Ahi Evran Üniversitesi
Kırşehir Eğitim Fakültesi Dergisi

ISSN: 2147 - 1037

http://kefad.ahievran.edu.tr

Programlama Öğreniminde Görselleştirmenin Etkisi: Alice*

Ebru Solmaz
Tolga Güyer

DOI:10.29299/kefad.2019.20.03.011 Makale Bilgileri

Yükleme:25/03/2019 Düzeltme:20/07/2019 Kabul: 10/10/2019

Özet

Yıllardır süregelen bir problem olarak eğitimcilerin ve araştırmacıların karşısına çıkan programlama derslerindeki yüksek dersten kalma oranları, araştırmacıları farklı çözüm yolları üretmeye yöneltmektedir. Çeşitli çalışmalarda problemin sebeplerini ortadan kaldırmaya yönelik, web tabanlı araçlar, oyunlar, iki ve üç boyutlu programlar, arayüz yazılımları gibi bir takım araçlar geliştirildiği ve kullanıldığı görülmektedir. Bu araçların birçoğunun programlama öğretiminde gerekli olan becerilerin (soyut düşünme, eleştirel düşünme, algoritmik düşünme, problem çözme gibi) kazandırılmasını kolaylaştırmaya yönelik soyut kavramları somutlaştırmaya yarayan görselleştirme yazılımları olduğu açıktır. Tanınmış görselleştirme araçlarından biri Alice'dir. Alice, nesne yönelimli ve 3 boyutlu etkileşimli görsel destekli bir programlama ortamıdır. Bu çalışmada; programlamaya giriş dersindeki düşük başarı oranlarından hareketle öğrencilerin söz konusu becerileri geliştirmelerine yardımcı olacak Alice ortamının PHP programlama dilinin öğretimi için kullanımı değerlendirilmiştir. Deney ve kontrol gruplarının oluşturulduğu araştırmada deney grubunda Php programlama dili Alice yardımı ile öğretilirken, kontrol grubunda herhangi bir yardımcı araç kullanmadan öğretim yapılmıştır. Araştırma; deney grubunda 20, kontrol grubunda 19 kişi olmak üzere toplam 39 lisans öğrencisi ile gerçekleştirilmiştir. Süreç içinde ALICE ortamının; öğrencilerin eleştirel düşünme ve problem çözme becerileri ile üstbilişsel farkındalık düzeylerine etkisi incelenmiş, araştırma sonucunda elde edilen verilerin analizi ile Alice ortamının bu 3 değişken üzerinde anlamlı bir fark yaratmadığı ortaya çıkmıştır. Bulgular, benzer araştırma sonuçları ile karşılaştırılarak irdelenmiş ve gelecek araştırmalar için çeşitli öneriler verilmiştir.

Anahtar Kelimeler: Alice, Programlama öğretimi, Eleştirel düşünme, Problem çözme, Üstbilişsel farkındalık

Sorumlu Yazar : Ebru Solmaz, Dr., Gazi Üniversitesi, Türkiye, ebrusolmaz@gazi.edu.tr, 0000-0003-4893-450X
Tolga Güyer, Prof.Dr., Gazi Üniversitesi, Türkiye, tguyer@gmail.com, 0000-0001-9175-5043
*Bu çalışma, Ebru SOLMAZ'ın Tolga GÜYER danışmanlığında hazırladığı "Programlama dili öğretiminde Alice yazılımının ders başarısı, eleştirel düşünme ve problem çözme becerileri ile üstbilişsel farkındalık düzeyine etkisi" başlıklı tezden üretilmiştir.

Atıf için: Solmaz, E. ve Güyer, T. (2019). Programlama öğreniminde görselleştirmenin etkisi: Alice. *Kırşehir Eğitim Fakültesi Dergisi, 20*(3), 1376-1416.

Giriş

Günümüzde insanlar bilgisayarı herhangi bir bilgiye ulaşmak, alışveriş yapmak ve fatura ödemek gibi yalnızca temel düzeyde bilgisayar becerisi gerektiren işler için değil; web sitesi hazırlamak, belirli işlemleri gerçekleştiren uygulamalar geliştirmek gibi programlama bilgisi gerektiren işler için de kullanmaktadır. Bu durum bilgisayar kullanıcıları arasında programlama konusuna olan ilgiyi arttırmış, insanlar herhangi bir programlama dilini öğrenmek için özel kurslara gitmeye başlamıştır. Ayrıca başta mühendislik gibi teknik alanlar olmak üzere pek çok sektörde çalışabilmek için programlama becerisine sahip olma gerekliliği programlama öğreniminin ve öğretiminin önemini arttırmaktadır. Örneğin son yıllarda programlamayı öğretmek için organizasyonlar ve kampanyalar düzenlendiği dikkat çekmektedir. Bunlardan biri Kodlama Saati (Hour of Code) kampanyasıdır. Code.org tarafından organize edilen, halka açık, kar amacı gütmeyen bu kampanya, Bilgisayar Bilimleri Haftası boyunca (3-9 Aralık) herkesin programlamanın temellerini öğrenmesini amaçlamaktadır. Code.org; 2013 yılında piyasaya sürülen, Bill Gates, Mark Zuckerberg gibi tanınmış kişiler ile Google, Apple gibi şirketlerin desteklediği bir platform olup, öğrencilerin sürükle-bırak yöntemiyle programlamayı öğrenmelerini sağlamaktadır (Kalelioğlu, 2015; Partovi ve Sahami, 2013; Wilson, 2015).

Programlama Öğrenimi ve Yaşanan Zorluklar

Araştırmacılar ilk programlama dilini öğrenmenin genellikle zor bir süreç olduğunu ifade etmektedir (Shneiderman ve Mayer, 1979; Hansen ve Kristensen, 2008; Adán-Coello, Tobar, Faria, Menezes ve Freitas, 2011). Bunun en büyük nedeni programlamanın, öğrenilmesinde ve öğretilmesinde çaba ve özel bir yaklaşım gerektiren karmaşık bir konu olması ile programlama hakkında bilgi edinmek ve edinilen bu bilgiyi geliştirmenin hayli karmaşık bir süreç olmasıdır (Rogalski ve Samurçay, 1990; Gomes ve Mendes, 2007). Bir öğrenci, iyi bir programcı olmak için, programlama dillerinin söz dizimini bilmenin ötesinde bir seri beceri edinmek zorundadır. Bunlar; çeşitli bilişsel aktiviteler, program tasarımı ile ilgili zihinsel betimlemeler, programı anlama, yazılan programla ilgili değişiklik yapma ve hata ayıklama, kavramsal bilgileri yapılandırma ve basit işlemleri yapma (döngüler, koşul cümleleri gibi) gibi becerileri içermektedir. Bu beceriler; soyutlama, soyut düşünme, algoritmik düşünme, problem çözme, genelleme, transfer, üstbiliş ve eleştirel düşünme olarak da ifade edilmektedir (Rogalski ve Samurçay, 1990; Shaft, 1995; Bucci, Long ve Weide, 2001; Futschek, 2006; Gomes ve Mendes, 2007; Gundurao, Manjunath ve Nachappa, 2010). Birçok beceri gerektiren bu karmaşık süreçten dolayı, programlama öğrenimi sırasında öğrencilerin çeşitli zorluklar yaşadığı bilinmektedir (Winslow, 1996). Programlama öğretimi boyunca öğrencilerin yüz yüze geldiği zorluklar genellikle programlama becerilerini öğrenme ile ilişkili derslerde ve direk

bu becerilere bağlı derslerde düşük başarı oranlarına ve dolayısıyla yüksek kalma oranlarına sebep olmaktadır (Sagisaka ve Watanabe, 2008; Adán-Coello ve diğerleri, 2011).

Literatürde programlama öğreniminde yaşanılan problemleri çözmeye yönelik, farklı yollarla programlama öğrenimini desteklemesi hedeflenen bazı yaklaşımlar ve araçlar önerilmektedir (Gomes ve Mendes, 2007). İlgili çalışmalarda programlama öğretiminde kullanılan yaklaşımlar şu şekilde sıralanabilir; problem tabanlı öğrenme (Kay ve diğerleri, 2000), harmanlanmış öğrenme (Mohorovicic ve Tijan, 2011; Zhao, Chis, Muntean ve Muntean, 2018), bilgisayar destekli öğrenme (Vihtonen, Alaoutinen ve Kaarna, 2001), uzaktan eğitim (Nascimento, Mendonça, Guerroro ve Figueiredo, 2010), oyun tabanlı öğrenme (Zhao ve diğerleri, 2018). Aynı şekilde konu ile ilgili incelenen araştırmalarda kullanılmış olan araçlar ise; web tabanlı araçlar (Wiki, Weblog, Web sitesi vb.) (Cervesato, 2008; Karsten, Kaparti ve Roth, 2005; Radosevic, Orehovacki ve Lovrencic, 2009; Kosurkar, Zade, Tikas, Prasad ve Sure, 2017), oyunlar (Doherty ve Kumar, 2009; Zapušek ve Rugelj, 2013; Ibrahim ve diğerleri, 2018), oyun motorları (Hernandez ve diğerleri, 2010), robotlar ve robotik (Lin ve Kuo, 2010; Costa, Aparicio ve Cordeiro, 2012), iki boyutlu programlar (Maloney ve diğerleri, 2004; Gallant ve Mahmoud, 2008; Kölling, 2008; Ouahbi, I., Kaddari, Darhmaoui, Elachqar ve Lahmine, 2015), üç boyutlu programlar (Cooper, Dann ve Pausch, 2000a; Cooper ve diğerleri, 2000b; Wang ve diğerleri, 2009), arayüz yazılımları (Baldwin ve Kuljis, 2001), arttırılmış gerçeklik ortamları (Dass, Kim, Ford, Agarwal ve Chau, 2018) şeklinde listelenebilir. Araştırmalarda araçların kullanımının pozitif sonuçları rapor edilmiş olmasına rağmen, bu araçların hiçbirinin genel bir kullanımı bulunmadığı söylenebilir.

