaşılır olarak soruların sorulmuş olması göz önünde bulundurulmuştur (McMillan ve Schumacher, 2010).

İnandırıcılık: Araştırmada toplanan verilerin gözlem, ekran kaydı ve mülakat gibi farklı veri toplama kaynaklarından çeşitleme yapılarak elde edilmiş olmasıyla farklı kaynakların birbirini doğrulaması bağlamında inandırıcılığı arttırıcı niteliktedir.

Uzun süreli uygulama: Araştırma 10 hafta boyunca haftada 4 saat olacak şekilde yürütülmüş, bu sayede öğrenci davranışlarının anlık veya sosyal etkileşime bağlı gerçek olmayan davranışlar olmasının önüne geçilmiştir. Ayrıca araştırmacının seçilen belirli sahnelerdeki etkinliklerin hemen ardından mülakatları gerçekleştirmiş olması da öğrenci davranışlarının doğru ve detaylı betimlenebilmesini mümkün kılmıştır.

Uzmandan yararlanma: Uygulamada yürütülecek sahnelerin belirlenmesi, veri toplama şeklinin belirlenmesi ve mülakatlardaki temel soruların belirlenmesi sırasında BTKO'da problem çözme öğretiminde uzman olan öğretim üyelerinden yararlanılmıştır. Bu yönüyle araştırmanın sağlıklı veriler elde edecek biçimde yürütülmesine katkı sağladığı düşünülmektedir.

Aktarılabilirlik: Araştırma probleminin belirlenmesinden verilerin çözümlenerek sonuçların çıkarılmasına kadar bütün boyutlarıyla ayrıntılı biçimde sunulmuştur. Araştırmanın kuramsal çerçevesinin detaylı bir şekilde anlatılması, çalışma grubunun seçilmesi ve özelliklerinin ortaya konulması, uygulama sürecindeki etkinliklerin açıklanması, veri analizlerinin kod ve temalardan yola çıkılarak temel aşamalara nasıl ulaşıldığının belirtilmesi araştırma bulgularının aktarılabilir olduğuna işaret etmektedir.

Teyit edilebilirlik: Veri çeşitlemesi çerçevesinde gerek sözlü, gerek görsel verilerden yararlanılarak verilerin toplanması ve yorumlanması araştırmanın teyit edilebilir olduğunu göstermektedir.

Tutarlılık: Verilerin kayıt edilirken veri kaybına yol açmayacak şekilde ses kayıt cihazıyla kaydedilmesi, transkriptlerin tekrar dinlemelerle kontrol edilmesi, ekran kayıtlarının mekanik olarak kayıt altına alınmış olması tutarlılık noktasındaki önlemler arasında değerlendirilebilir. Yine verilerin analizinin uzman tarafından kontrol edilmiş olması tutarlılık çerçevesinde değerlendirilebilir.

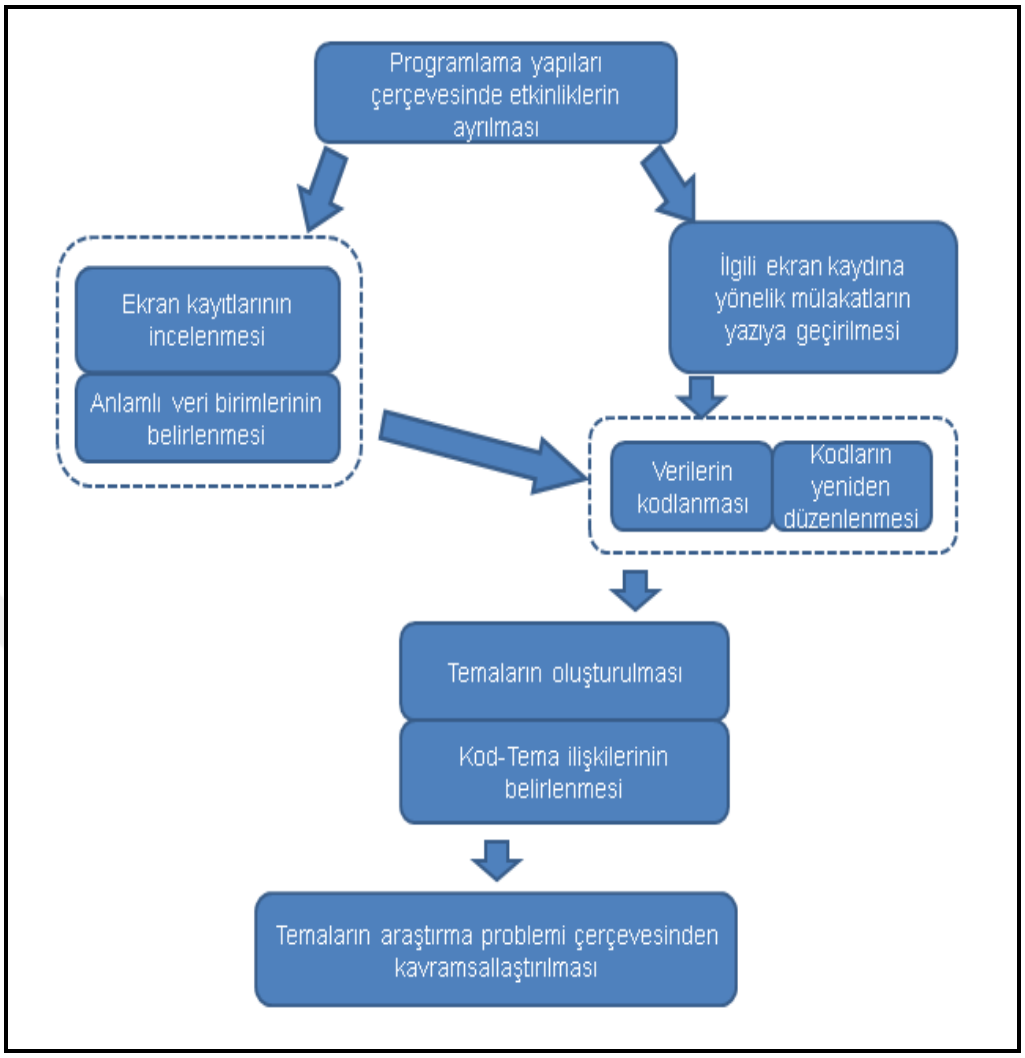
3. 4. Verilerin Analizi

Çalışmada *ekran kayıtları, mülakatlar ve gözlemler* birlikte değerlendirilerek analiz edilmiştir. Bu analiz sırasında merkeze ekran kayıtları alınmış, diğer veri kaynaklarından

gelen veriler doğrulayıcı ve açıklayıcı olarak kullanılmıştır. Ayrıca mülakatlardan yapılan alıntılar ile içerik analiziyle ortaya çıkan tema ve kodlara ilişkin durumlar detaylandırılmıştır (Denzin ve Lincoln, 2000). Nitel olarak toplanan mülakat verileri ilişkin ilişkileri belirlemek için verilerde nitel analiz programı Nvivo11 kullanılmıştır.

İlk olarak, programlama temel yapıları üst çerçeveler olarak değerlendirilip bu yapıları içeren sahneler ayrılmıştır. Bu sahnelerin her birinde öğrencilerin ekran kayıtlarındaki davranışlar amaçlarına göre küçük birimlere ayrılmıştır. Bu davranışlar programlama sürecine ilişkin anlamlı bir yapı ifade ettiğine kanaat getirildiğinde bu veriler kavramsal olarak anlamlı kodlara ayrılmıştır. Elde edilen kodlar mülakat verilerinde ilgili davranışa yönelik ifadeler ile ekran kaydındaki davranış birbirini desteklediğinde davranış kod olarak değerlendirilmeye başlanmıştır. Bazı durumlarda ekran kaydı-mülakat verileri birbirini destekler durumda iken bazı durumlarda bu ilişki birbirini tamamlama biçiminde de gerçekleşebilmiştir. Verilerin analizinde öğrenciler Ö1, Ö2, Ö3…vb. şekilde adlandırılmıştır.

Tüm sahnelerdeki problem çözme süreçleri içerisinde değerlendirilebilecek anlamlı veriler ortaya çıkarılarak kod listesi oluşturulmuştur. Daha sonra oluşturulan kod listesinde birbirine benzeyen kodlar yeniden değerlendirilmiş ve tek bir kod altında birleştirme veya gerekliyse birden çok kod oluşturma yoluna gidilmiştir. Ardından ortak özelliği olan kodlar bir araya getirilerek ana temalar oluşturulmuştur. Bu temalar oluşturulurken, araştırmanın süreç tanımlamaya yönelik olduğu göz önünde bulundurularak problem çözmenin doğası, kodların (ilgili davranışların) gerçekleşme zamanı gibi durumlar da dikkate alınmıştır. Kod ve temalar oluşturulurken uzman görüşüne başvurulmuştur. Son aşamada ise veriler ayrıntılı bir biçimde tanımlanmış, açıklanmış, yorumlanmış ve görsel hale getirilmiştir. Bu doğrultuda içerik analizi süreci Şekil 10'daki gibi özetlenebilir.



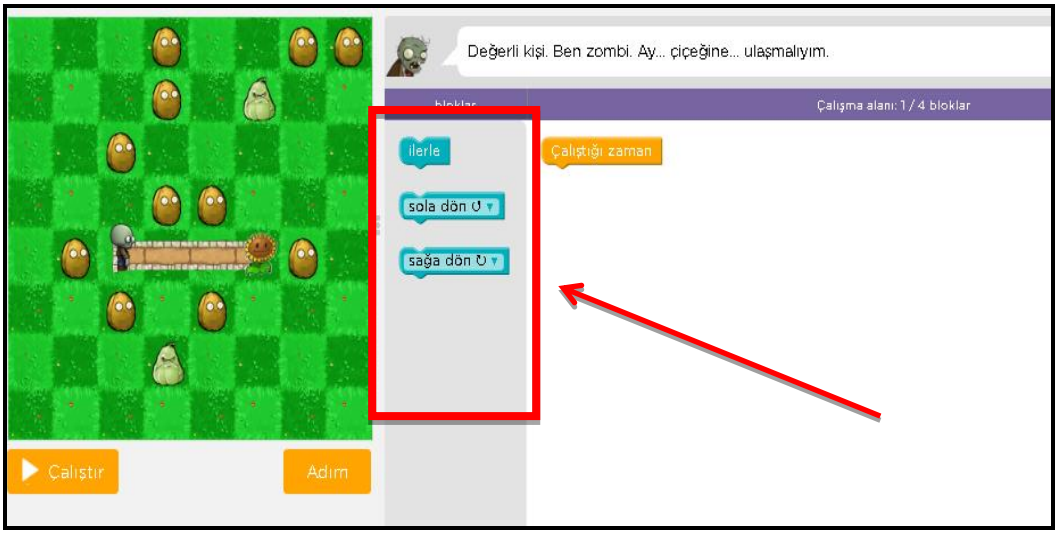
Şekil 10. Farklı kaynaklardan gelen verilerin analiz süreci

4. BULGULAR

Çalışma süresince Code.org sahnelerinde belirlenen problem çözme etkinlikleri sınıfta uygulatılmıştır. Uygulama esnasında öğrencilerin ortam üzerindeki davranışları gözlemlenmiş ve ekranları kayıt altına alınmıştır. Ekranlar üzerinden yapılan mülakatlar ile çalışmadan elde edilen bulgular BTKO'da genel olarak kullanılan programlama yapılarına göre 5 ana bölümde (Temel Taşlar, Döngüler, Koşul Yapıları, Fonksiyonlar, Hata Ayıklama) sunulmuştur. Programlama yapılarına göre oluşturulan bu 5 farklı bölüm birden fazla sahne içermektedir. Farklı sahnelerde farklı programlama yapıları bazen iç içe kullanıldığından bölümler arasında bazı etkinlikler (ilerleme, dönme, nesne alma vb.) zaman zaman tekrar etmektedir. Problem çözümü sırasında öğrencilerin davranışları bu 5 bölümde farklı amaçlarla tekrarlanabilmektedir. Bu nedenle bulguların sunumunda, öncelikle her bölüm için sıklıkla tekrar edilen, daha sonra ilgili bölüme özel gelişen davranışlar ele alınmıştır.

4. 1. Temel Taşlar (Tüm Sahneler)

Code.org veya benzer BTKO'da genellikle ilk sahnelerde çözüme ulaştıracak ana unsur olarak diğer sahnelerde ise ara hareketleri gerçekleştirmek için kullanılan "ilerle, sağa / sola dön" blokları bu çalışmada temel taşlar şeklinde isimlendirilmiştir. Şekil 11 ile sahne içerisinde blokların bulunduğu alan ve temel taşlar vurgulanmıştır.



Şekil 11. Sahnelerde blokların bulunduğu alan

Tüm sahnelerin içerisinde yer alan temel taşlarla ilgili öğrencilerle Sahne1 kayıtları üzerinden yapılan mülakatlarda elde edilen bulgular Tablo 6'da ifade edilme sıklığına göre çoktan aza sıralı bir şekilde gösterilmiştir.

Tablo 6. Temel Taşlara İlişkin Kodlar

Sıkça Tekrar Eden Davranışlar(Temel Taşlar)	Sahne1 (f)
Blokların kodlama bilgisini kavrama	17
Sahnede isteneni ifade etme	15
Çözüme uygun blokların seçimi	12
Problem sahnesini inceleme	10
Hatayı fark etme	7
Düşünme	7
Zorlanınca ipucundan yararlanma	6
Önce ipucunu okuma	6
Yapılar arası benzerlikleri fark etme	5
Blok sayısı ile yeniden deneme	5
Zihinde canlandırma	5

Çalışmada nitel bulgular analiz edilirken ilgili bloğun programlamada hangi yapıya karşılık geldiği ve nasıl kullanıldığına ilişkin bilgi *blokların kodlama bilgisini kavrama* olarak ele alınmıştır. Code.org sahnelerinin her birinden tamamlanması gereken görevler bulunmakta ve hedefe ulaşılabildiğinde bölüm geçilebilmektedir, öğrencilerin bölümlerde tamamlamaları gereken bu görevleri ifade edebilmeleri *sahnede isteneni ifade etme* şeklinde kodlanmıştır. Tablo 6 incelendiğinde; ekran kayıtlarındaki davranışlarına yönelik öğrencilerle yapılan mülakatlardan *blokların kodlama bilgisini kavrama* ve *sahnede isteneni ifade etme* davranışlarını öğrencilerin genellikle çok zorlanmadan gerçekleştirebildikleri görülmüştür. Sağa-sola dön, ilerle bloklarına dair açıklamaya ve soru sormaya ihtiyaç duymayan öğrenciler farklı bir blokla karşılaştığında o bloğa ilişkin kodlama bilgisi olarak zorluk çekmediklerini göstermişlerdir. Örneğin,

> *(Ö6): "…sayısı belli değil çiçeğe kadar gidiyor onu kendi belirliyor çiçeğe kadar gidince orda duruyor çünkü blok diyor ki ayçiçeğine kadar yap diyor yani belli bir sayısı yok onun…"*

ifadesiyle Sahne1'de bulunan Şekil 12 ile gösterilen *kadar tekrarla-yap* bloğunun kodlama bilgisini anladığını açıkça ortaya koymuştur. Nitekim bu blok tekrar eden yapılar arasında tekrar sayısının bilinmediği durumlarda kullanılarak belirlenen şart yerine gelinceye kadar içerisine yazılan diğer blokları döngü içerisine almayı sağlamaktadır.

Şekil 12. Kadar tekrarla-yap bloğunun görüntüsü

Benzer şekilde öğrencilerin tamamı Sahne1 içinde kendilerinden isteneni "…*zombiyi ayçiçeğine götürmemi istiyor şeklinde*" ifade etmiştir. Kendilerinden istenen sorulduğunda bazı öğrenciler

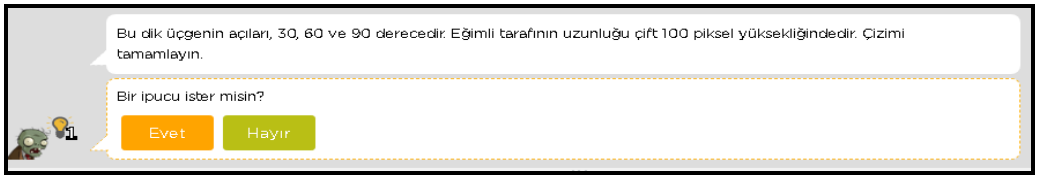
> *(Ö2): "…şimdi bu sahne zombinin çiçeği yemesini istiyor bizden de kodu yazarak zombiyi çiçeğe göndermemizi istiyor…"*

ifadesine benzer biçimde sahnede kendilerinden isteneni daha detaylı bir şekilde ifade etmişlerdir.

BTKO'da kullanıcılardan tamamlanması istenen görevler düşünüldüğünde, tasarlanan hedeflere yönelik alanlarda kullanılan görselin şekli, konumu, metni vb. değişse de alanlardaki temel mantığı değişmemektedir. Code.org arayüzünde de benzer şekilde her bölümde sahneler verilmekte ve kullanıcılardan sahnelere özgü görevleri tamamlamaları beklenmektedir. Bu açıdan öğrencilerin Code.org ile çalışırken *problem sahnesini inceleme* ve o sahnede verilenlere dayanarak *çözüme uygun blokların seçimine* karar verme eylemleri programlamanın diğer yapılarında olduğu gibi temel taşların tanınması ve kullanılması çerçevesinde de gerçekleşmiştir.

Süreç içerisinde araştırmacı gözlemlerinden, seçilen sahnelere yönelik ekran video kayıtlarında öğrencinin fare hareketleri ve alanların üzerinde bekleme sürelerinden öğrenciler *problem sahnesini inceleme* davranışında genellikle iki farklı yol izledikleri anlaşılmaktadır. Öğrencilerin büyük bir çoğunluğunun öncelikle görevlerin sunulduğu alanı incelediği ve ardından blokların seçimi üzerinde düşündüğü görülmüştür. Ö2, Ö3, Ö6, Ö7, Ö8 ve Ö13 yapılan mülakatlarda bir bölümü geçerken nasıl davrandıklarını "…*ilk önce sahneye bakarım daha sonra kod bloklarını incelerim…*" ifadesine benzer ifadelerle ortaya koymuşlardır. Bazı öğrencilerin ise bir bölüme geldiklerinde önce ilgili sahneye özgü kod bloklarının bulunduğu alanı inceledikleri Ö5, Ö10 ve Ö14'ün "*…ilk önce bloklara bakıyorum sonra resimli yere*..." şeklindeki benzer ifadelerinden anlaşılmaktadır.

Code.org bölümlerinde tamamlanması istenen bölüm bu çalışmada problem sahnesi olarak adlandırılmış ve öğrencilerin o alan üzerine odaklanması *problem sahnesini inceleme şeklinde* kodlanmıştır. Sahnelerin üzerinde yer alan açıklama kısımları bazen istenilen görevleri tanımlamakta bazen ise göreve ulaşmaya kolaylaştırıcı bilgiler sağlamaktadır, bu çalışmada yukarıda açıklanan alan ipucu olarak adlandırılmış ve Şekil 13'de gösterilmiştir.



Şekil 13. İpucu alanı

Temel taşların görev ve kullanımlarının bilinmesi ve kullanımına yönelik çoğu zaman başarılı sonuçlar ortaya koyan öğrencilerin bu sahneleri tamamlamak için *problem sahnesini ve kod bloklarını inceleme* davranışlarına ek olarak *düşünme, sahnede*

olabilecekleri zihinde canlandırma ve çözüme başlamadan *önce ipucunu okuma* gibi eylemler içerisinde oldukları da görülmüştür. Birçok öğrenci görevi anlama aşamasında problem sahnesini incelemeden önce ne yapacağını belirlemek için ipucu kısmından yararlandığını belirtmiştir. Bu çerçevede örneğin;

> *(Ö12): " …şurada yazı yazan yerde ayçiçeğine ulaşmam gerektiği yazıyor oradan anladım*"

şeklinde görüş belirtirken, akış yazdıktan sonra olabilecekleri ve karakterin izleyeceği hareketi zihinde canlandırmak konusunu ise

> *(Ö12): "…aklımda kuruyorum böyle benim sağım onun sağıdır diye kendimi aklımdan karakterin yerine koyuyorum öyle yapıyorum"*

ifadesiyle açıklamıştır.