Programlama Öğreniminde Görselleştirme

Bilgisayar bilimindeki araştırmalar programlama öğretiminde öğrencilerin genellikle soyut düşünmeyi uygularken zorlandıklarını göstermektedir (Berge, Borge, Fjuk, Kaasboll ve Samuelsen, 2003; Bucci ve diğerleri, 2001). Programlama öğrenme sürecindeki problemler; öğrenciler daha programlama öğretiminin ilk başında iken, başka bir deyişle soyut program yapılarını öğrenirken ve uygularken ya da somut problemleri çözmek için algoritma oluştururken başlamaktadır (Gomes ve Mendes, 2007). Öğrenciler bilgisayarın bir program kodlarını çalıştırırken gerçekten ne yaptığını ve bilgisayar hafızasında eş zamanlı olarak ne olduğunu anlamakta zorlanmaktadır (Miyadera, Kurasawa, Nakamura, Yonezawa ve Yokoyama, 2007). Bu problemleri çözmek için araştırmacılar tarafından çeşitli görsel programlama ortamları geliştirilmiştir. Bu araçlar öğrencilere program kodlarını ve bir programın nasıl çalıştığını daha iyi anlamaları konusunda yardımcı olmak için kullanılmaktadır (Tekdal, 2013). Aynı zamanda bu ortamlar grafiksel gösterimler (notation) kullanarak öğrenenlerin program yazabilmelerini sağlamaktadır (Ben—Ari, 2013). Programlamada görsel öğeler kullanmak tanıtılan yapıların anlamlarının anlaşılmasını sağlamakta, kavram

yanılgılarının oluşmasını önlemekte; görsellik, keşfedici öğrenme ve problem çözme için bir geribildirim sağlamaktadır (Brusilovsky, Kouchnirenko, Miller ve Tomek, 1994).

Alanyazında karşımıza çıkan ilk görsel programlama çevrelerinden birisi Agentsheetsdir (Repenning ve Sumner, 1995) . Daha sonra akademik projeler tarafından geliştirilen Greenfoot, Alice ve Scratch gibi görselleştirme araçları literatüre dahil olmuştur. Bunlara ek olarak Jeroo (Sanders ve Dorn, 2003), RAPTOR (Carlisle, Wilson, Humphries ve Hardfield, 2005), Jype (Helminen ve Malmi, 2010), cMinds (Tsalapatas, Heidmann, Alimisi ve Houstis, 2012) gibi farklı görsel programlama çevrelerinin kullanıldığı araştırmalar da yer almaktadır.

Bu araçlar arasında 3 boyutlu bir ortam sunması özelliği ile dikkat çeken Alice yazılımının, özellikle temel programlama derslerindeki etkisini araştıran önceki çalışmalarda (Cliburn, 2008; Moskal, Lurie, Cooper, 2004; Wang ve diğerleri, 2009) ortamın kullanıldığı derslerde öğrencilerin oldukça pozitif tecrübelere sahibi oldukları ve bu deneyimlerin programlamaya olan ilgilerini arttırdığı bulunmuştur. Ayrıca öğrencilerin Alice programının kullanılmasıyla programlama becerileri ile ilgili güvenlerinin arttığı (Cooper ve diğerleri, 2003; Howard, Evans, Courte ve Bishop-Clark, 2006), temel programlama kavramlarını anladıkları, algoritmalar ve Alice hikayeleri arasındaki ilişkiyi kavradıkları, eğlendikleri ortaya çıkmıştır (Howard ve diğerleri, 2006). Moskal ve diğerleri (2004) tarafından yapılan çalışmada Bilgisayar Bilimleri 1 (CS1) dersinde elde edilen veriler genel olarak Alice programının öğrencilerin performansını geliştirdiğini göstermiştir. Ölçme araçlarından elde edilen veriler öğrencilerin CS1 dersindeki performanslarını, bilgisayar bilimi ile ilgili akılda kalıcılığı ve öğrencilerin bilgisayar bilimine yönelik tutumlarını geliştirmek konusunda Alice ortamının dersin etkililiğini desteklediğini belirtmektedir. Benzer şekilde Price (2003) tarafından yapılan çalışmada CS1 dersinde Alice programının kullanılmasının nesneleri ve sınıfları tanımlama ve kullanma konusunda öğrenci performansını anlamlı bir şekilde etkilediği belirtilmiştir. Cooper ve diğerleri (2000a, 2000b) tarafından yapılan iki çalışmada ise gözlem yoluyla elde edilen veriler sonucunda öğrencilerin Alice programındaki nesneleri rahatça kullanabildikleri ve nesnelere metotları rahatlıkla uyguladıkları gözlenmiştir. Gözlem sonuçlarına göre öğrenciler yazdıkları programın nasıl çalıştığını hemen izleyebilmekte, programlarında yanlış giden noktaları görebilmekte ve kolaylıkla hataları bulabilmekte ve onları düzeltebilmektedir. Ayrıca öğrencilerin Alice'yi eğlenceli buldukları, üç boyutlu canlandırılmış dünyalar yaratmayı sevdikleri, programda beklenenden daha çok zaman geçirdikleri gözlenmiştir. Wang ve diğerleri (2009) ise Alice programı ve C++ programlama dilini karşılaştırmış, öğrencilerin özellikle tekrar yapılarında (for yapısı gibi) C++ programlama dilinde zorlandığı ve sıkıldığı görülmüş, öğrenciler alice programını öğrenmenin C++ programlama dilini öğrenmekten daha kolay olduğunu, Alice kullanmanın C++ dilini öğrenmeyi kolaylaştırdığını, öğrenirken eğlendiklerini ve zevk aldıklarını ifade etmişlerdir. Çalışmada Alice grubunun anlamlı bir

şekilde daha iyi öğrendiği ve Alice programının öğrencilerin temel programlama kavramlarını anlamalarını kolaylaştırmak konusunda daha etkili göründüğü ortaya çıkmıştır.

Bu araştırmada üniversite düzeyindeki programlamaya giriş dersinde ders tekrarı yaşayan öğrenci sayısının yüksek olmasından yola çıkılarak öğrencilerin programlama öğrenimi sırasında yaşadıkları problemlere yönelik bir çözüm yolu bulmak istenmiştir. Bu problemlerin en önemlilerinden biri öğrencilerin soyut kavramları somutlaştıramamalarıdır. Bu sebeple öğrencilerin temel programlama kavramlarını somutlaştırmalarını sağlayacak görsel bir ortamın yardımcı araç olarak Alice yazılımının kullanılması planlanmıştır. Alice yazılımını konu edinen çalışmalar incelendiğinde araştırmaların özellikle öğrenci performansı (başarı), kalıcılık gibi değişkenlere odaklandığı ve öğrencilerin görüşlerinin alındığı nitel çalışmaların yoğunlukta olduğu görülmektedir. Ancak ortamın; programlama becerisini etkileyen ve çeşitli ölçüm araçları ile ölçülebilen beceriler olan eleştirel düşünme, problem çözme ve üstbilişsel farkındalık becerilerine etkisinin incelendiği nicel çalışmaların çok az sayıda olduğu görülmektedir. Bununla birlikte Cooper, Dann ve Pausch (2000b) yapmış oldukları çalışmada Alice ortamının algoritmik düşünme ve problem çözme becerileri gibi yeterlilikleri desteklediğini ifade etmektedir. Bu sebeple Alice ortamının kullanılmasının öğrencilerin bu becerileri geliştirme ile ilgili problemlerini gidermek için katkı sağlayabileceği ve olumlu sonuçlar elde edilebileceği düşünülmüştür. Bu sebeple bu araştırma ile Programlamaya Giriş dersinde yardımcı araç olarak Alice ortamı kullanımının, öğrencilerin programlama öğrenimlerindeki performanslarını etkileyen ve programlama becerisi için gerekli olan eleştirel düşünme ve problem çözme becerileri ile üstbilişsel farkındalık düzeyi üzerinde etkisini incelemek amaçlanmıştır.

Yöntem

Nicel bir araştırma olarak tasarlanan çalışmada yarı deneysel desenlerden öntest-sontest kontrol gruplu desen kullanılmıştır. Deney ve kontrol grubu öğrencilerin gruplara rastgele atanması ile oluşturulmuştur.

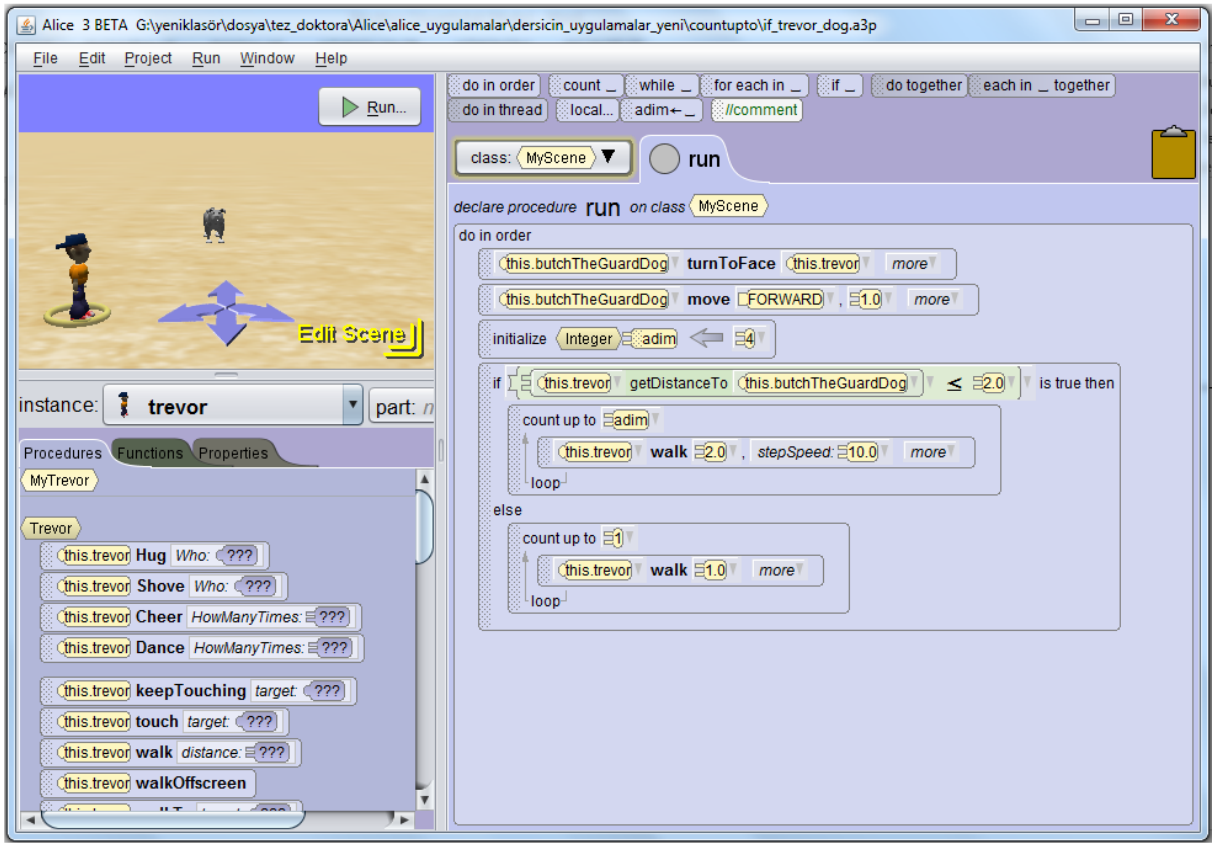
Katılımcılar

Araştırma bir devlet üniversitesinde Eğitim Fakültesi Bilgisayar ve Öğretim Teknolojileri Öğretmenliği 2.sınıf öğrencilerinden oluşan 39 kişi ile gerçekleştirilmiştir. Deney ve kontrol grubu olarak iki şubeye ayrılan öğrencilerden daha önce programlama dersi alan ve dersi alttan alanlar çalışma grubuna dahil edilmemiştir. Kontrol grubunda 8 erkek, 11 kız olmak üzere toplam 19 öğrenci, deney gurubunda 7 erkek, 13 kız olmak üzere toplam 20 öğrenci bulunmaktadır.

Öğretim Materyali

Araştırma için kullanılacak olan görselleştirme aracını belirlemek için programlamaya giriş dersinin sonraki programlama derslerine de temel oluşturduğu göz önünde tutularak, dersin belirlenmiş olan kazanımlarına uygun olacak bir ortama karar verebilmek amacıyla öncelikle çalışmalarda adı geçen ve kullanımı mümkün olan görselleştirme araçları seçilmiştir. Kurulum ve kullanım kolaylığı açısından en uygun araçların Scracth ve Alice olduğu görülmüştür. Ancak ortam seçiminde öğrenen özellikleri de önemli bir unsurdur. Bu bağlamda Scracth; alanda kullanım oranı yüksek ve tanınmış bir ortam olmasına rağmen 8-16 yaş arasındaki kullanıcılar için, Alice ise daha büyük yaştaki kullanıcılar için tasarlanmıştır (Maloney, Resnick, Rusk, Silverman ve Eastmond, 2010). Bu sebeple Alice ortamının hedef kitle için daha uygun olduğu belirlenmiştir.

Nesne yönelimli ve 3 boyutlu etkileşimli görsel destekli bir programlama ortamı olan Alice yazılımı Carnegie Mellon Üniversitesinde Randy Rausch'un yönetiminde bir araştırma grubu tarafından 3 boyutlu ortamlar oluşturmak amacıyla geliştirilmiştir. Yazılım kullanıcıların sanal dünyalar yaratmasını ve bu dünyalar içinde nesneleri canlandırmasını sağlamaktadır (Wang ve diğerleri, 2009). Dili İngilizce olan yazılım www.alice.org sitesinden ücretsiz indirilebilmekte ve popüler işletim sistemleri için sürümleri bulunmaktadır. Orijinal hali Python programlama dili üzerine kurulmuş olsa da program şu an tamamen Java tabanlı olarak çalışmaktadır. Kullanıcılar Java ya da Python programlama dillerini öğrenmeden kolaylıkla Alice ortamında program yazabilmektedir (Cooper, Dann ve Pausch, 2003).



Resim 1. Alice ortamı arayüzü

Alice ortamı kullanıcılara basit komutlar yazarak nesnelerin özelliklerini ve hareketlerini kontrol edebilme (Cooper ve diğerleri, 2000b; Wang ve diğerleri, 2009), programlarının nasıl çalıştığını anında görebilme ve program kodları ile animasyon hareketleri arasında ilişki kurabilme imkanı vermektedir (Cooper ve diğerleri, 2000b). Scratch gibi sürükle –bırak özelliğine sahip bir arayüze sahip olan Alice ortamı bu özelliği ile söz dizimi hatalarını engellemekte, böylece öğrenci kod ayıklamaktansa programlama kavramlarına odaklanmaktadır (Cliburn, 2008). Böylece öğrenciler program içinde var olduğu varsayılan nesneleri hayal etmek ve ayrıca bilgisayar dışındaki olaylar ile bilgisayar içindeki nesneler arasındaki ilişkiyi düşünmek zorunda kalmamaktadır (Berge ve diğerleri, 2003). Ayrıca Alice ortamında öğrencilerin farklı programlama dili yapılarının (fonksiyonlar, döngüler, olaylar vb.) işlevlerini canlandırılmış bir şekilde görerek programlama dilinin komutlarını anlamalarının kolaylaştığı, ortamın öğrencilerin ilgi ve katılımlarının sürdürülmesini de sağladığı belirtilmektedir (Cooper ve diğerleri, 2000b; Wang ve diğerleri, 2009).

Uygulama Süreci

Çalışmanın uygulama süreci 2013-2014 Eğitim-Öğretim Yılı Güz döneminde, Programlama Dilleri-1 dersinde, her gruba haftada beşer ders saati olmak üzere 7 hafta devam etmiştir. Programlamaya Giriş dersi olan bu derste PHP programlama dili uygulamalı olarak laboratuvar ortamında verilmektedir.

Uygulama sürecinde işlenecek konular en az 5 yıldır programlama dersi veren Programlama Dilleri-1 dersinden sorumlu öğretim elemanlarının görüşleri alınarak ve programlama öğretimi ile ilgili ders kitapları incelenerek belirlenmiş ve aşağıda Tablo 1'de gösterilmiştir;

Tablo 1. *Uygulama süresinde her hafta işlenen konu başlıkları*

Haftalar	Konu Başlıkları
1	Veri türleri • String tabanlı türler • Sayısal türler
2	Değişkenler ve Sabitler • Değişkenler • Sabitler
3	Operatörler • Aritmetik operatörler • String operatörleri • Değer atama operatörleri • Karşılaştırma Operatörleri • Mantıksal Operatörler
4	Akış Kontrol Deyimleri • Koşullu İfadeler o If o If/Else o If/Elseif/Else o Switch/Case
5	Akış Kontrol Deyimleri • Döngüler o Sayaçlı Döngüler ▪ For Döngüsü ▪ Foreach Döngüsü o Koşullu Döngüler ▪ While Döngüsü ▪ Do…While Döngüsü
6	Diziler • Tek boyutlu diziler • Çok boyutlu diziler
7	Fonksiyonlar • Kullanıcı Tanımlı Fonksiyonlar- o Parametresiz fonksiyonlar o Parametreli fonksiyonlar • Dahili (içsel) Fonksiyonlar

Uygulama sürecinde öğretim tasarımı için Gagne'nin 9 aşamalı modeli kullanılmıştır. Modeli oluşturan; dikkat çekme, hedeflerden haberdar etme, önbilgileri hatırlatma, materyali sunma, rehberlik etme, performansı ortaya çıkarma, geribildirim sağlama, performansı değerlendirme, kalıcılığı ve performansı arttırma basamakları; Gagne'ye göre, öğrenme sürecindeki içsel olaylar için dışarıdan destek sağlamanın potansiyel bir yolu olarak düşünülmelidir (Reigeluth, 1987). Ayrıca

modeldeki aşamalar bir derste ya da müfredatta seçilen öğretime göre farklı şekillerde düzenlenebilmektedir (Driscoll, 1994).

Bu model ile Gagne (1985), birbiri üstüne inşa edilen bir öğrenme hiyerarşisi tanımlamıştır. Ona göre, öğrenme entelektüel beceriler gerektirmekte, düşük seviyedeki beceriler üst seviyedeki becerilere alt yapı hazırlamaktadır. Ayrıca belirli bir beceriyi öğretmek için, ön beceriler tanımlanmalı ve öğrencilerin onlara sahip olduğundan emin olunmalıdır. Bu çalışmada da birbiri üzerine eklenen, basit becerilerden karmaşık becerilere doğru giden bir beceri olan programlama becerisi kazandırmak amaçlanmaktadır. Ayrıca Şendağ ve Başer (2013), Gagne'nin öğretim teorisini giriş düzeyindeki bilgilerin verilmesinde yararlanılabilecek bir yöntem olarak ifade etmiştir. Bu çalışma giriş dersi niteliğinde olan Programlama Dilleri- I dersinde yapılmakta ve çalışmanın amacı Alice ortamının temel programlama kavramlarını kavrama üzerindeki etkisini ortaya koymaktır. Bu sebeple derste hedeflenen bilgi ve becerilerin kazandırılması için uygulama sürecinde öğretim tasarımı bu model temel alınarak planlanmış ve gerçekleştirilmiştir. Her hafta uygulanacak öğretim, modelin her bir basamağı göz önünde bulundurularak ayrıntılı bir şekilde düzenlenmiştir.

Ders içeriğinin öğretimi sırasında deney grubunda yardımcı araç olarak Alice programı kullanılırken, kontrol grubunda herhangi bir yardımcı araç kullanmadan PHP programlama dili öğretilmiştir. Deney grubunda her haftanın konusu öncelikle Alice programı üzerinden gösterilmiş, dersin ilk yarısı Alice programı üzerinde uygulama yapılarak değerlendirilmiştir. Dersin ikinci yarısında ise, Alice programı üzerinden öğrenilen bilgiler PHP programlama dilinde tekrar edilmiş, öğrencilerin Alice programında öğrendiklerini PHP programlama diline transfer etmeleri istenmiştir. Bu esnada Alice ve Php programlama dili arasında kodlarda ya da gösterimlerde var olan farklılıklar konusunda öğrenciler bilgilendirilmiştir. Ders süresince öğrencilere uygulamalar yaptırılmış, öğrencilerden haftanın konusu ile ilgili basit program parçaları oluşturmaları istenmiştir.

Kontrol grubunda ise, dersin ilk yarısında ilgili haftanın konusuna ait temel kavram ya da kavramlar öğretim elemanı tarafından, herhangi bir yardımcı araç kullanmadan, Not defteri üzerinde doğrudan Php programlama dili kullanılarak yazılan program parçaları ile tanıtılmış, dersin ikinci yarısında öğrencilerin konu ile ilgili farklı uygulamalar yaparak öğrenilen kavram ya da kavramları uygulamaları ve pekiştirmeleri sağlanmıştır. Uygulama süreci Tablo 2'de gösterilmektedir.

Tablo 2. *Uygulama süreci*

	Ön test	İşlem	Son test
Deney Grubu	Eleştirel Düşünme Eğilimi Ölçeği Problem Çözme Envanteri Üstbilişsel Farkındalık Envanteri	Alice + Php	Eleştirel Düşünme Eğilimi Ölçeği Problem Çözme Envanteri Üstbilişsel Farkındalık Envanteri
Kontrol Grubu		Php	

Deney grubundaki öğrenciler temel kavramları Alice programında öğrenmiş, daha sonra Alice programında öğrenilen kavram ya da kavramların tekrar anlatılmasına gerek kalmadan Php programlama dilinde uygulamalara geçilmiştir. Kontrol grubunda ise dersin ilk yarısı temel kavramlara ayrılmış, dersin ikinci yarısı uygulamalar yapılmıştır. Bu sebeple deney grubunda dersler iki ayrı yazılım üzerinden işlense de bir zaman kaybı olmamış, iki grupta da her hafta aynı konular işlenmiştir. Ayrıca her hafta işlenen konu ya da konular ve ders sürecinde yapılan etkinlikler her iki grupta da aynıdır. Böylece gruplar arasındaki eşitlik sağlanmıştır.