Temel taşları içeren sahnelerde olduğu gibi diğer tüm sahnelerde de Code.org öğrencilere çalışma alanında görevi tamamlayabilmek için kullanması gereken en üst blok sayısını vermektedir. Öğrenci eğer fazla blok kullanarak bölümü tamamlar ve geçerse o bölüme ait numara açık yeşil ile boyanmakta ve öğrenciye fazla blok kullandığına dair uyarı vermektedir. Çalışma alanında sınırlandırılan blok sayısı ile görevler tamamlandığında ise bölüme ait numara koyu yeşil renk ile sistem tarafından işaretlenmektedir. Bu açıdan öğrencilerin davranışları incelendiğinde, fazla blok uyarısı alsa dahi öğrenciler bölümü geçebilmeyi kendine asıl hedef olarak seçtikleri görülmüştür. Örneğin, fazla blok uyarıları karşısında davranışları sorulduğunda

> *(Ö1): "…bölümü bitirince geçiyorum",*

> *(Ö12): "…benim için bölümü geçmek daha önemli…",*

> *(Ö14): "…algoritmam çalışmasa uğraşırdım ama çalıştığı için uyarıdan sona tekrar bakmadım*"

şeklinde görüşlerini belirtmişlerdir. Bu durum bu çalışmada *blok sayısı ile yeniden deneme* şeklinde ele alınmış olup, temel taşlar ile ilgili sahnelerde sıklıkla görülmüştür. Diğer yandan temel taşlar ile ilgili olarak *hatayı farketme, zihinde canlandırma* gibi davranışlar da sık olmasa da mülakatlarda ortaya çıkan davranışlar arasında yer almıştır.

4. 2. Döngüler (Sahne 1-2-6-7-8-9)

Code.org ortamında tekrarlayan yapılar (döngüler) için birden fazla programlama yapısına ait blok yapısı kullanılmaktadır. Code.org Ders3 kapsamında döngüler; tekrarla, olana kadar tekrarla, olduğu sürece tekrarla şeklinde 3 farklı yapı ile sahnelerde karşımıza çıkmaktadır. Bu döngü yapılarını içeren sahnelere yönelik yapılan ekran kaydı ve mülakat analizlerinden elde edilen bulgular döngüler başlığı altında toplanmış ve Tablo 7'de sıklık durumlarına göre çoktan aza doğru sıralı bir şekilde kodlar sunulmuştur.

Tablo 7. Döngü Yapılarına İlişkin Kodlar

Sıkça Tekrar Eden Davranışlar (Döngüler)	sahne1 (f)	sahne2 (f)	sahne6 (f)	sahne7 (f)	sahne8 (f)	sahne9 (f)	Toplam (f)
Blokların kodlama bilgisini kavrama	17	12	21	6	24	6	86
Sahnede isteneni ifade etme	15	15	12	10	11	13	76
Çözüme uygun blokların seçimi	12	7	10	13	11	20	73
Problem sahnesini inceleme	10	8	7	9	8	10	52
Akış-kod kurma	4	6	8	7	10	16	51
Rastgele blokları karıştırma	4	8	3	15	5	3	38
Anlamadan çözüme ulaşma	0	4	11	11	6	1	33
BTKO özelliği ile hata tespiti	2	11	5	3	5	6	32
BTKO özelliği ile anlama	0	1	11	15	4	1	32
Ekrandakileri sayma	4	10	0	7	1	10	32
Blokları değiştirme	1	6	8	4	5	4	28
Yapılar arası benzerlikleri fark etme	5	2	5	1	14	0	27
Hatayı fark etme	7	2	3	6	5	3	26
Sonucu tahmin etme	2	3	7	3	5	5	25
Önce ipucunu okuma	6	9	2	2	5	0	24

Tablo 7'nin devamı

Blok sayısı ile yeniden deneme	5	7	4	1	0	5	22
Yeniden düşünme	1	8	2	1	3	6	21
Sonucun sebebi üzerine düşünme	2	1	2	2	5	8	20
Zorlanınca ipucundan yararlanma	6	5	2	5	1	0	19

Tablo 7 incelendiğinde *blokların kodlama bilgisini kavrama, sahnede isteneni ifade etme, çözüme uygun blokların seçimi, problem sahnesini inceleme, akış-kod kurma* davranışları döngü yapıları içeren sahnelerde de öğrenciler tarafından sıklıkla gerçekleştirilmiştir.

Nitel veriler kodlanırken sahnedeki bloklar arasından çözüm için gerekli olan blokları belirleme ile ilgili işlemler *çözüme uygun blokların seçimi*, blokları seçtikten sonra çözüme uygun biçimde sıralayabilmeye yönelik davranışlar ise *akış-kod kurma* şeklinde ele alınmıştır. Bunun yanında çözüme ulaşılamadığı anda öğrencilerin *rastgele blokları karıştırma, anlamadan çözüme ulaşma, blokları değiştirme* gibi davranışlara sıklıkla başvurdukları görülmüştür. Döngüler ile ilgili sahnelere yönelik mülakatlar ve ekran kayıtlarından elde edilen bulguları içeren Tablo 7'ye göre öğrencilerin tekrarlayan programlama yapılarını bulunduran sahnelerde *ekrandakileri sayma* gibi farklı bir stratejiye yönelmesi dikkat çekicidir. Öğrencilerin özellikle tekrar eden durumları tespit edebilmek ve ekrandaki hamlelere yönelik akışı oluşturabilmek için sıklıkla sahnede bulunan öğeleri sayma yoluna gittiği görülmüştür. *Ekrandakileri sayma* davranışı için Sahne2'de gerekli olan döngüyü kurabilmek noktasında

(Ö5): "…hemen şu çizgileri saydım ve 10 kere tekrarlayacağımızı gördüm" ,

(Ö9): "…öğretmenim ben ilk çizgileri saydım 10 kere tekrarlamam gerektiğini anladım sonra tekrarla 10 kez tekrarlayı koyup içine taşları koyuyorum"

şeklinde görüşlerini ifade etmektedir. Sahne7 içerisinde bulunan tekrarlayan yapıdaki tekrar sayısını tespit etme hakkında

(Ö1): "…sahnedeki üçgenleri sayarak oraya yazıyorum"

şeklinde ekrandaki ilgili nesneleri sayma davranışını sergilediğini ifade etmiştir.

Code.org sahnelerinde öğrencilere sahneleri kavramalarında yardımcı olabilecek bazı ipuçları verilmektedir. Tablo7'ye göre öğrencilerin bir kısmı döngü içeren sahneleri incelemeye başlamadan *önce* sunulan *ipucunu okuduğunu* ardından *problem sahnesini incelediğini* ifade etmiştir. *Önce ipucunu okumak* konusunda

> *(Ö5):* "*…önce okurum çünkü orayı okuyunca daha hızlı yapacağımı düşünüyorum, mesela yeşil çizgileri tamamlayın çizgiler 300 piksel uzunluğunda ve aralarındaki mesafe 15 pikseldir diyor okumazsam 300 pikseli bulamazdım*"

şeklindeki ifadesi dikkat çekicidir. Ö5 ve onun gibi düşünen diğer öğrenciler sahnelere ilk baktıkları anda BTKO'nın onlara sağladığı ipucu alanından yararlanmışlardır. Buna karşın bazı öğrenciler ise sahnede istenilen hedefe ulaşamadığı zamanlarda, çözüm için *zorlanınca ipucundan* yararlandığını *ifade etmiştir.* Bu duruma örnek olarak Sahne1'e yönelik yapılan mülakat esnasında

> *(Ö6): "zor geliyorsa yukarıyı okurum",*

İfadesiyle zorlandığı durumlarda bilgilere baktığını açıklamıştır. Benzer şekilde

> *(Ö8): "…çok zorlandığımda okuyorum"*

ifadesi öğrencilerin zorlanmadıkları zamanlarda bilgilendirmelere dikkat etmediklerine örnek olarak verilebilir. Diğer bir öğrenci aynı sahne için verilen bilgileri okuma noktasında

> *(Ö12): "…aklıma bir şey gelir ondan dolayı okumam o doğru olunca hiç okumadan geçerim doğru olmazsa okurum orayı*"

ifadesiyle zorlandığı durumlarda ipucu bölümünden yararlandığını açıklamıştır. Genel olarak ipucuna doğrudan bakmadığını ifade eden

> *(Ö9): "…ilk başlarda da okuduğum oluyor mesela bir olay yerine bakınca anlamadığım da orayı okuyorum*"

ifadesiyle anlayamadığı zamanlarda *önce ipucu okuyarak* sahneyi anlamaya çalıştığını vurgulamıştır.

Bunlara ek olarak tekrarlama mantığının blok sayısına etkisinin öğrenciler tarafından bu sahnelerde fark edildiği, fazla blok kullanma uyarısı ile öğrencilerin döngüleri bazen yeniden kullanmayı denediği görülmektedir. *Blok sayısı ile yeniden deneme* hakkındaki görüşünü

> *(Ö7) "…fazla kullanınca gereksiz blok oluyor, tekrarla koymam gerektiğini düşünüyorum*"

şeklinde ifade etmiştir.

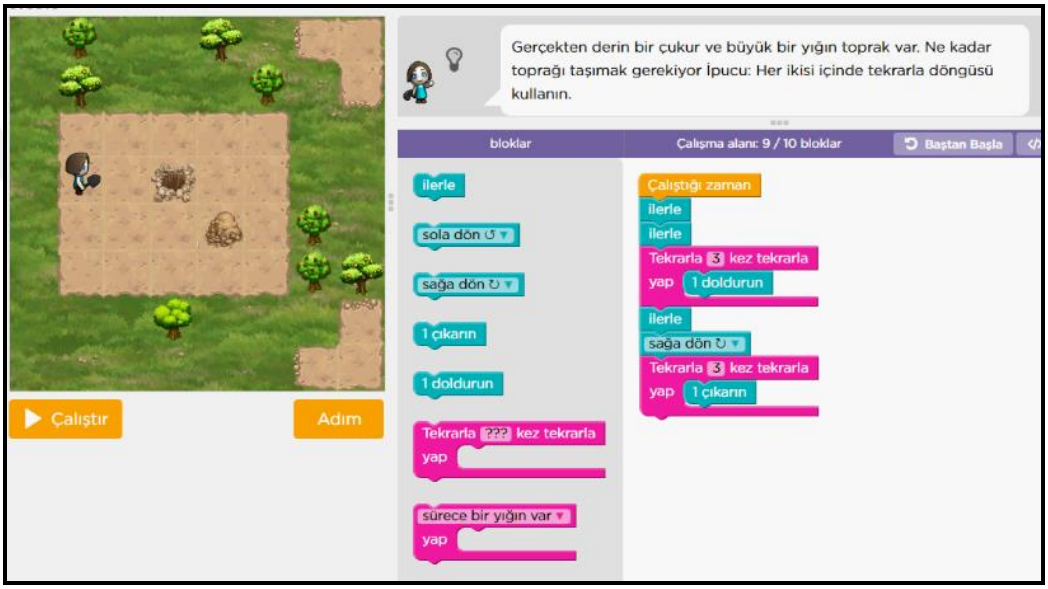
Programlama yapıları düşünüldüğünde öğrencilerin tekrarlayan döngüleri kurarken birbirinden farklı programlama mantığına sahip blokları birbirlerinin yerine kullanmayı düşünebildiği görülmüştür. Öğrencilerin büyük bir çoğunluğu farklı kullanıma sahip döngü bloklarına ait yapılar arası benzerlikleri fark edebilmiş ve aralarındaki ilişkiyi kurabilmişlerdir. Buna benzer durumlar analiz edilirken *Yapılar arası benzerlikleri fark etme* şeklinde ele alınmıştır. Bu noktada Sahne8 içinde bulunan sürece bir yığın/delik var bloğu hakkında Ö4, Şekil 14 ile belirtilen Sahne8'e yönelik mülakatta

> *(Ö4) "…ayçiçeğine kadar ilerleye benziyor, bu blok çukuru doldurana kadar o da ayçiçeğine gidene kadar çalışıyor*"

ifadesini kullanmıştır.

Şekil 14. Sahne8 Ö4'e ait ekran görüntüsü

Burada Ö4, Şekil 14'de bulunan bloklarla daha önceki sahnelerde kullandığı "ayçiçeğine kadar tekrarla" bloğu arasında benzerlik yönünden bir ilişki kurmuştur. Programlama yapıları düşünüldüğünde her iki blokta sayısı bilinmeden tekrar eden durumlar için kullanılabilir. *Yapılar arası benzerlikleri fark etme* çerçevesinde kullanım şekli farklı olan döngü yapılarını da öğrencilerin birbiri yerine düşünebildiği mülakatlar esnasında ortaya çıkmıştır. Örneğin; Şekil 15'de görüldüğü üzere Sahne8 için Ö11 öncelikle tekrarla bloğunu kullanarak çözüme ulaşmaya çalışmıştır.



Şekil 15. Sahne8 Ö11'e ait ekran görüntüsü

"Tekrarla" ile çözüme ulaşamayan öğrenci daha sonrasında o sahne için verilen "sürece bir yığın/delik var" bloklarını kullanmıştır. Ekran kaydı üzerinden gerçekleştirilen mülakat esnasında neden bu şekilde davrandığı sorulduğunda; bu durumun sebebini

> *(Ö11): " …ikisi de aynı işe yarıyor ama birinin sayısını biliyoruz diğer bitene kadar içindeki blokları tekrar ediyor, eğer sayısını bilseydim sürece olan bloğu kullanmazdım "*

şeklinde ifade etmiştir. Ö11 sahnedeki sürece "bir yığın/delik var" bloklarıyla diğer sahnelerde de gördükleri "tekrarla" bloğu arasındaki ilişkiyi kurabildiğini hatta hangi durumlarda bunları birbirinin yerine kullanabileceğini kavradığını bu sayede ortaya koymuştur.

4. 3. Koşul Yapıları (Sahne 5-6-8)

Code.org sahneleri içerisinde ders akışı ve benzer BTK sistemlerinde temel taşlar ve tekrarla yapıları ardından kullanılan “eğer, eğer-değilse, olduğu sürece” blokları bu çalışmada koşul yapıları içerisinde ele alınmıştır. Ekran kayıtları çerçevesinde çalışma içerisinde bulunan Sahne5, Sahne6 ve Sahne8 sahnelerinin sahip oldukları koşul yapılarına ilişkin öğrencilerin davranışları, ekran kayıtları ve mülakatlarda verdiği yanıtlara göre Tablo 8’de bulunan kodların sıklıkla ifade edildiği belirlenmiştir.

Tablo 8. Koşul Yapılarına İlişkin Kodlar

Sıkça Tekrar Eden Davranışlar (Koşul Yapıları)	sahne5(f)	sahne6(f)	sahne8(f)	Toplam (f)
Blokların kodlama bilgisini kullanma (amacına uygun kullanma)	15	21	24	60
Sahnede isteneni anlama	11	12	11	34
Akış için blokların seçimi	11	10	11	32
Akış-kod kurma (oluşturma)	8	8	10	26
Yapılar arası benzerlikleri fark etme	3	5	14	22

Tablo 8’de öğrencilerin koşul yapıları ile ilgili sahnelerde hangi davranışı ne amaçla ne kadar sergilediği ortaya konulmaktadır. Koşul yapıları içeren sahnelerde diğer yapılarda olduğu gibi öğrencilerin oldukça yoğun bir şekilde *“blokların kodlama bilgisini kullanma”* davranışını sergilediklerini ifade etmektedirler. Benzer şekilde *“sahnede isteneni anlama, akış için blokların seçimi, akış-kod kurma ve yapılar arası benzerlikleri farketme”* gibi davranışlar koşul yapılarıyla ilgili sahnelerde de sıklıkla dile getirilmiştir.

Koşul yapıları ile ilgili sahnelerde verilen problemi çözerken öğrencilerin birçoğu koşul içeren blokların *kodlama bilgisini* anlamaları gerektiğini ortaya koymuşlardır. Bununla ilgili olarak

> *(Ö2): “…Mesela şimdi çiçekte bal var biz nektar al yaptık o yüzden çalışmaz eğer nektar varsa nektarı al ya da değilse bal yapı kullanmamız gerekiyor”*

ifadesiyle kullanması gereken blok yapılarını işaret etmektedir. Benzer biçimde

> *(Ö6): “…eğer bloğu ile soru işaretinin altında petek ya da bal varsa çiçekte eğer çiçekte mesela nektar varsa yap değilse bal yap bu bloğu kullanarak hepsinde ne olduğunu tahmin edebiliyoruz”*

ifadeleri ile koşul yapıları ile ilgili blokların problem çözümü için nasıl kullanılması gerektiğine ilişkin kodlama bilgisini kullanmak istediğini göstermektedir. Koşul içeren sahnelerdeki öğrenci ekran kayıtları incelendiğinde bu blokların amacına uygun yerleştirilmesinin de blokların kodlama bilgisi kapsamında öğrenciler tarafından ele alındığı görülmektedir. Bu açıdan

> *(Ö8): "…burada bulut koymuş ya bilinmiyor yani çiçek olduğu bilinmiyor çiçekten sadece nektar alabildiğimiz için eğer çiçek ise nektarı al değilse bal ama önce petekteyse de bal değilse nektar al yapılıyor"*

ile blokların koşula uygun dizilimine atıfta bulunması koşula uygun bir şekilde blok seçimi yapıldığına vurgu yapmaktadır. Benzer şekilde

> *(Ö9): "…ne olduğunu bilmiyoruz orda onun için eğer çiçekte bloğunu seçersek eğer çiçekte yap yanında boşluk var oraya nektarı al koyuyoruz değilse bal yap bloğunu…*"

gibi ifadeleri ile koşula uygun blok seçiminde hangi durumda nasıl davrandığına dikkat çekmektedir.

Öğrencilerin önemli bir kısmı koşul yapıları ile ilgili sahnelerde çalışırken sahnede kendilerinden yapılması isteneni, ulaşılması istenen hedefi anlamaya çalışmışlar ve bunu açıkça ifade edebilmişlerdir. Bu durum genellikle sahnedeki çalışmanın başında görülürken, bazen süreç boyunca da gerçekleşmiştir. *Sahnede isteneni anlama* bakımından sahnenin başlangıcına yönelik

> *(Ö4): "…şu soru işareti olan yerleri ne olduklarını önce anlayıp sonra onları toplayıp başa dönmesini istiyor"*

şeklindeki ifadesi ve buna benzer şekilde süreçteki hareketlerini dikkate alarak

> *(Ö7): "…bir tane arı var kare şeklindeki bir bahçede kenarlarında böyle soru işareti bulutlar var onun altındakini almamızı istiyor"* ,

> *(Ö12): "…o soru işareti yazan nektar veya balsa; çiçekte nektar ya da petekte balsa alıp ilerlememi istiyor",*

(Ö14): "…bulutun altındaki nesnenin ne olduğunu bulmam için bir taş koymuş eğer çiçekse nektarı al değilse bal yap dememi istiyor"

şeklindeki ifadeleri koşul yapıları ile ilgili bu sahnelerde kendilerinden isteneni anlayabildikleri görülmektedir. Koşul yapıları ile ilgili sahnelerinde *kendilerinden isteneni ifade etmede* öğrencilerin büyük bir bölümü *"…zombiyi ayçiçeğine götürmemi istiyor"* şeklindeki ifadeleriyle ortaya koymaktadır. Benzer biçimde Ö12 ve Ö13'ün "…*zombiyi zararlı canavarlara kendini yedirmeden ayçiçeğine götürmemi istiyor*" şeklindeki ifadeleri gibi sahneye bağlı bir şartı vurgulamaktadır. Bazı öğrenciler ise kendinden isteneni belirtirken

(Ö1): "…zombiyi sağa doğru yol varsa dönerek ayçiçeğine ulaştırmamı istiyor"

ifadesinde olduğu gibi blokların kodlama bilgisini içeren koşula dikkat çekmektedir.