Veri Toplama Araçları

Araştırmada eleştirel düşünme becerisini ölçmek için 1990 yılında Amerikan Felsefe Derneği'nin düzenlediği Delphi projesi sonucunda geliştirilen ve Kökdemir (2003) tarafından Türkçe'ye uyarlanan 51 maddelik California Eleştirel Düşünme Eğilimi Ölçeği kullanılmıştır. Ölçeğin Cronbach alfa iç tutarlılık katsayısı 0,88, ölçeğin açıkladığı toplam varyans ise % 36,13'tür (Kökdemir, 2003).

Problem çözme becerisi değişkeni için Heppner ve Peterson (1982) tarafından geliştirilen, Şahin, Şahin ve Heppner (1993) tarafından Türkçe'ye uyarlanan 35 maddelik Problem Çözme Envanteri kullanılmıştır. Ölçeğin iç tutarlılık katsayısı ,88, güvenilirlik katsayısı ,81 olarak bulunmuştur (Şahin, Şahin ve Heppner, 1993).

Üstbilişsel farkındalık değişkeni için ise Schraw ve Dennison (1994) tarafından geliştirilen ve Akın, Abacı ve Çetin (2007) tarafından uyarlanan 52 maddelik Üstbilişsel Farkındalık Envanteri kullanılmıştır. Ölçeğin iç tutarlılık katsayısı ,95, güvenilirlik katsayısı ,95 olarak bulunmuştur (Akın, Abacı ve Çetin, 2007).

Veri Toplama

7 hafta süren uygulama sürecinin başında öğrencilerin eleştirel düşünme, problem çözme ve üstbilişsel farkındalık düzeylerini belirlemek için öğrencilerin 3 veri toplama aracını doldurmaları sağlanmıştır. 7 hafta boyunca programlamanın temel kavramları ile ilgili öğretim sürecinde öğrencilerin ilgili 3 beceriyi kullanmalarını gerektiren uygulamalar yapıldıktan sonra sürecin sonunda aynı veri toplama araçları yeniden öğrencilere uygulanmıştır.

Veri Analizi

Yapılan uygulama sonunda elde edilen nicel verilerin analizi SPSS paket programı ile gerçekleştirilmiş, analizlerin anlamlılık düzeyi ,05 olarak kabul edilmiştir.

Borg ve Gall (1979; aktaran Cohen, Manion ve Morrison, 2007) deneysel araştırmalarda örneklem büyüklüğünün 15'den az olmaması gerektiğini söylemektedir. Benzer şekilde Büyüköztürk (2008), literatürde araştırmada kullanılan alt grupların her birinin büyüklüğünün 15 ve daha yüksek olduğu durumlarda parametrik testlerin kullanılmasının analizde hesaplanacak p anlamlılık düzeyinde önemli bir sapmaya yol açmadığını belirten incelemelerin bulunduğunu ifade etmektedir. Ancak parametrik testlerin kullanılabilmesi için verilerin normal dağılım göstermesi gerektiği için öncelikle eleştirel düşünme eğilimi, problem çözme becerileri ve üstbilişsel farkındalık düzeyleri puanlarının normal dağılım gösterip göstermediğini belirlemek için Kolmogorov-Smirnov Testi uygulanmıştır (Tablo 3).

Tablo 3. *Araştırma verilerinin dağılım değerleri*

		N	P
Eleştirel Düşünme Eğilimi Öntest Puanları	Deney	20	,20
	Kontrol	19	,20
Eleştirel Düşünme Eğilimi Sontest Puanları	Deney	20	,20
	Kontrol	19	,20
Problem Çözme Becerileri Öntest Puanları	Deney	20	,20
	Kontrol	19	,173
Problem Çözme Becerileri Sontest Puanları	Deney	20	,171
	Kontrol	19	,20
Üstbilişsel Farkındalık Öntest Puanları	Deney	20	,20
	Kontrol	19	,20
Üstbilişsel Farkındalık Sontest Puanları	Deney	20	,20
	Kontrol	19	,20

Değişkenlere ait verilerin anlamlılık değerleri (p) .05 anlamlılık düzeyinden büyük olduğu için verilerin normal dağılım gösterdiği görülmektedir. Buna göre deney ve kontrol gruplarının söz konusu 3 değişkene ilişkin puanları arasında anlamlı fark olup olmadığını belirlemek amacıyla iki faktörlü varyans analizi (ANOVA) kullanılması kararlaştırılmıştır. İki faktörlü Anova araştırmacıların sadece her bağımsız değişkenin etkisini değil aynı zamanda iki bağımsız değişkenin birbirleri üzerindeki etkileşim etkilerini de incelemektedir (Cohen, Manion ve Morrison, 2007). Bu araştırmada da deney ve kontrol gruplarından elde edilen veriler karşılaştırılmakta, bu sebeple yalnızca grup içi değil, gruplar arası etkiler de önem kazanmaktadır.

Bulgular

Eleştirel Düşünme Becerisi Puanlarına Ait Bulgular

Öğrencilerin deney öncesi ve sonrasındaki eleştirel düşünme eğilimi puanları arasında anlamlı bir farklılık olup olmadığını belirlemek için uygulanan iki faktörlü ANOVA sonuçları Tablo 4'te verilmiştir.

Tablo 4. *Eleştirel düşünme eğilimi ölçeği öntest-sontest puanlarının ANOVA sonuçları*

Varyansın Kaynağı		KT	sd	KO	F	P
Denekleraras	24521,75		38			
Grup (Deney/Kontrol)		564,825	1	564,825	0,872	,356
Hata		23956,925	37	647,484		
Denekleriçi	9925,952		39			
Ölçüm (Öntest-Sontest)		654,867	1	654,867	2,676	,110
Grup*Ölçüm		**214,847**	**1**	**214,847**	**0,878**	**,355**
Hata		9056,238	37	244,763		
Toplam	34447,702		77			

Tablo 4'e göre deney ve kontrol grubundaki öğrencilerin eleştirel düşünme eğilimlerinin deney öncesinden sonrasına anlamlı farklılık göstermediği, yani Alice ortamı kullanımının eleştirel düşünme eğilimi üzerinde anlamlı bir etki yaratmadığı bulunmuştur, F(1,37)= ,878, p>0,05. Ayrıca elde edilen bulgular deney ve kontrol gruplarının eleştirel düşünme eğilimi puanları arasında ve öntest-sontest sonuçları arasında da anlamlı bir fark olmadığını göstermektedir. Bu durum Alice ortamını kullanarak yapılan programlama öğretimi ile Alice ortamını kullanılmadan yapılan programlama öğretiminin eleştirel düşünme eğilimi üzerinde benzer etkileri olduğunu göstermektedir.

Problem Çözme Becerisi Puanlarına Ait Bulgular

Öğrencilerin problem çözme envanteri puanlarında deney öncesi ve sonrasında gözlenen değişmelerin anlamlı bir farklılık gösterip göstermediğine ilişkin iki faktörlü ANOVA sonuçları Tablo 3'te verilmiştir.

Tablo 5. *Problem çözme ölçeği öntest-sontest puanlarının ANOVA sonuçları*

Varyansın Kaynağı		KT	Sd	KO	F	P
Denekleraras	14254,949		38			
Grup (Deney/Kontrol)		110,286	1	110,286	,289	,594
Hata		14114,663	37	381,477		
Denekleriçi	3654,527		39			
Ölçüm (Öntest-Sontest)		574,912	1	574,912	,057	,012
Grup*Ölçüm		**65,373**	**1**	**65,373**	**,802**	**,376**
Hata		3014,242	37	81,466		
Toplam	17909,476		77			

Tablo 5 incelendiğinde deney ve kontrol grubundaki öğrencilerin problem çözme becerilerinin deney öncesinden sonrasına anlamlı farklılık göstermediği, yani Alice ortamı kullanımının problem çözme becerileri üzerinde anlamlı bir etki yaratmadığı bulunmuştur, F(1,37)= ,802, p>0,05. Ayrıca analiz bulgularına göre deney ve kontrol gruplarının problem çözme beceri puanları arasında ve grupların

öntest-sontest puanları arasında da anlamlı bir farklılık ortaya çıkmamıştır. Sonuç olarak programlama öğreniminde Alice ortamının kullanılması ile Alice ortamının kullanılmamasının problem çözme becerileri üzerinde farklı etkilerinin olmadığı söylenebilir.

Üstbilişsel Farkındalık Düzeyi Puanlarına Ait Bulgular

Öğrencilerin deney öncesinde ve sonrasında gözlenen üst bilişsel farkındalık envanteri puanları arasında anlamlı bir farklılık bulup bulunmadığını belirlemek amacıyla yapılan iki faktörlü ANOVA sonuçları Tablo 6'da verilmiştir.

Tablo 6. *Üstbilişsel farkındalık ölçeği öntest-sontest puanlarının ANOVA sonuçları*

Varyansın Kaynağı	KT	Sd	KO	F	p
Denekler arası	49254,718	38			
Grup (Deney/Kontrol)	82,632	1	82,632	0,062	,804
Hata	49172,086	37	1328,975		
Denek içi	4892,37	39			
Ölçüm (Öntest-Sontest)	280,421	1	280,421	2,335	,135
Grup*Ölçüm	168,832	1	168,832	1,406	,243
Hata	4443,117	37	120,084		
Toplam	54147,088	77			

Tablo 6'ya göre deney ve kontrol grubundaki öğrencilerin üst bilişsel farkındalık düzeylerinin deney öncesinden sonrasına anlamlı farklılık göstermediği, yani Alice ortamı kullanımının üst bilişsel farkındalık düzeyi üzerinde anlamlı bir etki yaratmadığı bulunmuştur, $F(1,37)=1,406$, $p>0,05$. Aynı şekilde araştırma bulguları değerlendirildiğinde deney ve kontrol gruplarının üstbilişsel farkındalık düzey puanları arasında ve grupların öntest-sontest puanları arasında da anlamlı bir fark olmadığı görülmektedir. Bu sonuçlar Alice ortamı kullanarak programlama öğrenimi ile bu programı kullanmadan yapılan programlama öğreniminin üst bilişsel farkındalık düzeyi üzerinde benzer etkilere sahip olduğunu ortaya koymaktadır.

Tartışma, Sonuç ve Öneriler

Alice 3 boyutlu görsel yazılım ortamının Programlamaya Giriş Dersi'ndeki etkisinin araştırıldığı araştırmanın verilerinden alınan sonuçlar ortamın eleştirel düşünme, problem çözme ve üst bilişsel farkındalık değişkenleri üzerinde anlamlı bir etkisi olmadığını göstermiştir. Alice ortamının bu değişkenler üzerindeki etkisinin incelendiği araştırma sayısı çok sınırlı olduğu için elde edilen bulgular, benzer ortamlarla ya da programlama dilleriyle yapılan araştırma sonuçları ile karşılaştırılmıştır.

Eleştirel düşünme becerisi ele alındığında bu değişkeni konu edinen bir çalışma ortaya koyan Grant (2003) çalışmasında geleneksel (prosedür tabanlı) programlama dersi ile nesne tabanlı programlama dersindeki öğrencilerin eleştirel düşünme eğilimi puanlarını karşılaştırmış, hem iki gurubun puanları

arasında, hem de öntest ve sontest puanları arasında anlamlı bir fark bulunmadığını ifade etmiştir. Bizim çalışmamızda da prosedür yönelimli bir programlama dili olan PHP öğreniminin bir grupta geleneksel yolla, diğer grupta nesne yönelimli bir ortam olan Alice yardımı ile gerçekleştirildiği düşünüldüğünde Grant (2003) tarafından yapılan çalışma ile bizim araştırmamızın paralel sonuçlar ürettiği görülmektedir. Prosedür temelli programlama ile nesne tabanlı programlamanın eleştirel düşünme eğilimine etkileri arasında anlamlı bir fark bulunmazken, nesne tabanlı bir ortam olan Alice kullanımının da eleştirel düşünme eğilimi üzerinde anlamlı bir fark yaratmaması kullanılan ortamın ve dilin türünün eleştirel düşünmeye etkisinin olmadığı gibi bir yorum yapılmasına izin verebilir.

Araştırmanın diğer bir değişkeni olan problem çözme becerisi ile ilgili yapılan çalışmalar incelendiğinde sonuçların benzer olmadığı görülmektedir. Klassen (2006) çalışmasında zayıf mantıksal düşünme becerilerine sahip öğrencilerin Alice ortamını kullanarak genel problem çözme becerilerini geliştiremediklerini rapor etmektedir. Bu çalışmada da öğrencilerin problem çözme becerilerinin gelişmemesinin sebebi zayıf mantıksal düşünme becerilerine sahip olmaları olabileceği düşünülebilir. Carlisle ve diğerleri (2005) ise çalışmalarında geliştirdikleri görsel programlama ortamı ile programlama öğretmenin, problem çözme becerilerini daha geleneksel ve görsel olmayan dillerde programlama öğretmekten daha iyi geliştirdiğini ifade etmektedir. Tsalapatas ve diğerleri (2012) tarafından yapılan çalışmada ise oyun tabanlı görsel programlama eğitimi veren ve görsel bir öğrenme çevresi olan cMinds isimli ortamın problem çözme kapasitesini arttırdığı belirtilmektedir. Araştırmalarda iki görsel programlama ortamının problem çözme becerisi üzerinde olumlu etkisi ortaya koyulmuşken, Alice ortamının bu değişken üzerinde anlamlı bir fark yaratmamış olması ortamın tasarım özelliklerinden kaynaklanıyor olabilir.

Üstbilişsel farkındalık değişkeni ile ilgili araştırmalar araştırıldığında bu değişken ile ilgili çok sayıda çalışmanın bulunmadığı dikkat çekmektedir. Kiser (1989) tarafından yapılan araştırmada problem çözme içeriklerinin bulunduğu logo öğretimi ile sadece logo öğretiminin öğrencilerin üstbilişsel becerilerini arttırmadığı görülmüştür. Bu durum, programlama eğitiminde uygulanan yöntem ya da seçilen dil veya araçların, öğrencilerin üstbilişsel becerileri üzerinde herhangi bir etki yaratmadığı şeklinde yorumlanabilir.

Sonuç olarak eleştirel düşünme ve üstbilişsel farkındalık değişkenleri üzerinde görselleştirme uygulamalarının etkili olmadığı, problem çözme becerisi üzerinde etkinin ise genel olarak ortama göre farklılık gösterdiği, özelde ise Alice ortamının bu değişken üzerinde etki yaratmadığı söylenebilir.

Yukarıda açıklanan sonuçlar temelinde gelecek araştırmalara ışık tutacak bazı araştırma ve uygulama önerilerinden bahsedilebilir. Özellikle eleştirel düşünme ve üstbilişsel farkındalık değişkenlerinin ele

alındığı çalışma sayısının azlığı dolayısıyla, bu araştırmaya benzer araştırmaların yapılması, elde edilen sonuçların desteklenmesi ya da aksinin kanıtlanmasını sağlayarak genellenebilir sonuçlar elde edilmesini sağlayabilir. Ayrıca bu çalışmada Alice ortamından PHP programlama dilinin öğretimi için yararlanılmış ve uygulama sırasında her ders, Alice ile PHP art arda kullanılmıştır. Söz konusu ders sonraki dönemlerdeki bazı derslere temel oluşturduğu ve takip edilmesi gereken bir müfredat olduğu için dersin süresinde ve yapısında herhangi bir değişiklik yapılamamıştır. Bu durum bir sınırlılık olarak ele alınmıştır. Cliburn (2008) çalışmasında, öğrenciler aynı derste Alice ortamını ve başka bir dili öğrendiği zaman, dersin hedefinin öğrenciler için belirsiz olabileceğini ifade etmektedir. Bu olasılıktan hareketle ve araştırmanın sınırlılığı kapsamında gelecek çalışmalarda, ortamın kullanım şekli değiştirilerek, bazı araştırmalarda görüldüğü gibi (Klassen, 2006; Zaccone, Cooper ve Dann, 2003), bir dönemlik ya da birkaç haftalık bir hazırlık dersi açılabilir ve bu hazırlık dersi kapsamında Alice ortamı ile programlamanın temel kavramları öğretilebilir. Sonrasında Alice ortamının mevcut programlamaya giriş dersinin konusu olan programlama dilini öğrenmeye yönelik etkisi ya da katkısının incelendiği araştırmalar tasarlanabilir. Böylece iki farklı dil yapısının art arda öğrenilmesi sebebiyle öğrencilerin yaşadığı olumsuz durum engellenmiş olacaktır.

Araştırmanın sınırlılıklarından birisi de dersin içeriğinde yer alan programlama dilinin PHP olmasıdır. Müfredat ile ilgili bir değişiklik yapılamadığı için çalışmada PHP programlama dilinin öğretimi gerçekleştirilmiştir. PHP prosedürel bir programlama dilidir. Ancak Alice nesne tabanlı bir ortam sunmaktadır. Price (2003) ile Moskal ve diğerleri (2004) tarafından yapılan çalışmalarda Alice ortamının nesne yönelimli programlama dillerinin öğreniminde etkili olduğuna dair sonuçların alınmış olması bu çalışmada anlamlı sonuçlar çıkmamasının sebebinin programlama dilinden kaynaklandığı yorumunun yapılmasına yol açabilir. Bu sebeple benzer araştırmaların farklı programlama dillerinin öğretiminin yapıldığı programlamaya giriş derslerinde tekrarlanması ve sonuçların karşılaştırılması literatüre katkı sağlayacaktır.

Araştırmada dikkat çeken bir bulgu da problem çözme becerisi ile ilgili sonuçlara göre diğer görselleştirme araçlarının söz konusu beceri üzerinde olumlu bir etkisi varken, Alice ortamının benzer sonucu vermemesidir. Bu durumun nedenleri Alice ortamının özelliklerinin ayrıntılı olarak incelendiği nitel çalışmalar ile araştırılabilir.

Kaynakça

Adán-Coello, J. M., Tobar, C. M., Faria, E. J. , Menezes, W. S. ve Freitas, R. L. (2011). Forming groups for collaborative learning of introductory computer programming based on students' programming skills and learning styles. *International Journal of Information and Communication Technology Education*, *7*(4), 34-46.

Akın, A., Abacı, R. ve Çetin, B. (2007). The validity and reliability of the Turkish version of the metacognitive awareness inventory. *Educational Sciences: Theory & Practice, 7*(2), 671-678.

Baldwin, L. P. ve Kuljis, J. (2001, January). Learning programming using program visualization techniques. Paper presented at the *34th annual Hawaii international conference on system sciences*, Maui, HI, USA.

Ben—Ari, M. (2013). Visualization of programming. İçinde Kadijevich, D. M., Angeli, C. ve Schulte, C. (Ed.), *Improving Computer Science Education (*ss. 61-74*).* New York: Routledge.

Berge, O., Borge, R.E., Fjuk, A., Kaasboll, J., ve Samuelsen, T. (2003, November). Learning object oriented programming. *Paper presented at the Norsk informatik konferanse (Norwegian informatics conference),* Oslo, Norway.

Borg, W. R. ve Gall, M. D. (1979). *Educational Research: An Introduction (third edition)*. London: Longman.

Brusilovsky, P., Kouchnirenko, A., Miller, P. ve Tomek, I. (1994, June). Teaching programming to novices: a review of approaches and tools. *Proceedings of ED-MEDIA 94--World conference on educational multimedia and hypermedia*, Vancouver, British Columbia, Canada, pp. 103-110.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A. ve Miller, P. (1997). Mini-languages: a way to learn programming principles. *Education and Information Technologies, 2*, 65-83.

Bucci, P., Long, T. J. ve Weide, B. W. (2001). Do we really teach abstraction?. *ACM SIGCSE Bulletin, 33*(1), 26-30.

Büyüköztürk, Ş. (2008). *Sosyal Bilimler için Veri Analizi El Kitabı,* 9.Baskı. Ankara: Pegem Akademi.

Carlisle, M.C., Wilson, T. A., Humphries, J. W., ve Hadfield, S. M. (2005). RAPTOR: A visual programming environment for teaching algorithmic problem solving. *ACM SIGCSE Bulletin 37*(1), 176-180.

Cervesato, I. (2008). *On teaching programming languages using a wiki*. Report, Computer Science Department, Carnegie Mellon University, USA. Retreived from http://repository.cmu.edu/compsci/2

Cliburn, D. C. (2008, October). Student opinions of Alice in CS1. *Paper Presented at the 38th Annual Frontiers in Education Conference*, Saratoga Springs, NY, USA, doi: 10.1109/FIE.2008.4720254

Cohen, L., Manion, L. ve Morrison, K. (2007). *Research Methods in Education* (Sixth Edition). London: Routledge.

Cooper, S., Dann, W. ve Pausch, R. (2000a). Alice: A 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Small Colleges, 15*(5), 107-116.

Cooper, S., Dann, W. ve Pausch, R. (2000b). Developing algorithmic thinking with Alice. In D. Colton (Ed.), *Proceedings of the 17th Information Systems Education Conference (ISECON 2000)* (pp. 506-539), Pensilvania, USA, Vol. 17.

Cooper, S., Dann, W. ve Pausch, R. (2003). Using animated 3D graphics to prepare novices for CS1. *Computer Science Education*, *13* (1), 3-30.

Costa, C. J., Aparicio, M. ve Cordeiro, C. (2012, June). A solution to support student learning of programming. *Proceedings of The Workshop on Open Source and Design of Communication* (pp. 25-29). ACM.