Diğer taraftan *akış için blokların seçimi* birçok öğrencinin problem çözüm sürecinde vazgeçilmez bir davranışı olarak karşımıza çıkmaktadır. Blok seçimi koşul yapıları ile ilgili sahnelerde problem çözme sürecinin hemen her anında ortaya çıkmıştır. Bu çerçevede

(Ö1): "…bir çıkarın bloğu bir kere uygulama yapıyor yığının tamamını çıkarmak için sürece bloğunun içine bir çıkarın bloğunu kullanmalıyız."

ifadesinde olduğu gibi blok seçimi hem sahnede isteneni anlamanın bir sonucu olarak karşımıza çıkmakta hem de çözüme ulaşmak için gerçekleştirilecek akışı başlatmaktadır. Örneğin Şekil 16'da görüldüğü gibi Sahne6'daki koşul yapısına ilişkin bloğu seçeceğini

(Ö3): "…sola dön koysak sola döner zararlı yani oyunu bitirecek bitkilere yenilir ayçiçeğine giden yolda hep sağdan gittiği için sağa dön bloğunu kullandım"

Şeklinde izah ederken, bir diğer öğrenci ise Sahne5'te çalışırken

(Ö4): "…burayı petek yaparsak nektar al ile bal yap, şu bal ile nektarın yerini değiştirmemiz gerekiyor"

ifadesiyle seçmesi gereken bloka işaret etmektedir (Şekil 17).

Şekil 16. Sahne6 Ö3'e ait ekran görüntüsü

Şekil 17. Sahne5 Ö4'e ait ekran görüntüsü

Şekil-17'de gösterilen Sahne5'teki problemi çözüme ulaştıracak bloklara yönelik

(Ö4): "…eğer çiçekse nektarı al değilse bal yap bloğunu kullandım"

şeklindeki ifadelerine benzer ifadeler diğer öğrenciler tarafından da dile getirilmiştir ve koşul yapıları için *akış için blokların seçimi* çerçevesinde ele alınmıştır. *Akış-kod kurma* çerçevesinde ise koşul yapılarındaki sahnelerde de öğrencilerin birçoğu koşul yapıları ile ilgili problem çözme için akış oluşturmanın öncesini ve sonrasını ayrı

değerlendirmektedirler. Bu sahnelerde öğrencilerin bir kısmı oluşturacakları akışı uygulama öncesi mutlaka koşula bağlı olarak düşündüklerini ifade etmektedirler. Bu çerçevede

(Ö1): "…önce tek tek düşünüyorum ve adımları diziyorum",

(Ö3): "…önce anlamazsak akışı kurmak zor olur"

ifadeleri sahne üzerinde düşündüklerine örnek olarak gösterilebilir. Diğer taraftan öğrencilerin bir kısmı da çözüme ilişkin blok akışlarını uygulama sürecinde oluşturdukları ortaya konulmuştur. Örneğin, 5 numaralı sahnedeki probleme ilişkin olarak fikrini

(Ö12): "…çözümünü düşünürken sağ sol ilerleyip nektar olmadığını üstten okudum oradan sonra da akışı kurmak için ilerle ve çiçekte nektar al değilse bal yap bloğunu sıraladım",

(Ö14): "…başta farklı kurmuştum ama baktım ki 4 tane bulut var ve 3 defada ona yetişmek için 3 defa ileri gitmemiz gerekiyor sonra da sağa dönmemiz gerekiyordu"

şeklinde ifade etmiştir.

Code.org koşul yapıları sahnelerinde öne çıkan öğrenci davranışlarından birisi de öğrencilerin *yapılar arası benzerlikleri fark etmeleridir*. Bu noktada öğrencilerden bazıları blokların kodlama bilgileri açısından, bazıları çalışma özellikleri açısından birbirlerine benzer veya birbirlerinden farklı yönlerini değerlendirerek kullanmışlardır. Örneğin, Sahne5'te sahnedeki bloğun görevi sorulduğunda

(Ö1): "…eğer okul varsa okula git eğer değilse evde kal gibi…"

şeklinde kodlama bilgisini farklı bir alana transfer ederek ifade etmiştir. Öte yandan

(Ö5): "…tekrarla sadece belirlediğimiz sayı kadar ilerliyor ya da dönüyor ama bu çiçeğe kadar tekrarla da içine koyduğumuz blokların hepsini çiçeğe ulaşana kadar tekrarlıyor"

şeklindeki ifadesi blokları çalışma prensipleri açısından kıyaslamakta olduğunu göstermektedir.

Koşul yapılarını içeren sahnelere yönelik yapılan mülakatlarda daha az sıklıkta da olsa öğrencilerin farklı sahnelerde "*sonucu tahmin etme, BTKO ile hata tespiti, BTKO ile yanlış sıra gözlemleme, zorlanınca ipucuna bakma*" gibi davranışları da sergiledikleri görülmüştür. Süreç içerisinde ise BTKO ile yanlış sıra gözlemlemeye bağlı olarak blokların kodlama bilgisini tam olarak kavrayamayan öğrencilere de rastlanmıştır. Örneğin, aynı blok yapısı için Ö5, Ö9 ve Ö15'e benzer şekilde Sahne6'da koşula bağlı düşünürken

> *(Ö2): "… başladığında ve her adımında beyaz dalgalarla sağa doğru yol var mı diye kontrol ediyor eğer varsa ona göre dönüyor"*

şeklindeki ortak ve doğru ifadelerine karşılık aynı sahne için ekranda gördükleri hakkında

> *(Ö6):* "*…o beyaz şeyler onun sesi canavar bağırıyor orada*"

şeklinde ekran kayıt videosunda izlediği görüntüye yönelik ifadesiyle görüntüdeki beyaz dalgaları ses yaymaya benzeştirerek bir yanılma içerisinde olduğu görülmektedir. Benzer şekilde Sahne5 için

> *(Ö4):* "*ilerle buradan dışarda o yüzden şu karenin dışında olduğu bir ilerleme o eğer oraya gelseydi buraya gene çarpacaktı*"

şeklindeki ifadesi blokların yerinin değiştirilmesi ile sonucu tahmin etme çerçevesinde göstermiş olduğu davranışa bir örnek olarak değerlendirilebilir.

4. 4. Fonksiyonlar (Sahne 3-4)

Fonksiyonlar ile çözüme ulaşılması istenen sahnelerde öğrencilerden öncelikle fonksiyon yapısını kavramaları ve ardından ayrılmış bir alanda yazılan fonksiyonları akış içerisinde adını kullanarak çağırıp çalıştırmaları beklenmektedir. Bu sahneler uygulanırken çekilen ekran kayıtları üzerinden yapılan mülakatlarda öğrencilerden fonksiyon ile yaptıkları çözümlere yönelik elde edilen bulgular Tablo 9'da gösterilmiştir.

Tablo 9. Fonksiyon Yapılarına İlişkin Kodlar

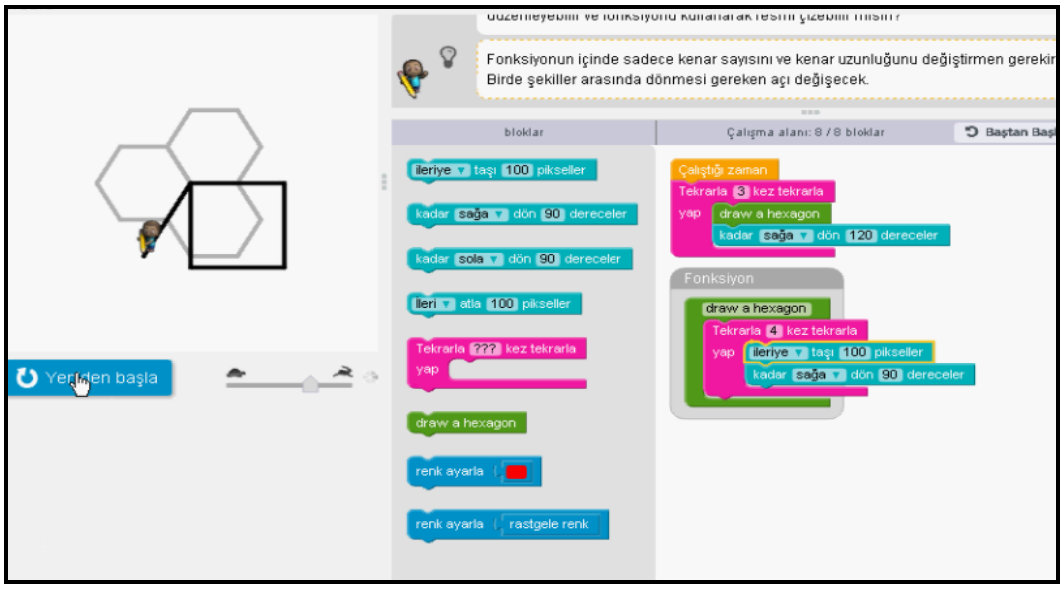
Sıkça Tekrar Eden Davranışlar (Fonksiyonlar)	Sahne3(f)	Sahne4(f)	Toplam (f)
Sahnede isteneni ifade etme	12	12	24
Blokların kodlama bilgisini kavrama	12	7	19
Rastgele blokları karıştırma	9	6	15
Anlamadan çözüme ulaşma	9	4	13
Problem sahnesini inceleme	4	9	13

Tablo9'da yer alan kodlar referans verilme sıklıklarına göre çoktan aza şeklinde sıralanarak sunulmuştur. Fonksiyon durumlarına yönelik öğrencilerin diğer programlama yapılarında da olduğu gibi "*sahnede isteneni ifade etme, blokların kodlama bilgisini kavrama, rastgele blokları karıştırma, anlamadan çözüme ulaşma ve problem sahnesini inceleme*" kodları çerçevesindeki ifadeleri öne çıkan ifadeler arasındadır. Fonksiyonlar ile ilgili öne çıkan davranışlar diğer yapılardaki benzer davranışlara oranla daha düşük sayıda gerçekleşmiştir.

Neredeyse her sahnede olduğu gibi fonksiyonlar ile ilgili sahnelerde (Sahne3, Sahne4) de öğrencilerin çoğunluğu çözüme başlarken *sahnede isteneni ifade ettiklerini* ortaya koymuşlardır. Örneğin Sahne3 için bir öğrenci dışındaki tüm öğrenciler sahnede verilen altıgenleri çizmemi istiyor şeklinde doğru bir ifade ile belirtmişlerdir. Buna karşın öğrencilerin kendilerinden isteneni sadece

> *(Ö10):* "*…bu sahne benden kare çizmemi istiyor yani önce kare çizerek bu şekli oluşturacağım*…"

Şeklindeki ifadesi ile Şekil 18'de gösterilen sahnede isteneni tam kavrayamadığını ifade etmiştir.



Şekil 18. Sahne3, Ö10'a ait ekran görüntüsü

Sahne5 için de benzer şekilde öğrencilerin çoğunluğu kendilerinden isteneni; "*Bu sahne bizden arının nektarları ve balları almasını istiyor*" şeklindeki cümlelerle açıklarken bazı öğrencilerin mülakat ifadeleri sahnede isteneni çözüme yönelik detaylarla birlikte ortaya koymuştur. Örneğin; Ö1, Ö6 ve Ö9'a benzer şekilde

> *(Ö14): "…arının ilerleyip 7 defa nektar alıp bir daha ilerleyip 7 defa bal almasını ve bunu 2 defa tekrarlamamı istiyor"*

şeklindeki ifadeleri sahnedeki göreve ait objelerin miktarına vurgu yaparken, Ö5 ile yakın ifadeler sahip olan

> *(Ö13):* "*Bir fonksiyon kurup çiçek ve baldaki nektarı çiçekteki nektarı baldaki balı yapmamızı istiyor*"

şeklindeki açıklamasıyla çözüme giden yolda izlenecek temel stratejiye işaret etmektedir.

Fonksiyon yapılarında diğer programlama yapılarından farklı olarak ayrı bölümde kurulan algoritmanın ana akışta tekrar çağırılması gerekmektedir. Bu blokların kodlanmasındaki değişikliği ve kullanım şeklini öğrencinin anlaması çözüme ulaşmaları açısından kritik bir noktadır. Bu durum ile ilgili öğrencilerin ifadeleri *blokların kodlama bilgisini anlama* çerçevesinde ele alınmıştır. Fonksiyonun ayrı bir bölümleme ile ekranda oluşturulmasının ve aynı isimle verilen başka bir bloğun olağan akışın içine yerleştirilmesinin fonksiyonun çalışma şekli olduğu öğrencilerin büyük bir bölümü

tarafından fark edilmiştir. Bu doğrultuda Ö1, Ö4, Ö5, Ö8, Ö12 ve Ö14'ün yarı yapılandırılmış görüşmelerde yakın cevaplar verdiği durum için

> *(Araştırmacı): "Bu sahnede bu akışı neden fonksiyon olarak oluşturdun?"*
> *(Ö4):* "*…fonksiyonun bloklar koyuyorsun sonra alıyorsun o fonksiyonun adı neyse şuradan alıyorsun orda yazdıklarının altına koyuyorsun öyle çalışıyor bunun adına fonksiyon deniyor*"

açıklaması ile öğrencilerin fonksiyonu çağırma ve çalıştırma konusunda asgari düzeyde bilgi sahibi olduklarını göstermektedir. Öte yandan

> *(Ö6): "…fonksiyon kullanmasaydık direkt yazsak karışırdı yani fonksiyon ile karışmıyor fonksiyon bize kolaylık sağlıyor*"

fonksiyonlara yönelik ifadesi fonksiyonun kullanılma gerekçesini ve kullanım yapısını tam olarak anlayamamakla beraber kodlama bilgisi olarak bir düzen içinde programlama yapmak için bu yapının kullanıldığını fark ettiği görülmektedir.

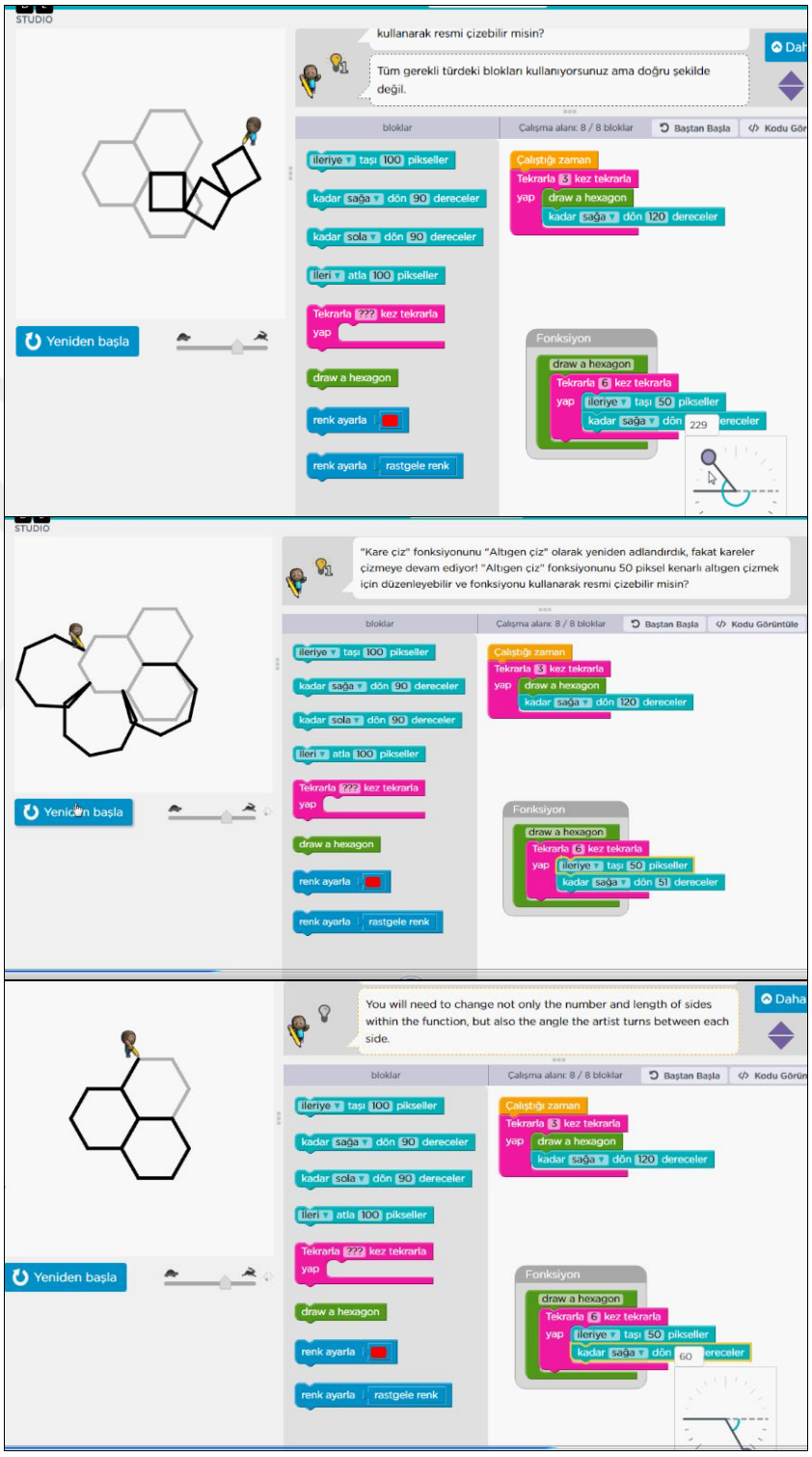
Rastgele blokları karıştırma, *blokların sırasını değiştirme* veya *bloğu başka bir blok ile değiştirme, farklı dizilimleri deneme* gibi davranışlar da çözüm için istenileni gerçekleştirme noktasında sıklıkla kullanılan bir yol olarak karşımıza çıkmaktadır. Örneğin; bu durumu

> *(Ö9):* "…öncelikle *bir şeyler düşünüyorum sonra da olmayınca yeniden deniyorum blokları atıyorum başka blok koyuyorum*"

gibi ifadeleri bloğu başka blok ile değiştirme davranışı çerçevesinde değerlendirilmektedir. Fonksiyonlar yapısında öne çıkan davranışlardan birisi de rastgele blokları karıştırmaktır. Bu doğrultuda, Sahne3 içerisinde bulunan altıgen çizecek fonksiyondaki hatayı düzeltmeye ve altıgen oluşturmaya çalışırken

> *(Ö11): "…çünkü 51 derece yaptığımda şekil büyük oluyor eğer büyük oluyorsa şekli küçültmem lazım yani sayıyı daha büyüteceğim ki o daha küçülecek ben de sayıyı büyütüyorum en yakın şimdi 60 yaptım*"
> *(Araştırmacı): "…peki doğru bulamasaydın nasıl olacaktı?"*
> *(Ö11): "…yine deneyecektim bu sefer mesela denedim denedim olmadı 65, 70, 75, 80 o şekilde denemeye devam edecektim bulana kadar…"*

şeklinde gerçekleşen diyalog öğrencinin şekil içeren bu sahnede aslında transfer etmesi gereken matematik bilgisini kullanmayı hiç düşünmediğini ortaya koymuştur. (Şekil 19)



Şekil 19. Sahne3, Ö11'e ait rastgele blokları karıştıran ekran görüntüleri

Rastgele denemeler sonucu çözüme ulaşmanın öğrenciler tarafından zaman zaman bir çözüm yolu olarak kullanıldığı süreç boyunca gösterdikleri davranışlardan görülmektedir. Benzer şekilde bazen tek deneme sonucunda bile olsa öğrencilerin anlamadan çözüme ulaşabildikleri görülmüştür. Bu durumun diğer programlama yapılarında olduğu gibi fonksiyonlarda da azımsanmayacak biçimde gerçekleştiği görülmektedir. Tesadüfi sonuçlar ile hedeflenen görevin tamamlanması öğrencilerin tekrar o görev ile ilgili olarak düşünmelerini gerektirmemektedir. Anlama noktasında

(Ö12): "… fonksiyon nedir anlamını bilmiyorum ama sahneleri yapıyorum "

ifadesi ile anlamadığı halde fonksiyonu kullanarak ilgili bölümleri tamamlayabildiğini belirtmiştir. Benzer şekilde tesadüfen doğru dizilime denk getirerek anlamadan da bölüm geçilebildiğini Şekil 20'de gösterilen sahneyi çözmeye çalışırken

(Ö13): "…fonksiyona ne yazacağımı bulamadım sonra gri blokları çıkartamadığım için gri renklerde ikinci blokta fonksiyon yazıyordu o yüzden fonksiyonun içine koydum onları öyle çalıştı" ifadesiyle ortaya koymuştur.

Şekil 20. Sahne4, Ö13'e ait ekran görüntüsü

Bu çalışmada görevlerin verildiği bu alanlar problem sahnesi olarak adlandırılan alanda fonksiyonlar ile ilgili uygulama sürecinde öğrencilerin tamamının *problem sahnesini inceleme* davranışını sergiledikleri görülmüştür. Öğrencilerin ekran kayıtları problem sahnesini inceleme davranışının çoğunlukla probleme başlarken gerçekleştiğini, bazen de süreçte beklenen sonuç alınamadığında geri dönüşler yapılarak problem sahnesinin

incelendiği görülmüştür. Problem sahnesini incelerken genellikle öğrenciler çözüm için gerekli olan bilgileri edindiklerini ifade etmişlerdir. Örneğin;

> *(Ö4): "…şimdi burada arının baktığı taraf hep nektar ve bal olduğu için nektarla bal yapmalıyız*"

> *(Ö12): "…oradaki görsellerden neleri alıp almayacağım belli oluyor ne kadar ilerleyip ilerlemeyeceğim sayarak bulabilirim*"

şeklindeki ifadeleri ile çözüme yönelik bilgileri problem sahnesini inceleme şeklinde değerlendirilmiştir. Bazı sahnelerde ise öğrenciler BTKO'ya ait ek özellikleri sahneyi ve ara yüzü inceledikleri esnada kavramaktadır. Örneğin Ö6, Sahne7 içerisinde bulunan gri bloklar için

> *(Ö6): "…onlar orada olması gerekiyor çıkartılamıyorlar olması gerekiyor ki o kodun tamamlanması gerekiyor…"*

ifadesi ile neden sönük renkteki blokların ilgili arayüzün bir parçası olduğunu fark etmiştir. Buna benzer biçimde

> *(Ö6): "…yerlerini değiştirebiliyoruz ama onları atamıyoruz, baktım yani silemiyoruz çözümde kullanmamız lazım*"

şeklindeki ifadesi ile de Şekil 21'de gösterilen sahnede bulunan gri renkli blokların işlevini sahneyi incelerken öğrendiği anlaşılmaktadır.

Şekil 21. Sahne7, Ö6'ya ait ekran görüntüsü

Fonksiyon yapıları kullanılan sahnelere yönelik gerçekleştirilen mülakatlarda Tablo 9'da sunulan davranışlara ek olarak "*hatayı fark etme, blokları değiştirme, sıralamayı değiştirme, önceki sahnelerden yararlanma*" gibi davranışların da öğrenciler tarafından tekrar edildiği görülmüştür. Code.org'da sahneler arasındaki ilerlemeler çoğunlukla basitten karmaşığa gelişen bölümler olacak şekilde düzenlenmiştir. Bu durum öğrencilerin Code.org'un bu özelliğinden faydalanarak zorlandığı bölümlerde çözüme ulaştığı görülmüştür. *Önceki sahnelerden yararlanma* ile ilgili olarak

> (Ö2)*: "…bir seviye atlıyoruz hani altıgen var ya burada 5. Bölüm, 6 ya geldiğimde orada vermişti zaten nasıl altıgenin yapılacağını oraya bakarak bende burada canlandırdım hemen oradan yaptım…*"

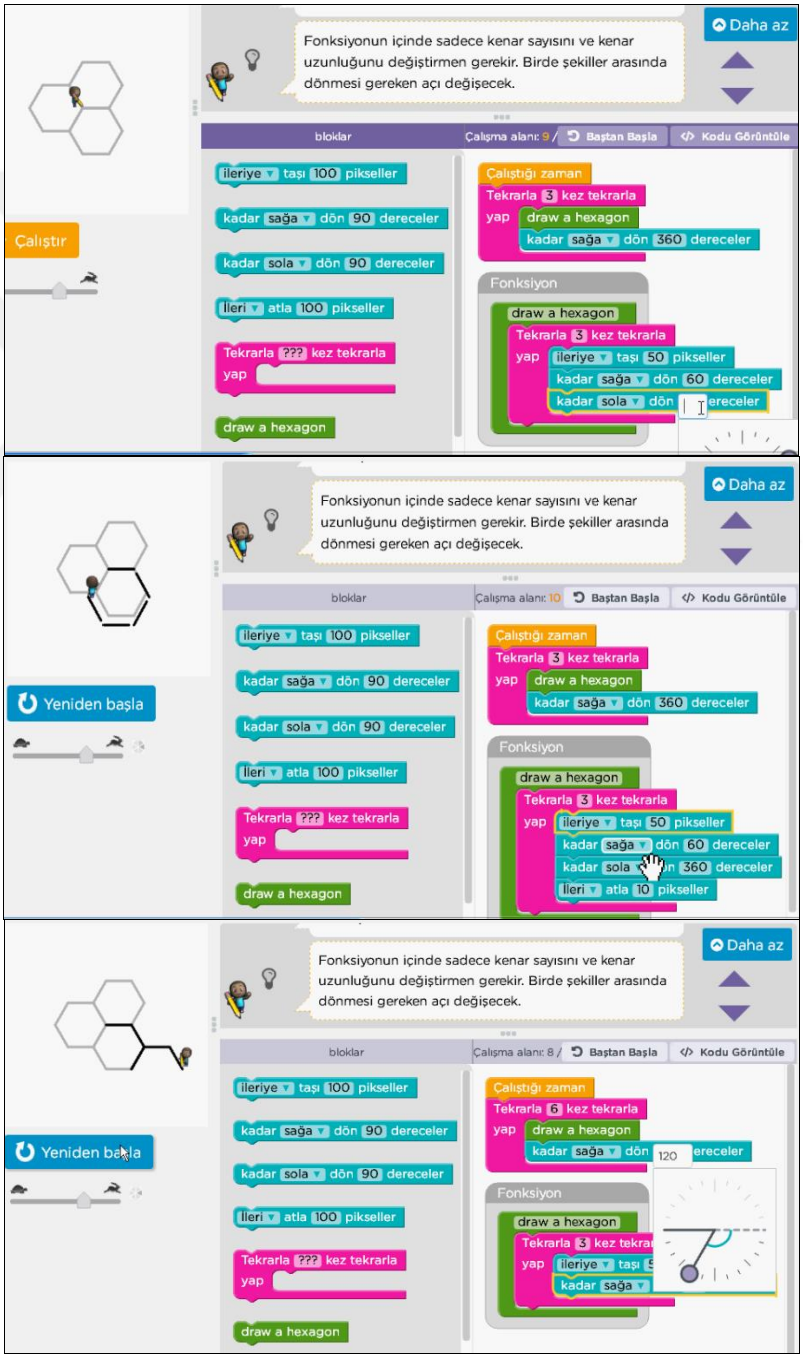
ifadesiyle bir önceki bölümden fonksiyonu kurmak noktasında faydalandığını ifade etmiştir. Benzer şekilde Sahne4 için

> *(Ö13): "…öğrendiğim bilgiyi sonraki sahnelerde kullanıyorum"*

ifadesiyle yalnızca kodlama veya blok yapısı değil bilgi düzeyinde de önceki bölümlerden sonraki bölümlere doğru bir transferin gerçekleştirildiğini ortaya koymuştur. *Blokları veya sıralamayı değiştirme* ile ilgili olarak; sahnede yazdığı kodun çalışmaması durumunda öğrencilerin hatayı fark etmek için çeşitli yollar denediği, fark ettikten sonra da çözüm için blokları veya sıralamayı değiştirme şeklinde bir süreç izlediği görülmüştür. Sahne4 için bu konudaki düşüncesini

(Ö10): "…ilk önce ilerle mesela nektar al onu deniyorum o olmuyor nektarı üste koyuyorum ileriyi alta koyuyorum öyle de olmuyorsa blokları değişiyorum"

şeklinde ifade etmiştir. Bu şekilde öğrencilerin önemli bir kısmının yaptıkları hatayı fark ettiklerinde kullandıkları blokları değiştirme veya sahnedeki mevcut blokların sıralamalarını değiştirme yoluna gittikleri görülmektedir. Örneğin, Şekil 22'de Ö12'nin Sahne5 için hata yaptıktan sonra izlediği sürecin bir kısmı gösterilmiştir.



Şekil 22. Sahne5, Ö12'nin izlediği işlem adımları örneği

Şekil 22 süreç içerisinde alınan ekran kaydında öğrencinin sahnede çözüme ulaşmaya çalışırken istenilen görseli oluşturamadığını fark ettikten sonra, ilk olarak fonksiyonda kullandığı açı değerini değiştirerek deneme yaptığı, ardından fonksiyona yeni bir blok eklediği yine istenilen şekle ulaşamayınca bu sefer ana akışta açı değerine müdahale ettiği görülmektedir.

4. 5. Hata Ayıklama (Sahne 3-4)

Verilen hazır bir akış içerisinde çözüme götürmeyen bölümlerin tespit edilmesinin ve gerekli düzeltmelerin yapılmasının kullanıcıdan istendiği bölümler, Code.org ders akışındaki adlandırma şekline benzer olarak bu çalışmada hata ayıklama olarak isimlendirilmiştir. Bu bölüme ilişkin öğrencilerin sıklıkla tekrarladıkları davranışları Tablo 10 'da gösterilmiştir.

Tablo 10. Hata Ayıklama Yapılarına İlişkin Kodlar

Süreçte Sıkça Tekrar Eden Davranışlar (Hata Ayıklama)	Sahne3(f)	Sahne10(f)	Toplam (f)
Blokların kodlama bilgisini kavrama	12	14	26
Sahnede isteneni ifade etme	12	13	25
Anlamadan çözüme ulaşma	9	11	20
BTKO özelliği ile hata tespiti	3	15	18
Rastgele blokları karıştırma	9	8	17
Problem sahnesini inceleme	4	13	17
Sonucun sebebi üzerine düşünme	1	14	15
Blokları değiştirme	6	8	14
BTKO ile yanlış sıra gözlemleme	4	8	12
Sıralamayı değiştirme	5	7	12
Akış kod kurma	3	9	12
BTKO özelliği ile anlama	2	8	10

Hata ayıklama içeren bölümlerde tamamlanacak görevler Ders3 içerisindeki tüm diğer bölümlerden yapısı gereği farklılaşmaktadır. Diğer bölümlerin tamamında öğrencilerden verilen hedefe ulaşmaları için amaçlarına uygun kod bloklarını seçerek bir akış oluşturmaları istenirken, hata ayıklama bölümlerinde mevcut olarak bulunan bir akışın eksik ya da hatalı yanlarını bulmaları ve bunları istenilen amaca ulaşılacak şekilde düzenleyerek kullanmaları beklenmektedir. Diğer Code.org bölümlerinde olduğu gibi

öğrencilerin hata ayıklama bölümlerinde de *blokların kodlama bilgisini kavrama* ve kendilerinden *sahnede isteneni ifade etme* noktasında zorluk çekmedikleri görülmektedir. *Blokların kodlama bilgisini kavrama* hakkında Sahne10'da bulunan bloklarla ilgili

> *(Ö1): "…ilk önce çalışma alanında akışı yazıyoruz fazla blok kullanmamamız için fonksiyon bloğunu dolduruyoruz içine yazıyoruz kodları sonra make 3 honey diyor ya onu çalışma alanında kendi yaptığımız yere ekliyoruz"*
> *Araştırmacı: "…üste eklemezsen o yeşil taşı çalışır mı?"*
> *(Ö1): "o make3honey fonksiyon bloğunu temsil ettiği için çalışmaz"*

şeklinde geçen diyalog öğrencinin blokların programlama yapılarını anladığını açıkça ortaya koymuştur.

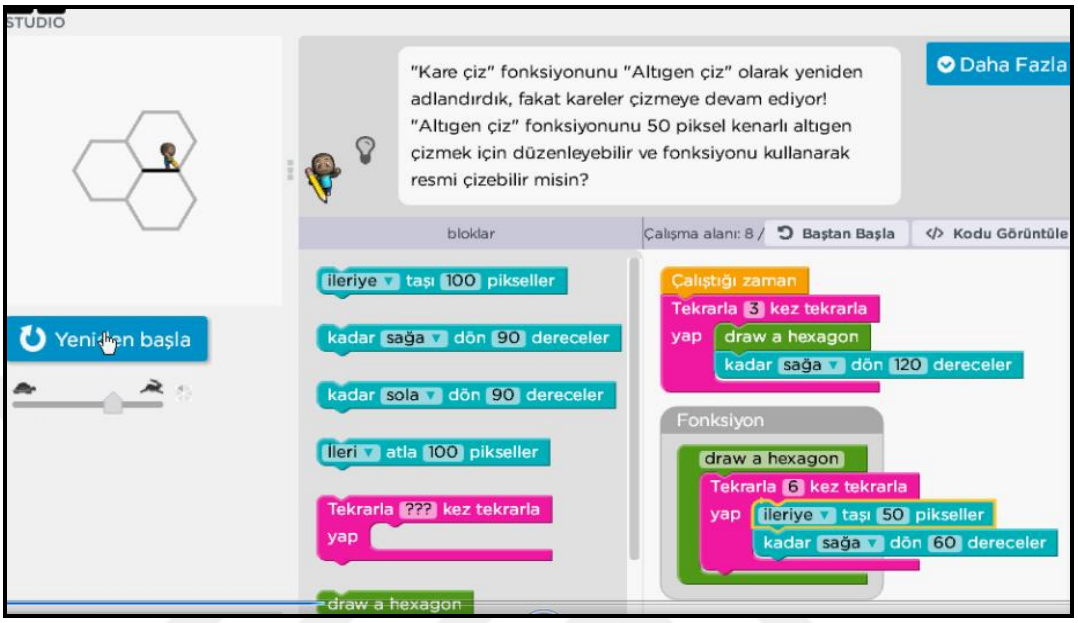
Hata ayıklama sahnelerinde öğrencilerin birçoğunun anlamadan dahi çözüme ulaşabildikleri görülmüştür. Bu konu hakkında

> (Ö2): *"…ilk bizim bloklarımız çalışır sonrada fonksiyon çalışır, bizim bloklarımız bittikten sonra fonksiyona sıra gelir…"*

şeklindeki akışın çalışmasına ilişkin yanlış olan ifadesi ve buna benzer ifadeler hata ayıklama yapısında *anlamadan çözüme ulaşma* başlığında ele alınmıştır. Bu ifade öğrencinin doğru olmayan bir biçimde çalışma alanındaki ana akış bittikten sonra fonksiyonun içindekilere sıranın geldiğini düşündüğü açıkça görülmektedir. Benzer şekilde düşünen diğer öğrencilerin ifadelerine ve sınıf içi uygulamalarına bakıldığında hata ayıklama sahnelerinin akışını takip ederken öğrencilerin BTKO özelliği olan çalışan taşların yanıp sönmesi noktasında yanılgıya düştükleri görülmektedir. Birçok defa öğrencilerin "çalıştır" butonuna tıklandığında sistemin hızlı bir şekilde akışı çalıştırması ve o esnada işlenen blokları sarı renkli bir çerçeve ile vurgulaması öğrencilerin yanlış sıra gözlemlemesine neden olmuştur. *BTKO ile yanlış sıra gözlemleme* noktasında Sahne3 ve Sahne10 çerçevesinde Ö2, Ö3 ve Ö6 öncelikle fonksiyon taşlarının çalıştığını ifade etmişlerdir. Programlama yapısı olarak fonksiyon içeren Sahne3 için

> *(Ö2): "çalıştığı zaman ilk fonksiyon çalışır sonra ileri tekrarla 6 kez yap ileri taşı 50 piksel kadar sağa dön 60 derece, fonksiyon bitince çalıştığı zaman yerine geçer"*

ifadelerini kullanmıştır. Şekil 23 ile öğrencinin yanılgıya düştüğü Sahne3 bünyesinde Code.org'un bir özelliği olan sarı çerçevenin yanlış sıra gözlemlemeye sebebiyet verdiği görülmektedir.



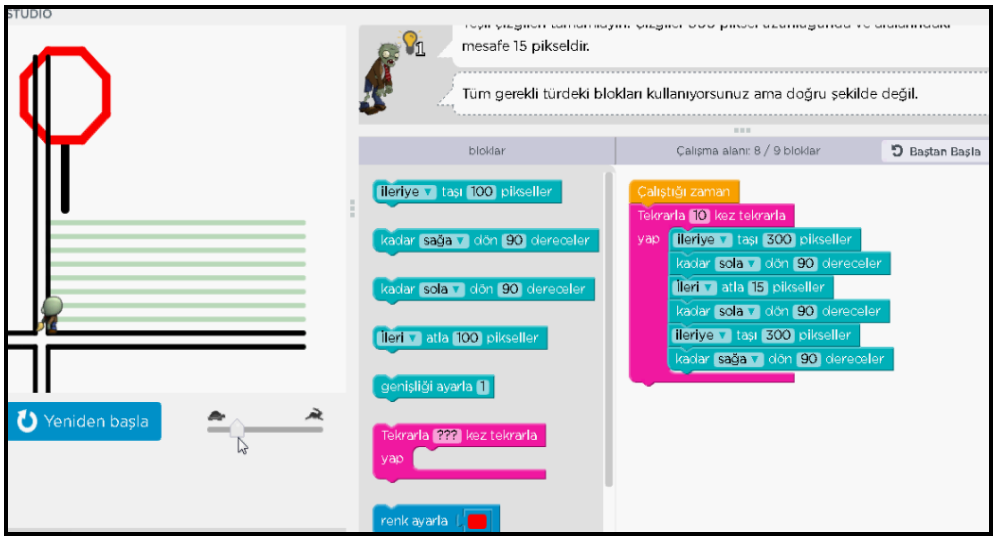
Şekil 23. Sahne3, Ö3'e ait ekran görüntüsü

Code.org'a özgü bir özellik olan kurgulanan *akışı yavaşlatarak test etme* ve oluşturacağı sonucu gözlemleyebilme durumu öğrenciler için hata ayıklama sahnelerinde sıklıkla başvurulan bir durum olduğu belirlenmiştir. Sahne3 içinde bulunan *Kaplumbağa - Tavşan* modellemesi ve Sahne10 içinde bulunan *Adım* mantığı öğrenciler tarafından çözüme ulaşmak için kullanılmıştır. Bu konudaki görüşlerini

> *(Ö5): "… o kaplumbağa yavaş tavşan da hızlı, hızını değiştirip artırıyor yani yanlışımızı anlamaya çalışıyoruz*",

> *(Ö7): "…çalıştırınca o hızlıca ilerliyor ama adıma basınca yazdığım şeyleri adım adım yapıyor, bir ilerliyor sonra bekliyor şurada taşların rengi ışıldıyor oradan anlıyorum hatamı nerede yaptığımı"*

şeklinde ifade etmiştir. Benzer biçimde öğrencilerin ekran kayıtları çerçevesinde Ö5'in Sahne2'de çalışırken yanlış sonuca ulaşması sonucunda ilk hamle olarak akışı yavaşlatmaya yönelmesi örnek olarak Şekil 24'te gösterilmiştir.



Şekil 24. Sahne2, Ö5'e ait ekran görüntüsü

Şekil 24'te de görüldüğü üzere, mevcut bir akış içerisindeki hatayı tespit edebilmek için öğrencilerin sıklıkla sahneyi yavaşlatarak sonucu takip etmeye başvurdukları görülmüştür. Hata ayıklama sahnelerinde de diğer sahnelere benzer olarak öğrencilerin çözüme ulaşamamaları durumlarında *problem sahnesini inceleme, blokları değiştirme, sıralamayı değiştirme, akış-kod kurma, rastgele blokları karıştırma* gibi davranışlara yöneldikleri süreçteki mülakatlardan elde edilmiştir. Örneğin; *sıralamayı değiştirmek* çerçevesinde Ö12 sahnedeki arı karakterini istenen yere ulaştırmaya çalışırken hatasını fark ederek ayıklamış ve bu işlemi ile ilgili olarak,

> *(Ö12): "ilk başta çalıştırdığımda duvara çarpıyordu, baktım onu görünce ilerle ile dön bloklarının yerini değiştirdim"*

şeklindeki ifadeyi ortaya koymuştur.

4. 6. Code.Org BTKO'da Ortaya Çıkan Problem Çözme Davranışlarına Bütüncül Bakış

Bu bölümde mülakatlarda ifade edilme sıklıkları da dikkate alınarak öğrencilerin süreçte gerçekleştirdikleri davranışlar sunulmuştur. Bölümlerde sadece ilgili programlama yapısına yönelik davranışlar açıklanmaya çalışılırken, en sık tekrarlanan davranışlar ise örneklendirilmiştir. BTKO Code.org sisteminde problem çözme sürecine yönelik bir çerçeve belirlemeyi amaçlayan bu çalışmada, süreci modelleyecek bir çerçeveye ulaşmak için tüm yapılardaki davranışları birlikte ele almak gereklidir. Bu şekilde tüm sahnelerde gerçekleşen ortak davranışlar, bu davranışların sıklığı, amacı ve yerinin ortaya

konulmasıyla problem çözümü sürecine ilişkin tanımlamalar mümkün olabilecektir. Bu sebeple, bulguların bu son bölümünde derslerin yürütüldüğü bütün sahneler birlikte dikkate alınarak öğrencilerin davranışları tanımlanmıştır. Davranışlara işaret eden kodlar, sıklık durumlarına göre çoktan aza sıralanmış bir şekilde Tablo 11'de gösterilmiştir.