Dann, W., Cooper, S. ve Pausch, R. (2000, July). Making the connection: programming with animated small worlds. *Proceedings of the 5th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE 2000) (pp. 41-44), Helsinki, Finland.

Dass, N., Kim, J., Ford, S., Agarwal, S. ve Chau, D. H. P. (2018, April). Augmenting coding: augmented reality for learning programming. *Proceedings of the Sixth International Symposium of Chinese CHI (*pp: 156-159*)*, Montreal, Canada.

Doherty, L. ve Kumar, V. (2009, August). Teaching programming through games. Paper presented at *International Workshop on Technology for Education*. Bangalore, India.

Driscoll, M. (1994). Gagne's theory of instruction. *Psychology of Learning for Instruction*. Boston, MA, Allyn and Bacon, 329-358.

Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. In R. T. Mittermeir (Ed.), *Proceedings of Evolution and Perspectives: 2nd* I*nternational Conference in Informatics in Secondary Schools* (ISSEP 2006) (pp. 159 – 168). Berlin: Springer.

Gagné, R. M. (1985). *The conditions of learning and theory of instruction*. New York: Holt, Rinehart and Winston.

Gallant, R. J. ve Mahmoud, Q. H. (2008, March). Using greenfoot and a moon scenario to teach java programming in CS1. Paper presented at the *46th Annual Southeast Regional Conference on XX*, Auburn, Alabama, USA, New York: ACM.

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

Grant, N. S. (2003, October). A study on critical thinking, cognitive learning style, and gender in various information science programming classes. *Proceedings of the 4th Conference on Information Technology Curriculum (*pp. 96-99*)*, Lafayette, Indiana, doi:10.1145/947121.947142

Gundurao, H.K., Manjunath, N. S. ve Nachappa, M. N. (2010). *Computer technology and computer programming*. IND: Global Media.

Hansen, M. R. ve Kristensen, J. T. (2008). Experiences with functional programming in an introductory curriculum. İçinde Bennedsen, J., Caspersen, M. E. ve Kölling, M. (Ed.), *Reflections on the teaching of programming methods and implementations* (ss. 30-46). Berlin: Springer-Verlag.

Helminen, J. ve Malmi, L. (2010, October). Jype - a program visualization and programming exercise tool for Python. *Proceedings of the 5th International Symposium on Software Visualization (*pp. 153-162*)*, Utah, USA, doi:10.1145/1879211.1879234

Heppner, P. P. ve Petersen, C. H. (1982). The development and implications of a personal problem-solving inventory. *Journal of Counseling Psychology, 29*(1), 66-75.

Hernandez, C. C., Silva, L., Segura, R. A., Schimiguel, J., Ledon, M. F. P., Bezerra, L. N. M. ve Silveira, I. F. (2010). Teaching programming principles through a game engine. *Clei Electronic Journal, 13*(2), 1-8.

Howard, E. V., Evans, D., Courte, J. ve Bishop-Clark, C. (2006). A qualitative look at Alice and pair-programming. *Information Systems Education Journal, 7*(80).

Ibrahim, R., Rahim, N. Z. A., Ten, D. W. H., Yusoff, R., Maarop, N. ve Yaacob, S. (2018). Student's opinions on online educational games for learning programming ıntroductory. *International Journal of Advanced Computer Science and Applications*, *9*(6), 332-340.

Karsten, R., Kaparti, S. ve Roth, R. M. (2005). Teaching programming via the web: a time-tested methodology. *College Teaching Methods and Styles Journal, 1*(3), 73- 79.

Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. H. ve Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education, 10*(2), 109-128.

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

Kiser, S. S. (1989). *Logo programming, metacognitive skills in mathematical problem solving and mathematics achievement*. Unpublished Doctoral Dissertation, North Carolina University, USA.

Klassen, M. (2006, July). Visual approach for teaching programming concepts. *Proceedings of 9th International Conference on Engineering Education* (ICEE 2006) (pp. 23-28), San Juan, Puerto Rico.

Kosurkar, V., Zade, A., Tikas, A., Prasad, S. ve Sure, B. (2017). Learning programming language with game-like elements integrated on a web-based platform and assessment using formative feedback. *International Journal of Engineering Science*, *7*(3), 5268-5271.

Kökdemir, D. (2003). *Belirsizlik durumlarında karar verme ve problem çözme*. Yayınlanmamış Doktora Tezi (Unpublished doctoral dissertation), Social Science Institute, Ankara University, Ankara, Turkey.

Kölling, M. (2008). Greenfoot: a highly graphical ide for learning object-oriented programming. *ACM SIGCSE Bulletin*, *40*(3), 327-327.

Lawrence, A., Badre, A. ve Stasko, J. (1994, October). Empirically evaluating the use of animations to teach algorithms. *Proceedings of 1994 IEEE Symposium on Visual Languages (*pp. 48-54*)*, St. Louis, MO, USA, doi: 10.1109/VL.1994.363641

Lin, H. ve Kuo, T. (2010, June). Teaching programming technique with edutainment robot construction. Paper presented at *The 2nd International Conference on Education Technology and Computer* (ICETC), Shanghai, China, doi: 10.1109/ICETC.2010.5529557

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B. ve Resnick, M. (2004, January). Scratch: a sneak preview. Paper presented at *the 2nd International Conference On Creating, Connecting And Collaborating Through Computing*, Kyoto, Japan, pp. 104-109. doi: 10.1109/C5.2004.1314376

Maloney, J., Resnick, M., Rusk, N., Silverman, B. ve Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 16, 1-15.

Miyadera, Y., Kurasawa, K., Nakamura, S., Yonezawa, N. ve Yokoyama, S. (2007). A real-time monitoring system for programming education using a generator of program animation systems. *Journal of Computers, 2*(3), 12-20.

Mohorovicic, S. ve Tijan, E. (2011). Blende learning model of teaching programming in higher education. *The Journal of Knowledge and Learning, 7*(1), 2, 86-98.

Moskal, B., Lurie, D. ve Cooper, S. (2004). Evaluating the effectiveness of a new instructional approach. *ACM SIGCSE Bulletin, 36*(1), 75-79.

Nascimento, M. R., Mendonça, A. P., Guerrero, D. D. S. ve Figueiredo, J. C. A. (2010, October). Teaching programming for high school students: a distance education experience. Paper presented at *the 40th ASEE/IEEE Frontiers in Education Conference,* Washington, DC, USA, 27-30 October 2010, doi:10.1109/FIE.2010.5673642.

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A. ve Lahmine, S. (2015). Learning basic programming concepts by creating games with scratch programming environment. *Procedia-Social and Behavioral Sciences*, *191*, 1479-1482.

Partovi, H. ve Sahami, M. (2013). The hour of code is coming!. *ACM SIGCSE Bulletin, 45*(4), 5-5.

Pausch, R. (head), Burnette, T., Capeheart, A. C., Conway, M., Cosgrove, D., DeLine, R., Durbin, J., Gossweiler, R., Koga, S. ve White, J. (1995). Alice: rapid prototyping system for virtual reality. *IEEE Computer Graphics and Applications, 15*(3), 8-11.

Price, K. W. (2003). *Using visual technologies in the ıntroductory programming courses for computer science majors.* Doctoral dissertation, Nova Southeastern University Graduate School of Computer and Information Sciences, USA.

Radosevic, D., Orchovacki, T. ve Lovrencic, A. (2009, September). New approaches and tools in teaching programming. Paper presented at *the 20th Central European Conference on Information And Intelligent Systems*, Varazdin, Croatia, pp.49-57. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2505767

Reigeluth, C. M. (1987). *Instructional theories in action. Lessons illustrating selected theories and models*. Hillsdale, NJ: Lawrence Elrbaum Associates.

Repenning, A. ve Sumner, T. (1995). Agentsheets: A medium for creating domain-oriented visual languages. *IEEE Computer, 28*(3), 17-25.

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

Rogalski, J. ve Samurcay, R. (1990). Acquisition of programming knowledge and skills. İçinde Hoc, J. M., Green, T. R. G., Samurcay, R. ve Gillmore, D. J. (Ed.). *Psychology of programming* (ss. 157-174). London: Academic Press.

Sagisaka, T. ve Watanabe, S. (2008, October). Toward the development of adaptive learning system --- investigations of beginners in programming course based on learning strategies and programming test with gradual levels. Paper presented at *IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2008),* Freiburg, Germany.

Sanders, D. ve Dorn, B. (2003). Jeroo: a tool for introducing object-oriented programming. *ACM SIGCSE Bulletin, 35*(1), 201-204.

Schraw, G. ve Dennison, R. S. (1994). Assessing metacognitive awareness. *Contemporary Educational Psychology, 19*(4), 460-475.

Shaft, T. M. (1995) Helping programmers understand computer programs: the use of metacognition. *ACM SIGMIS Database, 26*(4), 25-46.

Shneiderman, B. ve Mayer, R. (1979). Syntactic/semantic interactions in programmer behavior: a model and experimental results. *International Journal of Computer and Information Sciences, 8*(3), 219-238.

Şahin, N., Şahin, N. H. ve Heppner, P. P. (1993). Psychometric properties of the problem solving inventory in a group of Turkish university students. *Cognitive Therapy and Research, 17*(4), 379-396.

Şendağ, S. ve Başer, Y. G. (2013). Öğretim teorileri ve öğretim teknolojileri. İçinde Çağıltay, K. ve Göktaş, Y. (Ed.), *Öğretim teknolojilerinin temelleri: teoriler, araştırmalar, eğilimler* (ss. 133-151). Ankara: Pegem Akademi.

Tekdal, M. (2013). The effect of an example-based dynamic program visualization environment on students' programming skills. *Journal of Educational Technology & Society*, *16*(3), 400-410.

Tsalapatas, H., Heidmann, O., Alimisi, R. ve Houstis, E. (2012). Game-based programming towards developing algorithmic thinking skills in primary education. *Scientific Bulletin of the Petru Maior University of Targu Mures, 9*(1), 56-63.

Vihtonen, E., Alaoutinen, S. ve Kaarna, A. (2001, October). Computer supported learning environment for c programming language. *Proceedings of the First Annual Finnish / Baltic Sea Conference On Computer Science Education (*pp:27-32*)*, Koli, Finland. Retrieved from https://www.kolicalling.fi/old_cms/archive/2001/koli_proc_2001.pdf#page=31

Wang, T., Mei, W., Lin, S., Chiu, S. ve Lin, J. M. (2009, October). Teaching programming concepts to high school students with Alice. Paper presented at the *39th ASEE/IEEE Frontiers in Education Conference*, San Antonio, TX, USA, doi:10.1109/FIE.2009.5350486.

Wilson, C. (2015). Hour of code---a record year for computer science. *ACM Inroads, 6*(1), 22-22.