Tablo 11'de yer alan s1,s2,s3…vb. şekilde ifade edilen sahneler Code.org Ders-3 modülünde bulundukları bölümü içerdiği programlama yapılarını en iyi temsil edeceği düşünüldüğü için seçilen sahnelerdir. Bu çalışmada seçilen sahnelere 1'den 10'a kadar numaralar araştırmacı tarafından verilmiştir. Tablo 11'de yer alan sahneler ardışık bir sırada ilerlememektedir. Örneğin; "s1" ile temsil edilen sahne Code.org Ders-3 modülünde "Sahne2:Labirent/ Bölüm No:13", "s2" ile temsil edilen sahne "Sahne3:Aktör/ Bölüm No:11" şeklinde yer almaktadır. Tüm sahnelerin detaylı bölüm numaraları ve Code.org linkleri Ek-3'te sunulmuştur. Tüm sahneler bulgularda içerdiği programlama yapılarına göre gruplandırılarak 5 farklı bölümde sunulmuştur. Tablo 11'de ise tek tek yer alan sahnelerin içerdikleri programlama yapıları ilk satırda kullanılan; temel taşlar için "Tt", döngüler için "Dö", koşul için "Ko", fonksiyon için "Fo" ve hata ayıklama için "Ha" adlandırması kullanılmıştır.

Tablo 11. Tüm Yapılara Yönelik Davranışları Yansıtan Kodlar

Süreçte Sıkça Tekrar Eden Davranışlar (Bütün Programlama Yapıları)	s1(f)	s2(f)	s3(f)	s4(f)	s5(f)	s6(f)	s7(f)	s8(f)	s9(f)	s10 (f)	Top (f)
İçerdikleri Programlama Yapıları	Tt, Dö	Tt, Dö	Tt, Fo, Ha	Tt, Fo, Ha	Tt, Ko	Tt, Dö, Ko	Tt, Dö	Tt, Dö, Ko	Tt, Dö	Tt	
Blokların kodlama bilgisini kavrama	17	12	12	7	15	21	6	24	6	14	134
Sahnede isteneni ifade etme	15	15	12	12	11	12	10	11	13	13	124
Çözüme uygun blokların seçimi	12	7	4	6	11	10	13	11	20	6	100
Problem sahnesini inceleme	10	8	4	9	6	7	9	8	10	13	84
Akış-kod kurma	4	6	3	6	8	8	7	10	16	9	77
Rastgele blokları karıştırma	4	8	9	6	10	3	15	5	3	8	71
Anlamadan çözüme ulaşma	0	4	9	4	4	11	11	6	1	11	61
BTKO özelliği ile hata tespiti	2	11	3	5	5	5	3	5	6	15	60
BTKO özelliği ile anlama	0	1	2	5	4	11	15	4	1	8	51
Blokları değiştirme	1	6	6	4	2	8	4	5	4	8	48
Sonucun sebebi üzerine düşünme	2	1	1	3	5	2	2	5	8	14	43
Hatayı fark etme	7	2	5	5	2	3	6	5	3	3	41
Yapılar arası benzerlikleri fark etme	5	2	2	2	3	5	1	14	0	6	40
Sıralamayı değiştirme	1	3	5	5	7	3	6	2	1	7	40
Ekrandakileri sayma	4	10	0	1	4	0	7	1	10	3	40

Tablo 11'in devamı

Önce ipucunu okuma	6	9	6	3	3	2	2	5	0	3	39
Duyuşsal alana transfer	1	1	2	1	1	3	5	1	2	21	38
Blok sayısı ile yeniden deneme	5	7	0	5	6	4	1	0	5	5	38
Gündelik yaşamla ilişkilendirme	2	1	1	0	4	2	0	2	2	23	37
Sonucu tahmin etme	2	3	0	1	7	7	3	5	5	4	37
Zorlanınca ipucundan yararlanma	6	5	1	3	6	2	5	1	0	5	34
Yeniden düşünme	1	8	2	1	5	2	1	3	6	4	33
Durumu irdeleme	3	1	1	1	1	3	0	1	10	8	29
BTKO ile yanlış sıra gözlemleme	0	0	4	2	2	0	4	7	0	8	27
Düşünme	7	0	1	2	2	2	2	1	3	6	26
Zihinde canlandırma	5	4	0	2	1	1	2	1	4	4	24
Akışı yavaşlatarak test etme	0	6	4	3	2	0	1	1	0	6	23
Anlamlandırma	1	3	1	1	1	1	2	3	3	6	22
Yanlış çözümle bölüm geçme	1	0	1	1	3	3	3	5	3	2	22
farkındalık kazanma	0	0	0	0	0	0	0	0	0	19	19
Matematik bilgisini kullanma	0	2	7	1	1	0	5	0	1	1	18
Kendini karakterin yerine koyma	4	3	0	0	0	0	0	2	4	1	14
Önceki sahnelerden yararlanma	0	0	3	2	1	0	4	2	0	1	13
Açı göstergesini kullanarak parametre tespiti	0	3	6	1	0	0	2	0	0	0	12
Öğretmenden yardım alma	0	4	1	1	1	0	1	0	0	1	9
Obje ile karakteri canlandırma	0	1	0	0	1	0	0	0	0	1	3

Tablo11'den de görüldüğü üzere *blokların kodlama bilgisini kavrama, sahnede isteneni ifade etme, çözüme uygun blokların seçimi* gibi davranışlar süreç boyunca oldukça sık ortaya çıkmıştır. Ortaya çıkan kodlar değerlendirildiğinde temelde çözüme başlamadan önceki davranışlar, çözüm süreci ve çözümün bitirilmesi sırasındaki davranışlar olarak davranışları tanımlamak mümkündür. Bu davranışlar farklı sıklıkla olsa da bütün sahnelerde görülmektedir. Örneğin; *BTKO ile hata tespiti* Sahne10'da çokken, Sanhe1'de oldukça az ortaya çıkmıştır. Bazı kodların ise bazı sahnelerde hiç yer almayışı dikkat çekici bir durumdur. Örneğin *açı göstergesini kullanarak parametre tespiti* Sahne1, Sahne5, Sahne6, Sahne8, Sahne9, Sahne10 sahnelerinde hiç görülmemektedir. Bu durum çoğunlukla ilgili sahnedeki problemin yapısından kaynaklanmaktadır. Diğer yandan özellikle BTKO özelliklerine ilişkin bazı davranışların ortaya çıkması da dikkat çekicidir. Örneğin; Sahne2, Sahne3, Sahne6, Sahne7, Sahne10 sahnelerinde sadece BTKO ile ilgili özelliklerin az da olsa ifadeleri yansıması BTKO'nun kendi özellikleri bakımından süreçteki bazı yeni davranışlara yol açtığına işaret etmektedir. BTKO'ya has bu davranışlara;

BTKO özelliği ile hata tespiti, önceki sahnelerden yararlanma, akış yavaşlatarak test etme örnek olarak kabul edilebilir.

BTKO'da yürütülen bu süreçlerde çalışmada elde edilen öğrenci davranışları bir bakıma sürece ait işlem adımlarını ortaya koymaktadır. Temelde zamanlama bakımından düşünüldüğünde çözüme başlamadan önceki davranışlar, çözüm süreci ve çözümün bitirilmesi sırasındaki davranışlar birbirinden farklı sahnelerde benzerlik göstermiştir. Örneğin; *sahnede isteneni ifade etme, problem sahnesini inceleme, blokların kodlama bilgisini kavrama, düşünme gibi davranışlar* farklı program yapılarına sahip tüm sahnelerde sergilenmesinin yanında genellikle sürecin giriş aşamasında sıklıkla ortaya çıkmıştır. Bazı davranışların ise süreçler arasında tekrarlanabildiği veya bazen çözüme başlamadan önce ortaya çıkarken bazen çözüm sürecinde ortaya çıktığı görülmektedir. Özellikle bir sahnede hata yapılması durumunda süreci yeniden başlatan öğrenci tarafından ortaya koyulan davranışlar aşamalar arası geçiş özelliği göstermektedir. Örneğin; *akış-kod kurma, blokları değiştirme, sıralamayı değiştirme,* gibi davranışlar öğrenciler tarafından sonuca ulaşılamadığı durumlarda çözüm sürecinde sergilenen davranışlar olarak karşımıza çıkmaktadır. Buna karşın *yeniden düşünme, hatayı fark etme, uygulama ve farkındalık kazanma* gibi davranışların süreçler arası geçiş halinde sergilendiği görülmüştür. *Hedefe ulaşma, anlamlandırma* gibi davranışlar ise daha çok sürecin sonunda karşımıza çıkmaktadır.

Elde edilen kodlar, kodların sıklığı, problem çözme sürecinde ilgili kodun işaret ettiği davranışın ne zaman gerçekleştiği, kodun alt davranışlara bölünüp bölünememe durumları dikkate alınarak bütün sürece ilişkin kodlar gruplanarak, BKTO'da problem sürecini ifade eden temalar "Odaklanma", "Sınama" ve "Sonuçlandırma" olarak isimlendirilmiştir. Tablo11 de ortaya çıkan bazı kodlar birlikte ele alınarak süreçte ait oldukları isimlendirme altında birleştirilmiştir. Örneğin; *sonucu tahmin etme, zihinde canlandırma, obje ile karakteri canlandırma, kendini karakterin yerine koyma* gibi kodlar bir arada değerlendirilerek "sezgi"; *problem sahnesini inceleme, ekrandakileri saymak,* gruplandırılarak "inceleme"; matematik bilgisini kullanma, düşünme, yeniden düşünme gibi kodlar bir arada ele alınarak "düşünme"; *sahnede isteneni ifade etme, önce ipucunu okuma, ekrandakileri sayma* şeklindeki kodlar birleştirilerek "isteneni anlama" gibi ortak kodlarla ifade edilmiştir. *Sezgi, inceleme, düşünme ve isteneni anlama* şeklinde gruplandırılarak kodlarla ilişkilendirilen süreç "Odaklanma" olarak adlandırılmıştır. Diğer yandan *çözüme uygun blokların seçimi, akış_kod kurma* gibi kodlar bir arada ele alınarak sürece ilişkin tanımlamada "akış kurma"; *rastgele blokları karıştırma, sıralamayı değiştirmek, bloksayısı ile yeniden denemek, öğretmenden yardım almak, blokları değiştirmek,* "uğraşı"; *zorlanınca ipucundan yararlanma, akışı yavaşlatarak test etme, açı*

*göstergesini kullanarak parametre tespiti, önceki sahnelerden yararlanma, BTKO özelliği ile hata tespiti, BTKO özelliği ile anlama "*BTKO özelliklerinden faydalanma" davranışı olarak yansımıştır. *Akış kurma, uğraşı ve BTKO özelliklerinden faydalanma* olarak gruplandırılmış hale getirilen kodlar "Sınama" başlığı altında sunulmuştur. Son olarak, *yapılar arası benzerlikleri farketme, gündelik yaşamla ilişkilendirme, durumu irdeleme, sonucun sebebi üzerine düşünme* şeklinde daha çok birbirine benzer durumlar üzerinde yoğunlaşma şeklindeki kodlar "ilişkilendirme"; *duyuşsal alana transfer, anlamlandırma, farkındalık kazanma* gibi davranışlar "yorumlama"; *anlamadan çözüme ulaşma, yanlış çözümle bölüm geçme* kodları ise "yanılsama" olarak adlandırılmıştır. *İlişkilendirme, yorumlama ve yanılsama* olarak gruplandırılarak ifade edilen kodlar "Sonuçlandırma" olarak sürece ait şekilde yerini almıştır. Süreci aşamalandıran şekilde geçişli bir yapıda gösterilen *uygulama, farkındalık kazanma, yeniden düşünme ve hatayı fark etme* davranışlarının ise bu aşamalar arasında ve farklı yönlerde tekrar tekrar gerçekleştiği çalışma sürecinden elde edilen bulgulara dayanılarak ifade edilmektedir.

Özetle *Odaklanma* aşamasının "*sezgi, inceleme, düşünme ve isteneni anlama*" gibi temel davranışları içerdiği belirlenmiştir. Daha çok sürecin yürütüldüğü aşama olan *Sınama* aşamasının "*akış kurma, uğraşı ve BTKO özelliklerinden faydalanma*" ; bununla birlikte "*ilişkilendirme, yorumlama ve yanılsama*" gibi davranışların BTKO'da "Sonuçlandırma" aşamasının temel davranışları olduğu ortaya çıkmıştır. Bu aşamalar tüm sahnelerde yaşanmakta olup Ö8'in Sahne6 ile çalışırken kaydedilen programlama sürecine ilişkin ekran videosundan alınan örnek görüntüler Ek-3'de sunulmuştur.

5. TARTIŞMA

Bu çalışma Code.org ortamında problem çözümüne ilişkin bir çerçeve belirlemeye çalışmaktadır. Bu nedenle elde edilen bulgular, süreçte sergilenen davranışları çerçeveleyen temel problem çözme adımlarının ortaya konulması ve davranışların ortamın sunduğu arayüz ve problem çözmeye ilişkin yapısal özellikler çerçevesinde tartışılmıştır.

5. 1. Code.org'da Problem Çözme Süreci

Code.org üzerinden yürütülen çalışmalar incelendiğinde, genellikle bu çalışmaların öğrencilerin programlama öğrenme süreçlerinin ne şekilde gerçekleştiğinin belirlenmesinden çok öğrencilerin öğrenme performansları veya farklı beceriler üzerine etkisine odaklandığı görülmektedir. Bu tür çalışmalarda genellikle BTKO olarak Code.org kullanımının öğrencilerin motivasyonuna, programlama öğrenmeye karşı tutumuna (Du, 2016), akademik başarıya, yansıtıcı düşünme becerilerinin gelişiminde (Kalelioğlu, 2016) vb. olumlu etkileri olduğuna vurgu yapılmaktadır. Her ne kadar Code.org sahneleriyle sunulan etkinliklerin öğrencilerin bilgi-işlemsel düşünme becerilerinin gelişimine etkisine ilişkin bir takım kanıtlar sunulmuş olsa da, bu gelişim sürecinde öğrencilerin davranışlarının ne şekilde gerçekleştiğine ilişkin oldukça az çalışma söz konusudur. Örneğin Kalelioğlu (2016) çalışmasında Code.org ortamında ilkokul öğrencilerine programlama öğretmek, problem çözme üzerine yansıtıcı düşünme becerilerinde herhangi bir farklılığa neden olmadığını belirlemiştir. Çalışmasında Kızılkaya ve Aşkar (2009)'a ait problem çözme üzerine yansıtıcı düşünme ölçeği kullanılan araştırmada Code.org üzerinde öğrencilerin yaşadığı süreci BTKO'da farklılaşan problem çözme süreci açısından ele almayışı böyle bir değerlendirme yapmasına sebep oluşturmuş olabilir. Aynı araştırmada öğrenciler BTKO'da programlamaya karşı olumlu bir tutum geliştirmiş oldukları şeklinde sonuçlar ortaya çıkmışsa da, öğrencilerin etkinlikler ile ne şekilde etkileşime girdiklerine ilişkin açıklamalara yer verilmemiştir. Benzer biçimde Du (2017) "Hour of Code" etkinliğini kullanarak yapmış olduğu çalışmasında Code.org üzerindeki kodlama saatinin öğrencilerin programlamaya yönelik tutumları üzerinde olumlu bir etkisi olduğunu belirlemiştir. Lye ve Koh (2014) bilgi-işlemsel düşünme üzerine yapılan 27 farklı çalışmayı dikkate alarak yapmış olduğu analiz çalışmasında da birçok BTKO'larını kullanılarak yapılan çalışmaların sürece yönelik bir inceleme ve adımlamadan çok sonuca yönelik etkiler üzerinde odaklanıldığı görülmüştür. Olgun (2014) tez çalışmasında 13 farklı

düşünme stiline yönelik hazırlanan bir ölçek kullanmış ve programlama eğitimi almış öğrencilerin düşünme stillerinde nasıl bir değişiklik olduğuna dair araştırma yapmıştır. Düşünme stillerine yönelik analiz yaparken cinsiyete göre bazı düşünme becerilerinin değişkenlik gösterdiğini ortaya koymuş ve bu çalışmasında BTKO olarak Scratch arayüzünü kullanmıştır. Çetin (2012) çalışmasında çocuklar için bilgisayar programlama eğitiminin öğrencilerin problem çözme becerileri üzerine etkisinin olup olmadığının incelemiştir. Süreçte BTKO olarak Small Basic kullanılmış ve araştırma sonucunda çocuklar için bilgisayar programlama eğitiminin uygulanabilir olduğunu, programlama eğitiminin çocukların problem çözme becerilerine olumlu yönde katkı sağladığını belirlemiştir. Buna ek olarak öğrenci görüşlerine de başvurulan çalışmada öğrencilerin bilgisayar programlama eğitiminden memnun kaldıkları, bilgisayarla neler yapabileceklerine ilişkin düşüncelerinde olumlu yönde değişiklikler olduğu ve bu tarz eğitimlere devam etmek istedikleri sonuçlar arasındadır.

Özetle ulaşılabilen çalışmalar analiz edildiğinde BTKO kullanılarak gerçekleştirilen kodlama eğitimlerini içeren süreci tanımlamaya yönelik bir araştırmaya rastlanamamıştır. Bu çalışmada ise öğrencilerin süreçte nasıl davranışlar sergilediği adım adım incelenmiş, sergilenen davranışların nedenleri BTKO özellikleriyle ilişkilendirilerek tartışılmıştır. BTKO'lar herkes kodlama yapabilir mantığıyla oluşturulan ara yüzlere sahiptir ve geleneksel programlama dillerinden en büyük farklılaşması blok yapıları ile sürükle-bırak şeklinde kod yazmaya olanak tanımasıdır. Bayman ve Mayer (1988)'ın söz dizimi bilgisi şeklinde ifade ettikleri geleneksel programlama dillerine has ve BTKO'da olmayan yazım bilgisi doğal olarak bu ara yüzlerde devre dışı kalmaktadır. Bu açıdan bakıldığında geleneksel programlama dillerine özgü en temel bilgi olan yazım bilgisi dahi ortadan kalkan bu süreç yeniden tanımlanmaya çalışılmıştır.

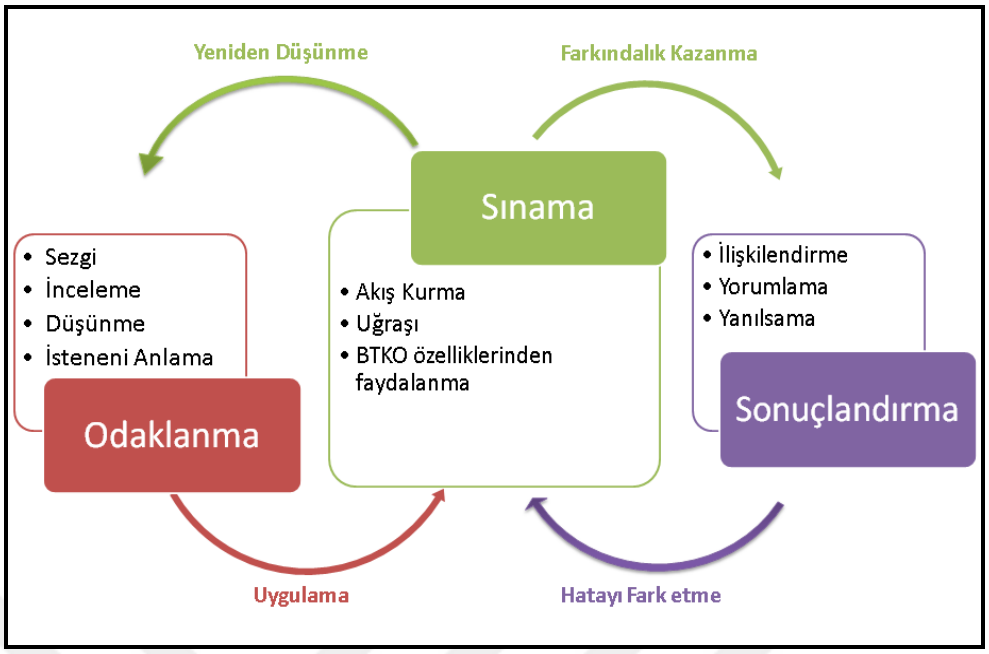
Yarı yapılandırılmış mülakatlardan elde edilen bulgular, BTKO'da çalışılan etkinliklerin farklı programlama yapılarını içerseler de öğrencilerin sergiledikleri davranışların bazılarının değişmediğini, kısmen programlama yapısı çerçevesinde şekil değiştirdiğini göstermektedir. Tüm mülakatlardan elde edilen bulgularda özellikle BTKO'ya özgü davranışların neredeyse tüm programlama yapılarında (temel taşlar, döngüler, koşul yapıları, fonksiyonlar ve hata ayıklama) ortaya çıktığı görülmüştür. Bu durum BTKO'ya ilişkin sahne özelliklerinin incelenmesi, blok yapılarının rastgele değiştirilmesi, akışı yavaşlatarak test etme, açı göstergesini kullanarak parametre tespiti, önceki sahnelerden yararlanma gibi davranışlar hemen hemen tüm programlama yapılarıyla ilişkili sahnelerde görülmüştür.

Örneğin; *problem sahnesini inceleme* davranışı hemen hemen tüm sahnelerde karşımıza çıkmaktadır. Bu davranış genellikle sürecin başında gerçekleşmiş olsa da

sahnede istenilen çözüme ulaşamayan öğrencilerin süreç içerisinde tekrar tekrar problem sahnesini inceleme davranışına başvurduğu belirlenmiştir.

Diğer taraftan bazı davranışların sadece bazı programlama yapılarını içeren sahnelerde ortaya çıktığı da görülmektedir. Nitekim yapılar arası benzerlikleri fark etme davranışları sadece döngüler ve koşul yapılarında ortaya çıkarken, diğer yapılarda görülmemiştir. Code.org sistemi programlama yapısı olarak benzer yapılarda benzer tasarımları kullandığı bilinmektedir. Örneğin; döngü anlamı taşıyacak birbirinden farkı kullanım şekli olan bloklarını aynı renk ve benzer formda tasarlamıştır, böylelikle öğrencilerin bu benzerlikten yola çıkarak tekrarlayan yapılar arasındaki ilişkiyi kolaylıkla kurabildikleri gözlemlenmiştir. Yine blokların programlama mantığının birbirine benzemesinden yola çıkarak koşul yapıları içeren eğer ve eğer/değilse blokları aynı renk ve benzer formda tasarlanmıştır. Ayrıca arayüzün ders kapmasında bulunan her sahnesinde akış yavaşlatmaya imkân vermemesi akış yavaşlatarak test etme davranışının ortaya çıktığı durumları da ilgili sahnelerle sınırlamıştır. Öte yandan gündelik yaşamla ilişkilendirme davranışı günlük yaşantılarındaki hareketlerden yola çıkarak temel taşlar, gün içindeki durumlara karşı karar verme mekanizmalarıyla kıyasladıkları için koşul yapılarında sıklıkla ortaya çıktığı, nadiren de olsa döngü yapılarında da öğrenciler tarafından kullanılabildiği görülmüştür. Hata ayıklama ve fonksiyon yapılarında ise bu davranışa hiç rastlanmamıştır. Bu iki yapının diğer programlama yapılarına göre daha soyut düşünme becerileri gerektirdiği sebebiyle öğrencilerin gündelik yaşamlarıyla bir ilişki kuramadıkları düşünülmektedir.

Bütün bu değerlendirmeler ışığında bu çalışmada yürütülen etkinliklerde problem çözme sürecinde ortaya çıkan davranışlar en genel anlamda; Odaklanma, Sınama ve Sonuçlandırma aşamalarıyla ortaya konulabilir. Bu aşamaların birbirleriyle olan geçişleri aşağıdaki Şekil 25'te sunulmaktadır.



Şekil 25. Code.org BTKO'da ortaya çıkan problem çözme süreçleri temel aşamaları

Buradaki temel aşamalar olan odaklanma, sınama ve sonuçlandırma aşamaları farklı faktörlerin etkisiyle bazı durumlarda birbirini takip eden sırada ileriye doğru devam eden biçimde gerçekleşirken, bazı durumlarda ise aşamalar arası döngüsel geçişler söz konusu olabilmektedir. Örneğin, bir problemde sınama aşamasında deneme yanılma süreçlerini içeren uğraşı davranışı gösteren öğrenci çözüme ulaşamadığında yeniden düşünme çerçevesinde odaklanma aşamasına geri dönebilmektedir.

Sürecin başlangıcı aşamasında programlama adımları farklı tanımlamalara sahip olsa da süreç genellikle Problemi Anlama ile başlayıp Değerlendirme ile son bulmaktadır. Bu çalışmada BTKO olarak kullanılan Code.org üzerinde öğrenci davranışları incelendiğinde, ilk etapta problemi anlamaktan çok sahneyi inceleme, isteneni anlama ve üzerine düşünme davranışlarını sergiledikleri ve bu esnada da sezgi ve tahminlere sıklıkla başvurdukları görülmüştür. Genellikle problemi anlama olarak verilen ilk adım yerine BTKO üzerinde farklı bir şekilde ortaya çıkan bu süreç odaklanma olarak yorumlanmıştır. Öğrencinin sahnelere ve blokların bilgisini kavramaya çalıştığı, problem durumunu anlamaktan çok sahnede isteneni ifade ederek çözüme ulaşmaya çalıştığı gözlemlenmiştir. Öğrenci sahnede kendinden isteneni rahatlıkla ifade ederken, sahnede mevcut problemi anlamaya çaba harcamamaktadır. Zaten sistemin de böyle bir beklentisi olmadığı, hatalı çözümlerle de bölümün geçilmesinin mümkün olduğundan çıkarılabilir. Bu doğrultuda genel olarak; odaklanma aşamasında öğrencilerin daha çok sezgi, inceleme, düşünme, isteneni anlama gibi davranışlar sergilemektedirler. Öğrenciler sınama aşamasında çoğunlukla akış kurma, uğraşı, BTKO özelliklerinden faydalanma gibi

davranışları sergilemektedirler. Diğer yandan ilişkilendirme, yorumlama, yanılsama gibi davranışlar ise daha çok sonuçlandırma aşamasında ortaya çıktığı söylenebilir. Elbette problem sürecini belirli bir zaman dilimi olarak ele aldığımızda bazı davranışların farklı aşamalarda sergilenebilir olduğu da dikkatlerden kaçırılmamalıdır.

Diğer yandan burada belirlenen aşamalar arası bazı durumlarda geri dönüşler de olabilmektedir. Bu dönüşler çoğunlukla sınamadan odaklanmaya doğru yeniden düşünerek veya sonuçlandırmadan sınamaya hatayı fark ederek gerçekleşmektedir. Örneğin odaklanmadan sınamaya geçişte öğrencilerin genellikle uygulamaya hazır hale geldiği görülürken sınamadan sonuçlandırmaya geçerken farkındalık kazandıkları değerlendirilmektedir.

Geleneksel programlama dillerinde problem çözme sürecini ele alan çalışmalar benzer bir adımlama ile süreci özetlemekte ve genellikle 4 veya 5 adımda süreci ifade ettikleri görülmektedir. Programlama sürecine ilişkin işlem adımları örnekleri incelendiğinde Çetin (2012) problem çözme sürecini: 1. Problemin anlaşılması, 2. Çözümle ilgili stratejinin seçilmesi, 3. Seçilen stratejinin uygulanması, 4. Çözümün değerlendirilmesi şeklinde ele alırken; Benzer şekilde Nance (2016) programlama ortamlarında problem çözmeyi; 1) problemi anlama 2) çözümü planlama, 3) planı uygulama ve 4) geri dönme ve tekrar bakma şeklindeki adımlardan oluştuğunu ifade etmektedir. Çok daha bütüncül bir yaklaşımla Casey (1997) programlamayı: Belirli Problemi Anlama, Derleme Hata giderme, Doğrulama aşamalarının bir araya gelmesiyle oluşan bir aktiviteler zinciri olarak ele almıştır.

Bu çalışmanın bulguları BTKO ile problem çözülürken programlama dillerindeki bazı süreç adımlarının hiç yaşanmadığını, bazılarının ise şekil değiştirdiğini göstermektedir. Programlama yapıları çerçevesinde farklı sahnelerdeki problemler üzerine yapılan analizler BTKO'da problem çözme sürecinin temel olarak 3 aşamada ele alınabileceğine işaret etmektedir. Diğer programlama dillerinde olmazsa olmaz bazı süreçler, BTKO'nun sağladığı özellikleri ile bu sürecin işlem adımlarının dışında bırakılabilmektedir. Yapılan süreç aşamalandırmalarında genellikle problemi anlama ile başlayan ilk adım, BTKO'da öğrencilerin anlamadan dahi sezgi ve tahminlerini kullanarak problem çözme sürecini başlatabilmesiyle diğer yapılarda elzem olan bu ilk adım devre dışı kalabilmektedir. Öte yandan hemen hemen tüm aşamalandırmalar değerlendirme basamağı ile sonuçlanmakta iken, BTKO olarak seçilen Code.org sisteminde öğrenci rastgele ulaşabildiği veya fazla blok kullanarak esas olan programlama yapısına ait blokları dahil etmedikleri akışları ile bölüm geçebildikleri için değerlendirme basamağı etkisini yitirmektedir. Nitekim Kaucic ve Asic (2011), Scratch ortamında yaptıkları araştırmada basitleştirilmiş ortamlarda verilmesinin programlamayı öğrenmeyi kolaylaştırabildiğini belirlemiştir. Bu sebeplerden

ötürü Şekil 25'te özetlenen BTKO problem çözme süreçleri temel aşamaları tipik programlama süreçlerine kıyasla sadeleştirilmiş ve BTKO'ya özgü süreçlere göre şekillendirilmiştir.

5. 2. Code.org Özelliklerinin Problem Çözmedeki Rolü

Genel olarak değerlendirildiğinde BTKO'ların ara yüzlerinde sunmuş oldukları araçların problem çözmeye uygunluğu ve problemi sunuş şekilleri çözüm sürecinde önemli görülmektedir.

Programlama yapılarına göre değerlendirildiğinde öğrencilerin genellikle ana blokların sahip olduğu kodlama bilgisini ve sahnelerde onlardan istenilen görevleri anlamada zorluk çekmediği, ancak temel taşlar ile birlikte diğer programlama bilgilerini kullanarak amaca uygun bir akış oluşturmakta zorlandığı çokça görülmüştür. Özetle öğrenci herhangi bir sahnede koşul veya döngü yapısı kullanacağına hemen karar verebilmekte ancak o blokların için doğru bir algoritma ile doldurmakta sıkıntı yaşamaktadır. Örneğin; ekran kaydına yönelik hemen tekrarla kullanmaya karar verebilmesi sorulduğunda Ö6'nın Sahne1 hakkında "içine ne dizeceğime karar vermek daha zor geldi" yanıtını vermiştir. Ö6'nın ifadesinden de anlaşılacağı gibi öğrenciler çoğunlukla bir problem durumu verilen sahne karşısında döngü mantığını kullanabileceklerini döngünün içinde tekrar eden akışı kurmaktan daha kolay çözümlemektedirler. Bu durumun BTKO'nun kullanılacak programlama yapılarının belirlenmesinde kolaylaştırıcı bir rol oynadığına işaret ettiği söylenebilir.

Bu çalışmada Code.org'un kullanılmasının muhtemel programlama yapılarını sahnenin ilgili kısmında sunuyor olması, öğrencilerin programlama yapılarını hatırlamasını kolaylaştırıcı rol oynadığı düşünülebilir. Her ne kadar öğrenciler için kullanabilecekleri blokları hatırlamak kolay olsa da bu bloklar arasında doğru seçim için karar verebilmek çok kolay olmamaktadır. Etkinliklerdeki problemlerin çözümü sırasında bu durumun benzer örneği sıklıkla yaşanmış birçok sahnede öğrenciler tekrarla bloğunu hemen almalarına rağmen içine dizecekleri akışı doğru sıralamakta zorluk çekmişlerdir.

Programlama için kullanılan derleyicilere kıyasla BTKO'da ara yüzün kullanıcılara sağladığı avantajlardan birisi olarak kurgulanan akışın sonucunu test edebilmesi gösterilebilir. Bu şekilde öğrencilerin zihninde dizdiği akışı bloklarla sahnelerde uygulayarak işlemin sonucunu görebilmektedir. Doğru çözüm gerçekleştiğinde öğrenciler çoğunlukla sahnelere tekrar dönüp bakma gereği hissetmemişlerdir. Bu durumun nedenlerinden birisi olarak Code.org'da öğrencilerin o sahnedeki etkinliği bir oyun veya kazanılması gereken bir durum olarak hissettirmesi olarak düşünülebilir. Öğrenciler o

sahnede istenen sonuca ulaştığında bir sonraki sahneyi merak ettiğinden mevcut sahnedeki gerek programlama yapılarını, gerekse yeniden gözden geçirerek neler yapmış olduğuna ilişkin değerlendirmeye yönelik davranışları fazlaca gösterememiş olabilirler. Dolayısıyla akışın hızlı biçimde sahnede test edilebilmesi bir yandan avantaj olarak görünse de diğer yandan öğrencilerin deneme yanılmalarla sonuca ulaşabilmelerine de yol açmaktadır. Benzer şekilde bazen tek deneme sonucunda bile olsa öğrencilerin anlamadan çözüme ulaşabildikleri görülmüştür. Tesadüfi sonuçlar ile hedeflenen görevin tamamlanması öğrencilerin tekrar o görev ile ilgili olarak düşünmelerini gerektirmemektedir. Bu durum BTKO'nun olumsuz bir durumu olarak değerlendirilebilir. Nitekim özellikle hata ayıklama sahnelerinde var olan akışın ne iş yaptığını ve nasıl ilerlediğini tahmin edebilmek eksiği bulma bakımından oldukça önemlidir. Buna karşın öğrencilerin birçoğunun anlamadan dahi çözüme ulaşabildikleri görülmüştür.

Bu çalışmada BTKO olarak seçilen Code.org sisteminde arayüze göre belirlenmiş görevlerin tamamlanmasının ardından gelen sahnelerde programlama olarak mantığını düşünüldüğünde fonksiyon yapıları diğer yapılardan daha farklı ilişkilendirmeler gerektirmektedir.

Geleneksel metin tabanlı programlama arayüzlerinde BTKO ile kıyaslandığında sadece sonucun kullanıcıya sunulduğu görülmektedir. Herhangi bir programlama dili kullanıcısının sonuç istediği yönde olmadığında başvurabileceği en büyük destekçisi hatalı yazımları gösterebilen bir derleyici olabilir, ancak bu derleyici programlar yalnızca programlama yapılarının hatalı kullanımını veya yazım hatalarını renklendirerek oraya dikkat çekebilmektedir. BTKO olarak adlandırılan sistemlerde ise metin tabanlı dillere kıyasla kullanıcı süreç içerisinde bulunan ve akışa eklediği her adımın sonucunu somut olarak görebilmektedir. Örneğin; Code.org üzerinde öğrenci bir akış yazdığında öğrenciler sahneyi çalıştırıp karakteri izlemektedir, karakteri gitmesi gereken hedefe ulaşamadığında süreci somut olarak izleyebildiği için hangi adımda hata yaptığını tespit edip orada bulunan bloklarına müdahale edebilmektedir.

Özetle, akışın çalıştırılabilir ve izlenebilir nitelikte olması BTKO'da sonucun tahmin edilebilmesini mümkün kılmaktadır. Sonuca nasıl ulaşılabileceği üzerine tahminde bulunma veya hata esnasında akışı düzeltirken sezgilerden, deneme yanılma yönteminden yararlanabilme Code.org gibi BTKO'nun problem çözme sürecine getirdiği yeni durumlar olarak değerlendirilebilir. Oluşturulan akışın izlenebilir olması ile sahnenin zihinlerinde canlandırılmasına alışan öğrencilerin bir süre sonra kendi zihinlerinde akışlarını canlandırarak sonuca ulaşacak akışı ve blok yapısını belirlemelerini kolaylaştırdığı düşünülebilir. Araştırmacılar analiz etme, sentez yapma ve değerlendirme gibi becerilerin programlamada problem çözme için vazgeçilmez olduğunu ifade

etmektedirler (Palumbo ve Reed, 1988). Code.org'un sahnedeki etkinliği sunma şeklinin problem çözümüne yönelik analiz ve sentezlerin yapılmasını kolaylaştırdığı söylenebilir. Nitekim öğrencinin çoğunlukla problemi analiz ederek yapması gerekenleri düşünmesi vegerekli kod yapıları öğrenciye sunulmakta öğrenciden bunları sentezlenmesi mümkün olabilmektedir. Ancak değerlendirme noktasında çoğu kez Sahneler arasında öğrencinin farklı nedenlerle hızlı geçişleri, neyi neden yaptığına dair düşünmesini sınırladığı düşünülebilir.

5. 3. Araştırmayı Farklılaştıran Bazı Durumlar ve Sınırlılıklar

Bu çalışma Code.org ortamında Ders-3 içerisinde yer alan bölümler üzerinden yürütülmüştür. Toplamda 21 farklı derse sahip Ders-3 modülünden bilgisayarsız etkinlikler ve oyun laboratuvarı etkinlikleri çıkarıldıktan sonra kalan tüm öğretici bölümlerin her birini temsil edecek nitelikteki seçilen sahneler üzerinden kayıtlar alınmış ve mülakatlar gerçekleştirilmiştir.

Bu çalışma sadece Ders-3 içerisindeki bölümleri içerse de, seçilen bölümlerin temel programlama yapılarını içeriyor olması, farklı zorluk derecelerinde olması, ilgili yaş grubundaki öğrencilerin mevcut matematik bilgilerini kullanabilecekleri etkinlikleri içeriyor olması bu bölümlerin seçiminde önemli olmuştur. Çalışmada tüm sahnelerdeki etkinlikleri yapmak yerine programlama yapılarının yansıtılması, ve farklı problem türlerinin temsil edilmesi esas alındığından, elde edilen bulguların genel olarak Code.org etkinliklerini yansıtan değerlendirmeler içerdiği düşünülmektedir.

Çalışma için belirlenen 10 sahne, 15 öğrenciden oluşan bir grupla bir dönem boyunca yürütülmüştür. Araştırmanın 15 kişilik bir grupla yürütülmüş olması, BTKO'da problem çözmeye yönelik olarak muhtemel davranışların tümüne yönelik bir çerçeve oluşturmada sınırlı kaldığı söylenebilir. Ancak, çalışmada temel taşlar, koşul, döngüler, fonksiyonlar, hata ayıklamaya yönelik olarak ortaya çıkan ortak davranış biçimleri, BTKO'da gerçekleşen problem çözme sürecinin modellenebilir olduğuna işaret etmektedir.

Sahnelerdeki etkinlikler okul dışında da yapılabilmeye imkan tanıyor olsa da, sistemdeki öğretmen kontrolü özelliğiyle öğrencilerin etkinlikleri sınıf içinde yapmaları sağlanarak, gözlenebilmeleri mümkün olmuştur. Araştırmacının aynı zamanda öğretici olması süreçteki davranışların gözlemlenebilmesi ve mülakatların yürütülmesini kolaylaştırmıştır. Ayrıca öğrencilerin sistem üzerindeki davranışlarında veri kaybı yaşanmaması için seçili her bölümde ekran videoları kayıt altına alınmış ve mülakatlar öğrencilere kayıtlarda sahneler üzerindeki eylemleri izletilerek eş zamanlı bir şekilde

gerçekleştirilmiştir. Böylelikle problem çözümü sürecindeki düşünme yolları ve programlama yapılarına yaklaşımları derinlemesine incelenmiştir.