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

Zaccone, R., Cooper, S. ve Dann, W. (2003, November). Using 3d animation programming in a core engineering course seminar. Paper presented at the *3rd ASEE/IEEE Frontiers in Education Conference*, Westminster, CO, USA. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1264764

Zapušek, M. ve Rugelj, J. (2013). Learning programming with serious games. *EAI Endorsed Trans. Game Based Learn*, *13*(01), e6.

Zhao, D., Chis, A. E., Muntean, G. M. ve Muntean, C. H. (2018, July). A large-scale pilot study on game-based learning and blended learning methodologies in undergraduate programming courses. Paper presented at *EDULEARN Conference, Palma de Mallorca, Spain*. Retrieved from http://www.newtonproject.eu/wp-content/uploads/2018/08/4.-Zhao_EDULEARN_CameraReady.pdf

http://kefad.ahievran.edu.tr

Ahi Evran University
Journal of Kırşehir Faculty of Education

ISSN: 2147 - 1037

The Effects of Visualization on Programming Learning: The Alice Environment*

Ebru Solmaz
Tolga Guyer

DOI:10.29299/kefad.2019.20.03.011 Article Information

Received:25/03/2019 Revised:20/07/2019 Accepted:10/10/2019

Abstract

For many years, the high failure rates in programming courses have forced educators and researchers to seek solutions to the problem. Numerous studies show that various solutions have been applied such as web-based tools, games, two- or three-dimensional programs and interface software. Most of those programs comprise visualization software that aids in acquiring the skills needed for programming learning (abstract thinking, critical thinking, algorithmic thinking, problem-solving, etc.) and to embrace abstract concepts. One of the best-known visualization tools is Alice, an object-oriented, 3D interactive visually-aided programming environment. Considering the underachievement rates in introduction to programming courses, in this study the Alice environment is evaluated as a tool for teaching the PHP programming language that might improve student scores. The study consists of experimental and control groups. In the experimental group, the PHP programming language was taught using Alice, while the control group was taught with no auxiliary tools. The research was carried out on 39 undergraduate students, 20 in the experimental group and 19 in the control. The effects of the Alice environment on students' critical thinking, problem-solving, and metacognition awareness levels were examined, and it was found that Alice did not create a meaningful difference in these three variables. The findings were compared to the results of similar studies and several suggestions were made for future research.

Key Words: Alice, Programming Teaching, Critical Thinking, Problem-solving, Metacognition Awareness

Author-in chief: Ebru Solmaz, Dr., Gazi University, Turkey, ebrusolmaz@gazi.edu.tr, 0000-0003-4893-450X
Tolga Güyer, Prof. Dr., Gazi University, Turkey, tguyer@gmail.com, 0000-0001-9175-5043
*This study was produced from the thesis titled "In programming language instruction the effect of Alice to course achievement, the skills of critical thinking and problem solving and metacognitive awareness level" written by Ebru SOLMAZ under the supervision of Tolga GÜYER.

Introduction

Some computers are needed not only for basic tasks such as searching for information, shopping, or paying bills but also to program information, create websites or develop applications for specific functions. This has raised awareness of programming among computer users, many of whom have started courses to learn programming languages. Also, the need for programming skills in technical fields such as engineering and many other sectors has increased the importance of programming learning. To illustrate, many organizations have recently run campaigns to promote the teaching of programming. One such is the Hour of Code. This campaign, which is open to the public, is a non-profit operation organized by Code.org., which aims to help anyone learn the basics of programming during Computer Sciences Week (3-9 December). Introduced in 2013, Code.org is supported by such well-known individuals as Bill Gates and Mark Zuckerberg as well as companies like Google and Apple. It enables students to learn programming using the drag & drop method (Kalelioğlu, 2015; Partovi and Sahami, 2013; Wilson, 2015).

Programming Learning and Difficulties

Researchers agree that learning a programming language is a difficult process (Shneiderman and Mayer, 1979; Hansen and Kristensen, 2008; Adán-Coello, Tobar, Faria, Menezes and Freitas, 2011). Learning and teaching programming are complicated subjects that require much effort and a special approach. Obtaining and improving information about programming is also a very complex process (Rogalski and Samurçay, 1990; Gomes and Mendes, 2007). To become good programmers, students should acquire a series of skills beyond learning the syntax of a programming language. These skills include various cognitive activities, mental descriptions related to program design, comprehension of the program, modification and error debugging related to written software, configuration of cognitive information, and performing basic functions (loop, condition sentences, etc.). These skills are also expressed as abstraction, abstract thinking, algorithmic thinking, problem-solving, generalization, transfer, metacognition, and critical thinking (Rogalski and Samurçay, 1990; Shaft, 1995; Bucci, Long and Weide, 2001; Futschek, 2006; Gomes and Mendes, 2007; Gundurao, Manjunath and Nachappa, 2010). Students experience difficulties during program learning because of the complicated process that requires various skills (Winslow, 1996). The difficulties that students encounter during their programming training cause overall underachievement in related courses that directly depend on those skills, hence the high failure rates (Sagisaka and Watanabe, 2008; Adán-Coello et al., 2011).

In the literature, various approaches are recommended to solve the problems encountered during programming teaching (Gomes and Mendes, 2007). Approaches used for programming teaching in related studies can be grouped: problem-based learning (Kay et al., 2000), blended

learning (Mohorovicic and Tijan, 2011; Zhao, Chis, Muntean and Muntean, 2018), computer-aided learning (Vihtonen, Alaoutinen and Kaarna, 2001), distance education (Nascimento, Mendonça, Guerroro and Figueiredo, 2010), and game-based learning (Zhao et al., 2018). Similarly, the tools that researchers have examined in this regard include: web-based tools (Wiki, Weblog, websites, etc.) (Cervesato, 2008; Karsten, Kaparti and Roth, 2005; Radosevic, Orehovacki and Lovrencic, 2009; Kosurkar, Zade, Tikas, Prasad and Sure, 2017), games (Doherty and Kumar, 2009; Zapušek and Rugelj, 2013; Ibrahim et al., 2018), game engines (Hernandez et al., 2010), robots and robotics (Lin and Kuo, 2010; Costa, Aparicio and Cordeiro, 2012), two-dimensional programs (Maloney et al., 2004; Gallant and Mahmoud, 2008; Kölling, 2008; Ouahbi, I., Kaddari, Darhmaoui, Elachqar and Lahmine, 2015), three-dimensional programs (Cooper, Dann, and Pausch, 2000a; Cooper et al., 2000b; Wang et al., 2009), interface software (Baldwin and Kuljis, 2001), and augmented reality environments (Dass, Kim, Ford, Agarwal and Chau, 2018). Although positive results are reported in the literature, none of these tools is in general use.

Visualization in Programming Learning

Studies of computer sciences show that most students have difficulties in applying abstract thinking to programming learning (Berge, Borge, Fjuk, Kaasboll and Samuelsen, 2003; Bucci et al., 2001). Problems during programming learning start at the very beginning of programming teaching. In other words, when students are learning and applying program structures or forming algorithms to solve concrete problems (Gomes and Mendes, 2007). Students have difficulty understanding what they are doing while running program codes when things are happening simultaneously on the scratchpad (Miyadera, Kurasawa, Nakamura, Yonezawa and Yokoyama, 2007). Various visual programming environments have been developed to solve these problems. These are used to help students better understand program codes and how a program is running (Tekdal, 2013). In addition, these environments allow students to writte program using graphic displays (notation) (Ben—Ari, 2013). Using visual elements in programming enables students to understand the meaning of introduced structures, prevents misconceptions, provides feedback for visuality, and allows exploratory learning and problem-solving (Brusilovsky, Kouchnirenko, Miller and Tomek, 1994).

One of the first programming environments encountered in the literature is Agentsheets (Repenning and Sumner, 1995). Then, visualization tools such as Greenfoot, Alice, and Scratch that were developed by academic projects appear. Also, there are studies that use different visual programs such as Jeroo (Sanders and Dorn, 2003), RAPTOR (Carlisle, Wilson, Humphries and Hardfield, 2005), Jype (Helminen and Malmi, 2010), and cMinds (Tsalapatas, Heidmann, Alimisi and Houstis, 2012).

According to previous research into the effects of Alice on basic programming courses (Cliburn, 2008; Moskal, Lurie, Cooper, 2004; Wang et al., 2009), students had positive experiences in lectures where Alice environments were used. In particular, the feature offering a 3D environment increased their interest in programming. Furthermore, thanks to the Alice program, students' self-confidence in their programming skills increased (Cooper et al., 2003; Howard, Evans, Courte and Bishop-Clark, 2006), they better understood basic programming concepts, better comprehended the relationship between algorithms and Alice stories and had fun (Howard et al., 2006). In the study by Moskal et al. (2004), data from the lecture on Computer Sciences 1 (CS1) show that the Alice program usually improves the performance of students. Data indicate that the Alice environment improves student performance, adds permanence related to computer sciences and improves the attitudes of students towards computer sciences. Similarly, the study conducted by Price (2003) showed that the use of the Alice program in the CS1 lecture meaningfully affected student performance. Additionally, two studies conducted by Cooper et al. (2000a, 2000b) using observation methods showed that students could use objects in the Alice program and apply methods to them easily. According to observation results, students could see the effectiveness of their programs, recognize the errors and correct them easily. Moreover, it was observed that students found Alice entertaining, liked to create three-dimensional animated worlds and spent much more time with the program than had been expected. Conversely, Wang et al. (2009) compared the languages of the Alice and C++ programs and found that students had difficulty and became bored, especially in repetition structure and the C++ programming language. Also, students said that learning Alice was easier than C++ programming and that using Alice would make learning C++ language easier. They also said they had fun and enjoyed learning. The study concludes that the group taught with Alice learned significantly better and that the Alice program is more effective in easing the understanding of basic programming concepts for students.

In this study, remembering the high number of students who have to repeat the introduction to programming course at university level, we aimed to find solutions to the problems that students encounter during programming learning. An important problem is that students cannot instantiate abstract concepts. Thus, we planned to use Alice software as an auxiliary visual tool to instantiate the concepts in basic programming. When examining the studies on Alice software, it is obvious that researchers particularly focus on variables such as student performance (achievement) and permanence. Most research comprises qualitative studies that take the students' opinions into account. However, it is clear that there are few quantitative studies into the effect of the environment on critical thinking, problem-solving, and metacognitive awareness – the very abilities that affect programming skills - and measuring them with various tools. Also, Cooper, Dann, and Pausch (2000b) state that the Alice environment supports competences such as the skills of algorithmic thinking and problem-

solving. Consequently, it is thought that Alice could eliminate problems related to improving these skills in students. Finally, the aim was to examine the effects of Alice as an auxiliary tool in the introduction to programming course on critical thinking, problem-solving, and metacognitive awareness, all of which are necessary elements of programming skill.