Diğer yandan, bu çalışma Code.org veya benzeri BTKO çerçevesinde yapılan çalışmalardan problem çözme sürecine odaklanması yönüyle farklılaşmaktadır. Önceki birçok çalışma bu tür ortamlarda problem çözmenin öğrenciye bilişsel veya duyuşsal alanda kazandırdıkları üzerine yoğunlaşırken bu çalışma problemlerin nasıl çözüldüğü üzerine eğilmiştir. Elbette, problemlerin çözümüne ilişkin yollar BTKO'daki problemlerin doğası, yaş grubu, ortam ara yüzü gibi bazı faktörlerden etkilenebilir. Örneğin Code.org'da problemler sahnede başlangıcı ve sonu belli olan, öğrenciden başlangıçtan sona ulaşmak için gereken programlamayı yapmasını isteyen bir yapıda sunulmuştur. Farklı BTKO'larda sadece temel yapılar verilerek, problemin öğrenci tarafından oluşturularak çözüldüğü ortamlarda problem çözüm süreci kısmen farklılaşabilir. Örneğin Scratch vb. gibi BTKO'larda probleme ilişkin sahne, karakter vb. düzenlemeleri de öğrenci hazırlayacağından öğrencinin ortamı kullanmaya yönelik daha çok bilgiye ihtiyacı olacaktır. Sonuç olarak BTKO'ların öğrenci için sağladıkları ve öğrencinin bunları nasıl kullanacağına ilişkin bilgisi problem çözme sürecini farklılaştırabileceği düşünülebilir. Nitekim bazı araştırmacılar süreci kolaylaştırmak için farklı BTKO'ların farklı özelliklerini bir arada kullanarak olumlu sonuçlar elde etmişlerdir. Örneğin, Patan (2016) çalışmasında okul öncesi öğrencileri için bir "Kodlama Saati" programının tasarlanması, geliştirilmesi ve değerlendirilmesi süreçlerinden oluşan bir öğretim programı geliştirmeye çalışmıştır. Çalışmanın programlama öğretiminde Kodable ve Code.org; animasyon tasarımında Flipboom ve dijital hikayelemede Scratch kullanılmış sonuç olarak BTKO'lar ile yapılan bu çalışmada öğrencilerin programlamaya karşı pozitif tutum sergilediklerini elde etmiştir. Bu çerçevede BTKO'ları oluşturan tasarımcıların problemin belirlenmesi ve çözümü için öğrenciye sunulacak araçların ne kadar olmasının belirlenmesinde, çözülecek problemlerin niteliği kadar, öğrencinin çözüm yollarını da dikkate almaları uygun olacaktır.

6. SONUÇ VE ÖNERİLER

Bu bölümde araştırmadan elde edilen bulgulara dayanılarak sonuçlar özetlenmiştir.

6. 1. Sonuçlar

Bu çalışma bir BTKO olarak kullanılan Code.org ortamı üzerinde 10 hafta boyunca yürütülen problem çözme etkinlikleri gerçekleştirilmiştir. Uygulama boyunca öğrencilerin problem çözme süreçleri incelenerek Code.org üzerinde problem çözme süreçlerini tanımlayan bir çerçeve ortaya konulmuştur.

1. 6. Sınıf öğrencilerinin Code.org BTKO'da problem çözme süreçlerinin en genel anlamda; odaklanma, sınama ve sonuçlandırma gibi 3 temel aşamayı içerdiği ortaya çıkmıştır.
2. Odaklanma, Sınama ve Sonuçlandırma aşamaları farklı faktörlerin etkisiyle bazı durumlarda birbirini takip eden sırada ileriye doğru devam eden biçimde gerçekleşirken, bazı durumlarda ise aşamalar arası döngüsel geçişler söz konusu olabilmektedir.
3. Odaklanma aşamasında öğrencilerin daha çok sezgi, inceleme, düşünme, anlama gibi davranışlar sergiledikleri; sınama aşamasının akış kurma, deneme yanılma, BTKO özelliklerinden faydalanma, blokların seçimi, sıralama-blok değiştirme, rastgele blokları karıştırma gibi davranışları içerdiği; sonuçlandırma aşamasında ise; hedefe ulaşma, anlamlandırma, anlamadan geçme gibi davranışların gerçekleşmektedir.
4. Aşamalar arası bazı durumlarda geri dönüşler olmaktadır. Bu dönüşler çoğunlukla sınamadan odaklanmaya "yeniden düşünmek" amacıyla, sonuçlandırmadan sınamaya "hatayı fark etme" sonucunda gerçekleşmektedir.
5. Odaklanmadan sınamaya geçişte öğrencilerin genellikle uygulamaya hazır hale geldiği; sınamadan sonuçlandırmaya geçerken farkındalık kazandıkları değerlendirilmektedir.
6. Code.org'da problem çözme sürecinde öğretilen programlama yapısı (koşul, döngü, hata ayıklama vb.) veya sahnedeki problemin doğası problem çözme sürecindeki 3 temel aşamanın doğrusal veya döngüsel yapıda olmasını etkileyebilmektedir.

7. Tipik programlama ortamlarından farklı olarak Code.org problem çözme sürecinde öğrencilerin problem çözme sürecinde birçok durumda sezgilerinden yararlanmalarını sağlamaktadır.
8. Code.org'da tipik problem çözme süreçlerindeki değerlendirme aşaması belirgin olarak farklılaşmakta, çoğu zaman problemi sonuçlandırma biçiminde gerçekleşmektedir.

6. 2. Öneriler

6. 2. 1. Araştırma Sonuçlarına Dayalı Öneriler

Bu Bu bölümde çalışmadan elde edilen sonuçlar doğrultusunda çevrimiçi öğrenme ortamları aracılığıyla BTKO'da öğretim yapacak olan eğitimcilere, ders tasarımcılarına, organizasyonlara ve öğrencilere yol gösterici bazı önerilere yer verilmiştir.

1. BTKO'lar kullanılarak gerçekleştirilen eğitimlerde dersin yürütücüsünün BTKO'nun programlama yapılarını ele alış özelliklerini ve etkinliklerin içerdiği problemlerin doğasını bilmeleri öğrencilerin bu ortamlarda problem çözme becerisi kazanmalarına katkı sağlaması anlamında önemlidir.
2. İlgili yaş grubuna göre sahnelerin seçimi problem çözme sürecinden hedeflenen kazanımlara ulaşılmasında faydalı olacaktır.
3. Çalışmada öğrencilerin özellikle tipik problem çözmedeki değerlendirme aşamasını çok yaşayamadıkları, farklı nedenlerle problemi sonuçlandırmaya yöneldikleri görülmüştür. Özellikle problemler ilgili değerlendirmelerin yapılmasına yönelik öğretim sürecinde tedbirler alınabilir.
4. Code.org BTKO'nun öğretmene sağladığı öğrenci takibi, öğrenciye sağladığı süreci izleme özelliklerinin gerektiği zaman kullanılması yapılan uygulamalarda problem çözme süreçlerini kolaylaştırabilir.

6. 2. 2. İleride Yapılabilecek Araştırmalara Yönelik Öneriler

1. Bu çalışma Code.org BTKO'da seçilen sahnelerdeki problemler ile sınırlıdır. Benzer çalışmalar diğer sahnelerde veya başka BTKO'da da tekrarlanıp süreçler modellenebilir.

2. Bu çalışmada ekran kayıtları merkezli üç farklı veri toplama aracından gelen verilerin analiz edilmesinde analiz birimi olarak belirlenen davranışlara bu çalışmada problem çözme süreci çerçevesinde yüklenen anlam farklı çalışmalarla test edilebilir.
3. Her ne kadar bu çalışma tüm BTKO'lar için genelleme yapmasa da; örneklem daha uzun süreler gözlenerek süreçteki temel aşamaları şekillendiren davranışların genellenebilirliği ile ilgili çalışmalar tasarlanabilir.
4. Öğrencilerin tipik problem çözme davranışlarının BTKO'da ne şekilde değiştiği belirlenebilir.

7. KAYNAKLAR

Aksoy, B. (2004). Coğrafya öğretiminde probleme dayalı öğrenme yaklaşımı. Yayınlanmamış doktora tezi, Gazi Üniversitesi, Ankara.

Altun, M. (2004). *İlköğretim ikinci kademede (6, 7 ve 8. sınıflarda) matematik öğretimi*. Bursa: Alfa Yayınları.

Ananiadou, K. and Claro, M. (2009). 21st century skills and competences for new millennium learners in OECD countries. *OECD Education Working Papers*, *41*, 1-34.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A. and Engelhardt, K. (2016). *Developing computational thinking in compulsory education*. https://www.researchgate.net/profile/Stefania_Bocconi/publication/312039564_Developing_Computational_Thinking_in_Compulsory_Education_Implications_for_policy_and_practice/links/586bb71508ae6eb871bb6946/Developing-Computational-Thinking-in-Compulsory-Education-Implications-for-policy-and-practice.pdf adresinden 18.03.2017 tarihinde erişildi.

Barr, D., Harrison, J. and Conery, L. (2011). Computational thinking: A dijital age skill for everyone. *Learning & Leading with Technology*, *38*(6), 20-23.

Bayman, P. and Mayer, R. (1988), Using conceptual models to teach basıc computer programming. *Journal of Educational Psychology*, *80*(3), 291-298.

Bers, M. I., Flannery, L., Kazakoff, E. R. and Sullivan, A. (2014). Computational thinking and tinkering: exploration of an early childhood robotics curriculum. *Computers and Education*, *72*, 145–157.

Berry, M. (2015). *QuickStart primary handbook*. Swindon (UK): BCS.

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M. and Rumble, M (2012). Defining 21st century skills. In P. Griffin, B. McGaw & E. Care (Eds.), *Assessment and teaching of 21st century skills* (pp. 17-66). New York: Springer.

Brown, Q., Mongan, W., Kusic, D., Garbarine, E., Fromm, E. and Fontecchio, A. (2013). *Computer aided instruction as a vehicle for problem solving: Scratch programming environment in the middle years classroom*. *Retrieved September*, *22*. http://www.pages.drexel.edu/~dmk25/ASEE_08.pdf adresinden 15.03.2017 tarihinde erişildi.

Brown, W. (2015). *Introduction to algorithmic thinking.* www.cs4fn.com/algoritmicthinking.php adresinden 21.04.2017 tarihinde erişildi.

Bundy, A. (2007). *Computational thinking is pervasive.* http://www.inf.ed.ac.uk/publications/online/1245.pdf adresinden 15.03.2017 tarihinde erişildi.

Burke, Q. (2012). The markings of a new pencil: Introducing programming-aswriting in the middle school classroom. *Journal of Media Literacy Education, 4*(2), 121–135.

Casey, P. J. (1997). Computer programming: A medium for teaching problem solving. *Computers in the Schools*, *13*(1-2), 41-51.

CCEA (2017). *Computing at school: Northern ıreland curriculum guide for post primary schools.* www.ccea.org.uk adresinden 21.10.2017 tarihinde erişildi.

Chang, C. K. (2014). Effects of using Alice and Scratch in an introductory programming course for corrective instruction. *Journal of Educational Computing Research*, *51*(2), 185-204.

Chiu, C. F. (2014, March). *Use of problem-solving approach to teach scratch programming for adult novice programmers*. Paper presented at Proceedings of the 45th ACM technical symposium on Computer science education, New York.

Curzon, P. (2015). Computational thinking: Searching to speak. http://www.inf.ed.ac.uk/publications/online/1245.pdf adresinden 18.05.2017 tarihinde erişildi.

Çatlak, Ş., Tekdal, M. ve Baz, F. Ç. (2015). Scratch yazılımı ile programlama öğretiminin durumu: bir doküman inceleme çalışması. *Journal of Instructional Technologies and Teacher Education, 4*(3), 13-25.

Çetin, E. (2012). Bilgisayar programlama eğitiminin çocukların problem çözme becerileri üzerine etkisi. Yayınlanmamış yüksek lisans tezi, Gazi Üniversitesi, Ankara.

Davey, L. (1991). The application of case study evaluations. *Practical Assessment, Research and Evaluation, 2*(9), 1-2.

Deek, F. P., Turoff, M. and McHugh, J. A. (1999). A common model for problem solving and program development. *IEEE Transactions on Education*, *42*(4), 331–336.

Demirer, V. ve Nurcan, S. A. K. (2016). Dünyada ve Türkiye'de programlama eğitimi ve yeni yaklaşımlar. *Eğitimde Kuram ve Uygulama*, *12*(3), 521-546.

Denner, J., Werner, L. and Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers and Education*, *58*(1), 240–249.

Denzin, N. K. and Lincoln, Y. S. (2011). *The Sage handbook of qualitative research*. London: Sage.

Du, J., Wimmer, H. and Rada, R. (2016). “Hour of code”: can ıt change students’attitudes toward programming? *Journal of Information Technology Education: Innovations in Practice, 15*, 52-73.

Eğitimi Araştırma ve Geliştirme Dairesi Başkanlığı [EARGED]. (2011). *MEB 21.yy öğrenci profili.* http://www.meb.gov.tr/earged/earged/21.%20yy_og_pro.pdf adresinden 24.10.2017 tarihinde erişildi.

Fessakis, G., Gouli, E. and Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers and Education*, *63*, 87–97.

Garner, S. (2003, February). *Learning resources and tools to aid novices learn programming.* Paper presented at Informing Science & Information Technology Education Joint Conference (INSITE), Pori: Finland.

Genç, Z. ve Karakuş, S., (2012, Eylül) *Tasarımla öğrenme: Eğitsel bilgisayar oyunları tasarımında scratch kullanımı.* Paper presented at 5th International computer and instructional technologies symposium, Elazığ.

Giordano, D. and Maiorana, F. (2015, March*). Teaching algorithms: Visual language vs flowchart vs textual language*. Paper presented at Global Engineering Education Conference (EDUCON), Tallinn*.*

Gomes, A. and Mendes, A. J. (2007, September). *Learning to program-difficulties and solutions*. Paper presented at International Conference on Engineering Education–ICEE, Coimbra, Portugal.

Grover, S. (2011, April). *Robotics and engineering for middle and high school students to develop computational thinking*. Paper presented at annual meeting of the American Educational Research Association, New Orleans, LA.

Grover, S. and Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43.

Gültekin, K. (2006). Çoklu ortamın bilgisayar programlama başarısı üzerine etkisi. Yayımlanmamış yüksek lisans tezi, Hacettepe Üniversitesi, Ankara.

Güneş, A. ve Karabak, D. (2013). Ortaokul birinci sınıf öğrencileri için yazılım geliştirme alanında müfredat önerisi. *Eğitim ve Öğretim Araştırmaları Dergisi, 2(3),* 21-33.

Günüç, S. Odabaşı, F. ve Kuzu, A. (2013). The defining characteristics of students of the 21st century by student teachers: A twitter activity. *Journal of Theory and Practice in Education, 9*(4), 436-455.

Hung, Y. C. (2008). The Effect of problem-solving instruction on computer engineering majors'performance in Verilog (HDL) programming. *IEEE Transaction on Education*, *51*(1), 131-137.

Ismail, M. N., Ngah, N. A. and Umar, I. N. (2010). Instructional strategy in the teaching of computer programming: a need assessment analyses. *TOJET: The Turkish Online Journal of Educational Technology*, *9*(2), 125-131.

International Society for Technology in Education [ISTE]. (2016). *ISTE's educational technology standards for students.* https://www.iste.org/standards/for-students adresinden 20.12.2016 tarihinde erişildi.

Johnson, L., Adams Becker, S., Estrada, V., Freeman, A., Kampylis, P., Vuorikari, R. and Punie, Y. (2014). *Horizon report europe: 2014 schools edition* http://www.leervlak.nl/wp-content/uploads/2014-nmc-horizon-report-EU-EN online.pdf adresinden 14.05.2016 tarihinde erişildi.

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educational Technology Research and Development*, *48*(4), 63-85.

Kalelioglu, F. and Gülbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: a discussion from learners'perspective. *Informatics in Education, 13*(1), 33-50.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior*, *52*, 200-210.

Kaucic, B. and Asic, T. (2011, May). Improving introductory programming with Scratch? Paper presented at Proceeding of the 34th MIPRO International Conference, Opatija, Croatia.

Keren, G. and Fridin, M. (2014, June). Kindergarten social assistive robot (KindSAR) for children's geometric thinking and metacognitive development in preschool education: A pilot study. *Computers in Human Behavior*, *35*, 400–412.

Kert, S. B. ve Uğraş, T. (2009, Mart). *Programlama eğitiminde sadelik ve eğlence: Scratch örneği*. First International Congress of Educational Research Kongresi'nde sunulan bildiri, Çanakkale.

Lee, Y. J. (2010). Developing computer programming concepts and skills via technology-enriched language-art projects: A case study. *Journal of Educational Multimedia and Hypermedia, 19*(3), 307–326.

Lin, N. (1976). *Foundations of social research*. New York: McGraw-Hill Companies.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M. and Simon, B. (2004, June). A multi-national study of reading and tracing skills in novice programmers. Paper presented at ACM SIGCSE Bulletin. Leeds, United Kingdom.

Lu, J. J. and Fletcher, G. H. (2009). Thinking about computational thinking. *ACM SIGCSE Bulletin*, *41*(1), 260-264.

Lye, S. Y. and Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51-61.

Mallios, N. and Vassilakopoulos, M. G. (2015). Evaluating students' programming skill behaviour and personalizing their computer learning environment using" the hour of code" paradigm. *International Association for Development of the Information Society, 5*, 131-135.

McCain, T. (2007). *Teaching for tomorrow: Teaching content and problem-solving skills*. California: Corwin Press.

Mcmillan, H. and Schumacher, S. (2010). *Researcher in education* (7th edition). Boston: Pearson

Milli Eğitim Bakanlığı [MEB]. (2016). *Talim ve terbiye kurulu başkanlığı ortaöğretim bilgisayar bilimi dersi (Kur 1, Kur 2) öğretim programı*. Ankara: MEB Yayınevi.

Milli Eğitim Bakanlığı [MEB]. (2017). *Talim ve terbiye kurulu başkanlığı ilköğretim bilişim teknolojileri ve yazılım dersi öğretim programı*. Ankara: MEB Yayınevi.

Meerbaum-Salant, O., Armoni, M. and Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, *23*(3), 239-264.

Nance, S. (2016). Using computer programming to enhance problem-solving skills of fifth grade students. Unpublished doctoral dissertation, University of Florida, Gainesville.

Governing Board of the National Research Council. (2008). *Taking science to school: Learning and teaching science in grades K– 8.* Washington, DC: National Academy Press.

Governing Board of the National Research Council [NRC]. (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington, DC: National Academies Press.

Organisation for Economic Co-operation and Development [OECD]. (2004). Problem Solving for Tomorrow's World. First Measures of Cross- Curricular Competencies, PISA 2003.

Olgun, K. B. (2014). Programlamanın ortaokul öğrencilerinin düşünme stilleri üzerine etkisi. Yayınlanmamış yüksek lisans tezi, İstanbul Üniversitesi, İstanbul.

Ozoran, D., Cagiltay, N. and Topalli, D. (2012, June). *Using scratch in introduction to programming course for engineering students.* Paper presented at 2nd International Engineering Education Conference (IEEC2012), Antalya.

Palumbo, D. B. (1990). Programming language/problem-solving research: A review of relevant issues. *Review of Educational Research*, *60*(1), 65-89.

Palumbo, D. B. and Reed, W. M. (1988). Intensity of treatment and its relationship to programming problem solving. *Computers in the Schools*, *4*(3-4), 119-128.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Patan, B. (2016). Okul öncesi kodlama öğretim programının geliştirilmesi. Yayınlanmamış yüksek lisans tezi, Bahçeşehir Üniversitesi, İstanbul.

Patton, M. Q. (2014*). Nitel araştırma ve değerlendirme yöntemleri*. (M. Bütün ve S.B. Demir, Çev., 3. baskı). Ankara: PegemA Akademi.

Pellas, N. and Peroutseas, E. (2016). Gaming in Second Life via Scratch4SL: Engaging high school students in programming courses. *Journal of Educational Computing Research, 54*(1), 108-143.

Pinto, A. and Escudeiro, P. (2014, June). *The use of Scratch for the development of 21st century learning skills in ICT.* Paper presented at Information Systems and Technologies (CISTI), Barcelona, Spain.

Polya, G. (1980*) On solving mathematical problems in high school*. In S. Krulik (Eds). *Problem solving in school mathematics* (pp.1-2). Reston, Virginia: NCTM.

Prensky, M. (2001). Digital natives, digital immigrants part 1. *On the HORIZON*, *9*(5), 1-6.

Repenning, A., Webb, D. and Ioannidou, A. (2010, March). *Scalable game design and the development of a checklist for getting computational thinking into public schools.* Paper presented at Proceedings of the 41st ACM technical symposium on Computer science education. Milwaukee, Wisconsin, USA.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... and Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67.

Rogozhkina, I. and Kushnirenko, A. (2011). PiktoMir: Teaching programming concepts to preschoolers with a thinking and metacognitive development in preschool education: A pilot study. *Computers in Human Behavior*, *35*, 400–412.

Sáez-López, J. M., Román-González, M. and Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers and Education*, *97*, 129-141.

Sakamoto, K., Takano, K., Washizaki, H. and Fukazawa, Y. (2013, August*). Learning system for computational thinking using appealing user interface with icon-based programming language on smartphones.* Paper presented at Proceedings of the 21st International Conference on Computers in Education (ICCE), Sydney, Australia.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G. and Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*(2), 351-380.

Shin, S., Park, P. and Bae, Y. (2013). The effects of an information-technology gifted program on friendship using scratch programming language and clutter. *International Journal of Computer and Communication Engineering*, *2*(3), 246.

Trilling, B. and Fadel, C. (2009). *21st century skills: Learning for life in our times.* Jossey-Bass: San Francisco, CA.

URL1 : (http://www.csprinciples.org/) CS principles. 15 March 2014.

Utting, I., Cooper, S., Kölling, M., Maloney, J. and Resnick, M. (2010). Alice, greenfoot, and scratch – a discussion. *ACM Transactions on Computing Education (TOCE), 10*(4), 1-11.

Volet, S. E. and Lund, C. P. (1994). Metacognitive instruction in introductory computer programming: A better explanatory construct for performance than traditional factors. *Journal of Educational Computing Research*, *10*(4), 297-328.

Weiss, M. A. (1996). *Algorithms, data structures, and problem solving with C++.* Redwood City: Addison Wesley Longman Publishing Co., Inc.

Wilson, C. (2013). What's up next for code. org? *Computer*, *46*(8), 95-97.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of The Royal Society of London A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725.

Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *ACM Sigcse Bulletin*, *28*(3), 17-22.

Yağcı, M. (2016). Bilişim teknolojileri (BT) öğretmen adaylarının ve bilgisayar programcılığı (BP) öğrencilerinin programlamaya karşı tutumlarının programlama öz yeterlik algılarına etkisi. *Journal of Human Sciences*, *13*(1), 1418-1432.

Su, A., Yang, S. J., Hwang, W. Y., Huang, C. S. and Tern, M. Y. (2014). Investigating the role of computer-supported annotation in problem solving-based teaching: An empirical study of a Scratch programming pedagogy. *British Journal of Educational Technology*, *45*(4), 647-665.

Zhou, Y. (2007). Computational thinking. *Chinese Computer Communications Society*, *3*(11), 111-113.

8. EKLER

Ek 1. Yarı Yapılandırılmış Mülakat Soruları

1) Bu sahnedeki problem durumu nedir? Bu sahne senden ne yapmanı istiyor?
2) Genelde bir sahneyi ilk gördüğünde ne yaparsın?
3) Blokları neye göre seçiyorsun ve akışını nasıl düşünerek oluşturuyorsun?
4) Algoritman çalışmaz ise nasıl bir yol izleyerek hatanı düzeltirsin?
5) Code.org sistemi sahnelerde bazı bilgiler veriyor, bunları okuyor musun? Okuyor isen ne zaman? Okumuyor isen neden?
6) Algoritma yazarken sahnede size çalışma alanında sınırlandırılarak verilen "en fazla blok kullanma sayısına" dikkat ediyor musun?
    a. Koyu yeşil veya açık yeşil olma durumu senin için önemlimi?
    b. Eğer blok sayısını aşmış isen tekrar deneyip çalışma alanında verilen sayıda akışı oluşturmaya çalışıyor musun?
7) Bu sahne kurduğun akış dışında farklı bir yolla da çözülebilir mi?
8) Bu şekilde etkinlikler yapman günlük hayatında düşünme şeklini etkiledi mi?

Ek 2. Ders Takip Çizelgesi

DERS AKIŞI TAKİP FORMU

DERS - 3

SAHNE ADI	BÖLÜM NO															HEDEF	LINK
Sahne2: Labirent	1	2	3	4	5	6	7	8	9	10	11	12	**13**	14	15	Tekrarla	https://studio.code.org/s/course3/stage/2/puzzle/13
Sahne3: Aktör	1	2	3	4	5	6	7	8	9	10	**11**	12	13	14	15	Tekrarla + Mat.- Geo.	https://studio.code.org/s/course3/stage/3/puzzle/10
Sahne5: Sanatçı	1	2	3	4	**5**	6	7	8	9	10	11	12	13	14	15	onksiyonda Parametre Girm	https://studio.code.org/s/course3/stage/5/puzzle/5
Sahne6: Arı	1	2	3	4	5	6	**7**	8	9	10	11	12	13	14	15	Fonksiyon Oluşturma	https://studio.code.org/s/course3/stage/6/puzzle/7
Sahne7: Arı	1	2	3	4	**5**	6	7	8	9	10	11	12	13	14	15	Eğer/Değilse Yapısı	https://studio.code.org/s/course3/stage/7/puzzle/5
Sahne8: Labirent	1	2	3	4	5	6	**7**	8	9	10	11	12	13	14	15	Olana Kadar Tekrarla + Eğer	https://studio.code.org/s/course3/stage/8/puzzle/7
Sahne11: Aktör	1	2	3	4	**5**	6	7	8	9	10	11	12	13	14	15	Yerleştirilmiş Döngüler	https://studio.code.org/s/course3/stage/11/puzzle/5
Sahne12: Çiftçi	1	2	3	4	5	6	7	**8**	9	10	11	12	13	14	15	Olduğu Sürece Döngüsü	https://studio.code.org/s/course3/stage/12/puzzle/7
Sahne13: Arı	1	2	3	**4**	5	6	7	8	9	10	11	12	13	14	15	İç içe döngü	https://studio.code.org/s/course3/stage/13/puzzle/4
Sahne14: Arı	1	2	3	4	5	6	**7**	8	9	10	11	12	13	14	15	hata ayıklama sahneleri var	https://studio.code.org/s/course3/stage/14/puzzle/7

****Numaralan renklendirilerek yukandaki tabloda belirtilen bölümlere geldiğinizde öğretmeniniz tarafından size verilen çalışma kağıdı ile o bölümü tamamlayınız.*

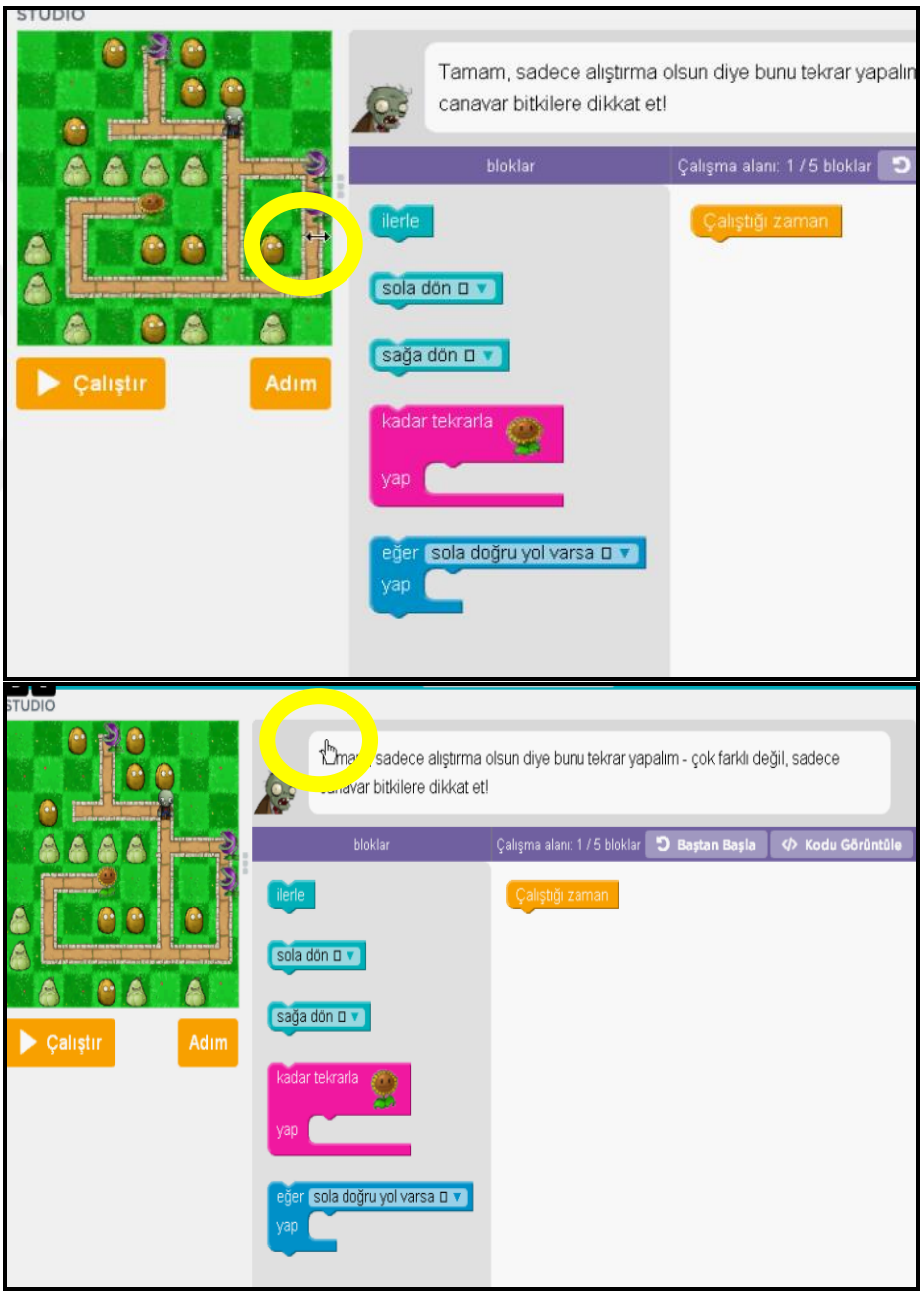
****Ekran videosu kaydetmeyi unutmayınız!*

-BOLŞANS-

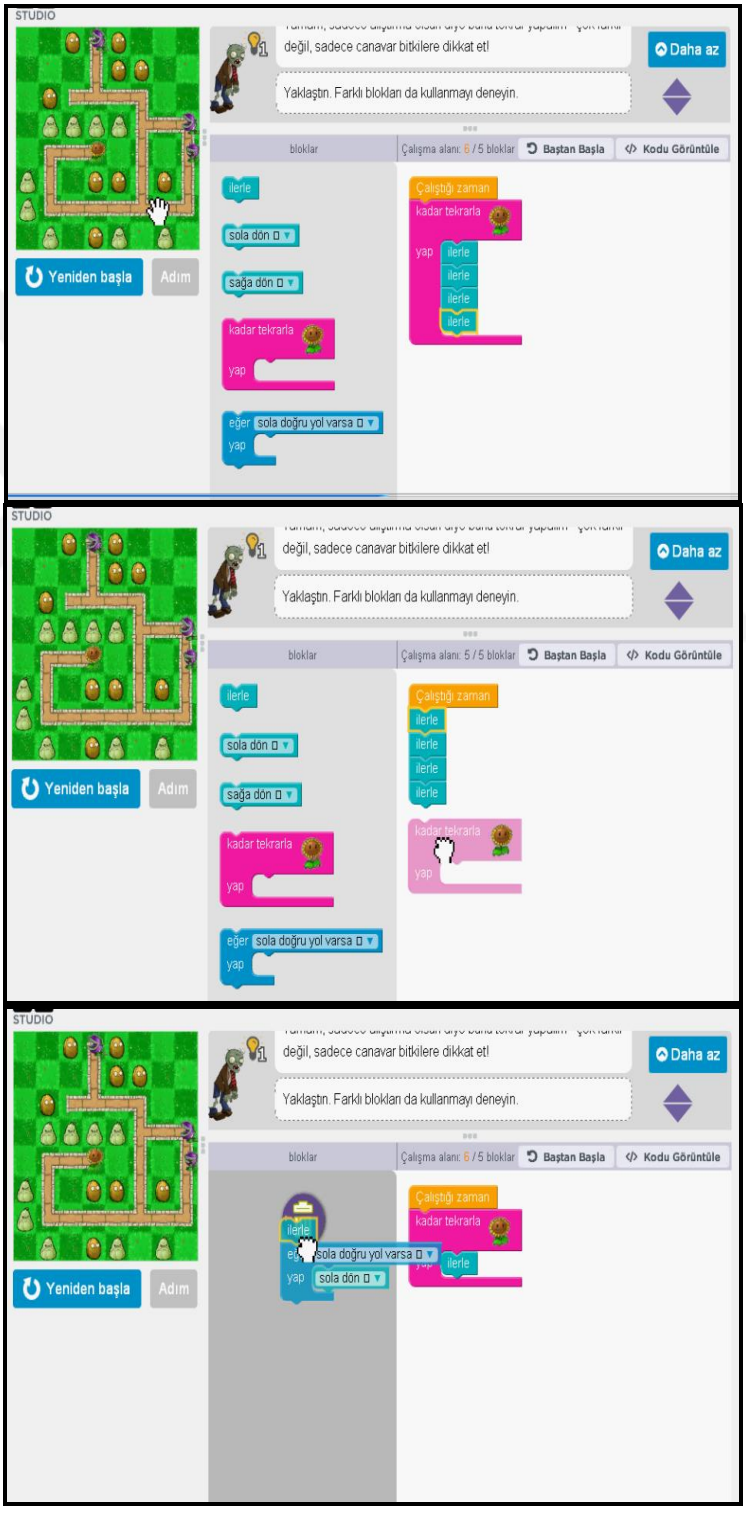
Suheda YILDIZ
Bilişim Teknolojileri Öğretmeni

Ek 3. Ö8'in Sahne6'ya ait Odaklanma, Sınama ve Sonuçlandırma Süreçleri

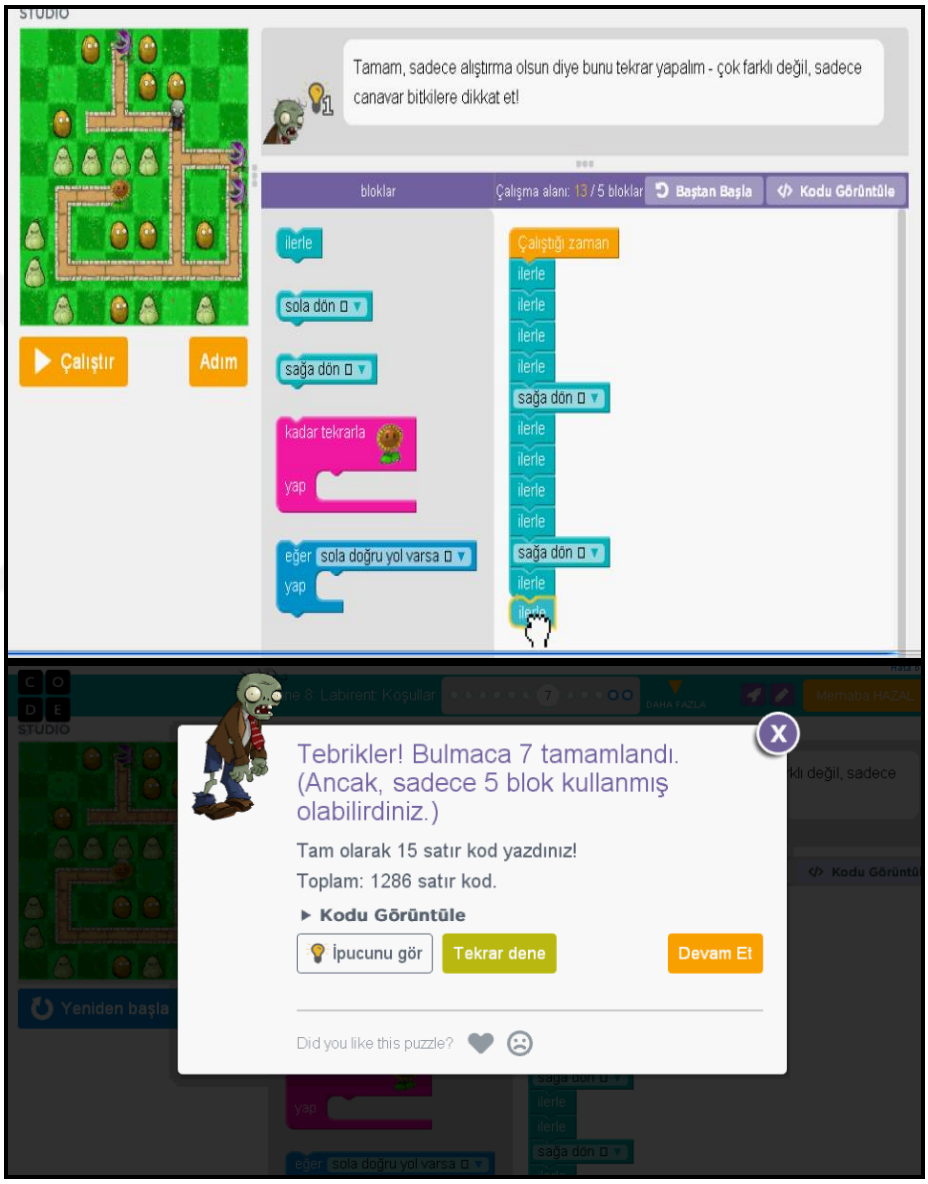
- Odaklanma aşamasında öğrencinin kendisine verilen sahnede isteneni anlama amaçlı *problem sahnesini inceleme, ekrandakileri sayma, önce ipucunu okuma* gibi davranışlar sergilediği ekran kayıtlarında incelenen fare hareketlerinde görülmektedir.



- Sınama aşamasında öncelikle *çözüme uygun blok seçimi, akış_kod kuma* ardından çözüme ulaşılamayınca *sıralamayı değiştirme, zorlanınca ipucundan yararlanma, blokları değiştirme davranışları* ekran kayıtlarında incelenen süreçte öğrencinin bloklar üzerinde çalışmasından anlaşılmaktadır.



- Sonuçlandırma aşamasında öğrencinin sahnede kullanması beklenen programlama yapısı ile görevi tamamlayamadığı ve bu noktada belirlenen en fazla blok kullanım sayısını da aşarak yeni bir akış oluşturduğu *anlamadan çözüme ulaşma* davranışına, buna bağlı olarakta kendisine verilen uyarıyı önemsemeyerek *yanlış çözümle bölüm geçme* davranışını sergilediği görülmüştür.



9. ÖZGEÇMİŞ VE İLETİŞİM BİLGİLERİ

26.07.1991 tarihinde Üsküdar / İstanbul'da doğdu. Trabzon 100. Yıl İlköğretim okulunda 8 yıllık temel eğitimini tamamlayan araştırmacı, 2009 yılında Trabzon Tevfik Serdar Anadolu Lisesi'nde lise öğrenimini tamamladı ve 2010 yılında Karadeniz Teknik Üniversitesi Eğitim Bilimleri Fakültesi Bilgisayar ve Öğretim Teknolojileri Öğretmenliği bölümünü kazandı. 2014 yılında lisans öğrenimini bölüm birincisi olarak tamamlayan araştırmacı aynı yıl Karadeniz Teknik Üniversitesi Eğitim Bilimleri Enstitüsü Bilgisayar ve Öğretim Teknolojileri Eğitimi Ana Bilim Dalı Tezli Yüksek Lisans programına kabul edildi. Mezuniyetinin hemen ardından aynı yıl Milli Eğitim Bakanlığına bağlı bir ortaokula atanınca bir sene lisans üstü eğitimine ara veren araştırmacı, bu aranın ardından hazırlık sınıfından muaf olarak, 2015/2016 Güz Döneminde Karadeniz Teknik Üniversitesi Eğitim Bilimleri Enstitüsü Bilgisayar ve Öğretim Teknolojileri Eğitimi Ana Bilim Dalı Tezli Yüksek Lisans eğitimine başladı. Halen Rize ilinde Bilişim Teknolojileri Öğretmenliği görevine devam eden araştırmacı orta derecede İngilizce bilmektedir.

İLETİŞİM BİLGİLERİ

Ad-Soyad : Suheda YILDIZ

E-Posta : suhedayildiz.b@gmail.com

Tel : 5516088870