Method

In this quantitative research, we used the pretest-posttest control group design, one of the quasi-experimental designs. Students were chosen at random for the experimental and control groups.

Participants

Research was conducted with 39 second-grade students in the Computer and Instructional Technology Faculty of a public university. Students were split into experimental and control groups. Students who had previously taken the programming course were excluded. There were eight males and 11 females in the control group and seven males and 13 females in the experimental group.

Materials

Remembering that the introductory programming course forms the basis for the programming courses that follow and to decide on the proper environment for the experiment, we chose visualization tools that are already used in studies. Alice and Scratch are the most appropriate tools in terms of ease of use and installation. On the other hand, learner characteristics are also important elements in the selection of the environment. Although Scratch is the best-known and most used environment, it is designed for ages 8-16, while Alice is designed for older users (Maloney, Resnick, Rusk, Silverman and Eastmond, 2010). Consequently, we found the Alice environment more appropriate for the target group.

Alice software - an object-oriented, 3D-interactive, visually-supported programming environment - was developed to form three-dimensional environments by a research group at Carnegie Mellon University led by Randy Rausch. The software allows users to create virtual worlds and to animate objects within them (Wang et al., 2009). The English-language software can be downloaded free from www.alice.org. There are different versions for popular operating systems. Although its original version was based on the Python programming language, it is now fully Java-based. Users can write programs in Alice without learning the Java or Python programming languages (Cooper, Dann, and Pausch, 2003).

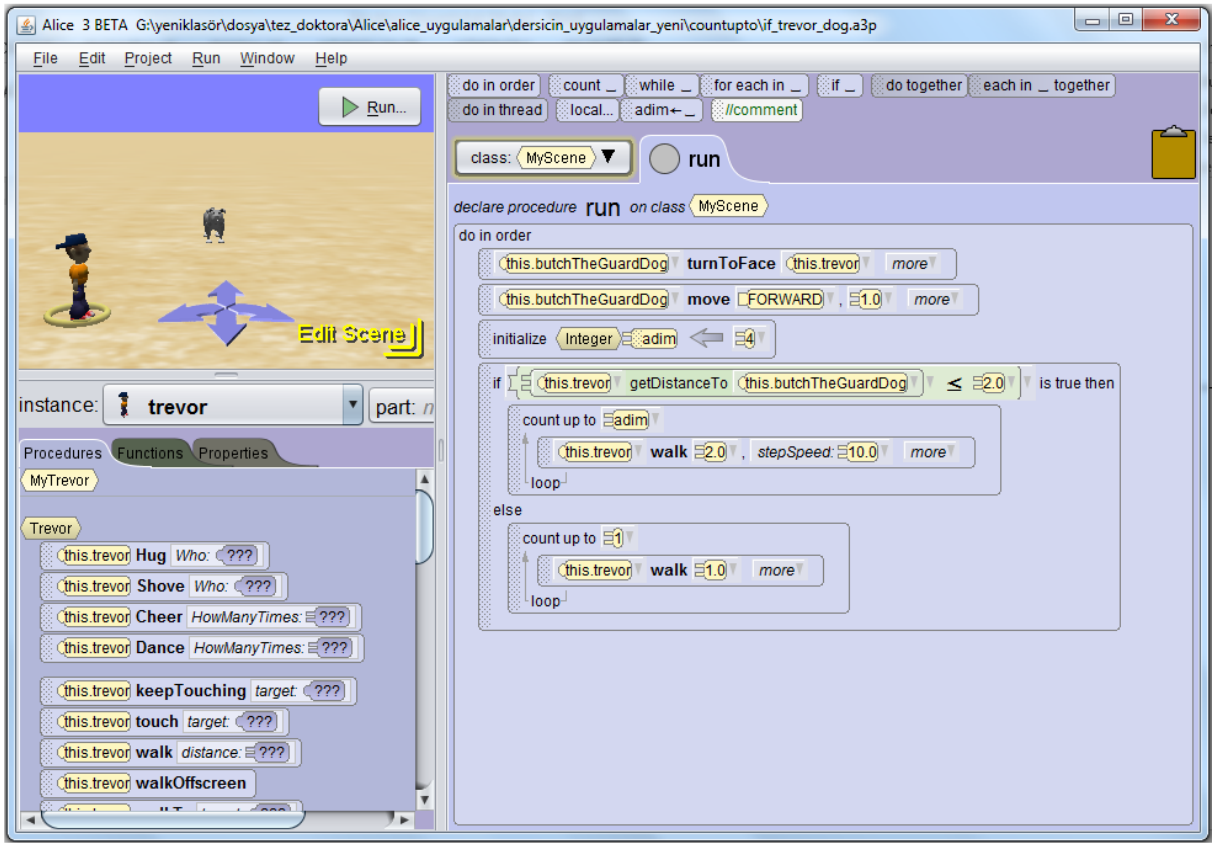


Figure 1. Interface of the Alice environment

Alice allows users to control features and move objects (Cooper et al., 2000b; Wang et al., 2009), see instantly how programs work and establish a relationship between program codes and animation movements by writing simple scripts (Cooper et al., 2000b). The Alice environment, which has a drag-drop interface like Scratch, prevents syntax errors so students focus on the concepts of programming rather than code debugging (Cliburn, 2008). Accordingly, students do not have to consider the relationship between non-computer events and objects within the computer or imagine default objects existing in the program (Berge et al., 2003). Additionally, students can better understand the scripts of programming language by seeing animated functions of different programming language structures (functions, loops, events, etc.). The environment also encourages students to develop their interest and level of participation (Cooper et al., 2000b; Wang et al., 2009).

Implementation

The research continued for seven weeks with five course-hours per week for each group, in the Programming Languages-I course in the 2013-2014 Academic Year Fall semester. In this course, which is the Introduction to Programming course, the PHP programming language is taught practically in a computer lab pratically. Subjects taught in the implementation process were determined after consulting instructors who have been teaching for at least five years and are in charge of the

Programming Languages-I course. We also examined textbooks on programming teaching. Details can be seen in Table 1:

Table 1. *Topics taught each week during the process of implementation*

Week	Titles
1	Data Types • String-based types • Numerical types
2	Variables and Fixed • Variables • Constants
3	Operators • Arithmetic Operators • String Operators • Assignment Operators • Comparison Operators • Logical Operators
4	Flow Control Statements • Conditional Statements ○If ○If/Else ○If/Else If/Else ○Switch/Case
5	Flow Control Statements • Loops ○Counter Loops ▪ For Loop ▪ For each Loop ○Conditional Loops ▪ While Loop ▪ Do…While Loop
6	Arrays • One-dimensional arrays • Multi-dimensional arrays
7	Functions • User-defined functions ○ Nonparametric functions ○ Parametric functions • Internal (inward) functions

In the process of implementation, Gagne's nine step model was used for the instructional design. According to Gagne, the steps of gaining attention, informing about objectives, stimulating recall of prior learning, presenting the content, providing learning guidiance, eliciting the performance, providing feedback, assessing the performance, enhancing retention and transfer are thought to be how to provide outsourcing for internal events (Reigeluth, 1987). Beyond that, steps in the model can be arranged differently according to instructions chosen in a lecture or curriculum (Driscoll, 1994).

With this model, Gagne (1985) introduced learning hierarchies built on top of each other. For him, learning needs intellectual skills and low-level skills set up as a substructure for high-level skills. Furthermore, pre-skills should be defined to teach specific skills and to ensure that students have them. In this study, we aimed for students to acquire programming skill, which is a skill going from simple to complex. Additionally, Şendağ and Başer (2013) state that Gagne's instruction theory is the method to use when giving introductory-level information. This study was conducted in the Programming Languages-I course to show the effect of the Alice environment on basic programming concepts. Consequently, the instructional design in the application process was planned and conducted based on this model to acquire the information and skills provided in this course. Training conducted each week is organized in detail by taking into consideration each step of the model.

While the Alice program was used as an auxiliary tool in the experimental group during the teaching of the course, the PHP programming language was taught to the control group without any auxiliary tools. Each week, the course subject was indicated primarily in the Alice program and the first half of the lecture was assessed by performing an application in that program. In the second half of the lecture, information was repeated in the PHP programming language and students were required to transfer what they learned in the Alice program into the PHP programming language. Meanwhile, they were informed about codes or display differences between Alice and the PHP programming language. Some exercises were shown during the lecture and students were asked to execute simple routines related to the subject of the week.

In the control group, the subject of the week was explained in the first half of the lecture by the instructor using no auxiliary tools in the PHP programming language. In the second half of the lecture, students were required to apply and consolidate the concept or concepts learned by conducting exercises related to the subject of the lecture. The application process is showed in Table 2:

Table 2. *Application Process*

<table>
<tr><th></th><th>Pretest</th><th>Practice</th><th>Posttest</th></tr>
<tr><td>Experimental Group</td><td rowspan="2">-Critical Thinking Disposition Scale
-Problem-solving Inventory
-Metacognitive Awareness Inventory</td><td>Alice + PHP</td><td rowspan="2">-Critical Thinking Disposition Scale
-Problem-solving Inventory
-Metacognitive Awareness Inventory</td></tr>
<tr><td>Control Group</td><td>PHP</td></tr>
</table>

Students in the experimental group learned basic concepts in Alice and then exercises were performed in the PHP programming language without repeating the concept or concepts learned in Alice. However, in the control group, basic concepts were explained in the first half of the lecture and

exercises were performed in the second half. Thus, even if lectures in the experimental group were taught using two different types of software, no time was lost. The same subjects were taught each week in the two groups. Also, the exercises performed during the lectures were the same in both groups so ensuring equality between the groups.

Data Collection Tools

In order to measure critical thinking skills, California Critical Thinking Disposition Scale (51 items), which was developed as a result of Delphi project organized by American Philosophy Association in 1990 and adapted to Turkish by Kökdemir (2003) was used. The coefficient of Cronbach alpha internal consistency of the scale was 0.88 and total variance explained by the scale was 36.13 % (Kökdemir, 2003).

The Problem-Solving Inventory (35 items) developed by Heppner and Peterson (1982) and adapted to Turkish by Şahin, Şahin, and Heppner (1993) was used to measure the problem-solving skills. The internal consistency coefficient of the scale was 0.88 and its reliability coefficient 0.81 (Şahin, Şahin, and Heppner, 1993).

The Metacognitive Awareness Inventory (52 items) developed by Schraw and Dennison (1994) and adapted to Turkish by Akın, Abacı, and Çetin (2007), was used to measure the metacognitive awareness. The internal consistency coefficient of the scale was 0.95 and the reliability coefficient 0.95 (Akın, Abacı, and Çetin, 2007).

Data Collection

At the beginning of the seven-week application process, students were asked to complete three data collection tools to determine their levels of critical thinking, problem-solving, and metacognitive awareness. At the end of seven weeks, the same data collection tools were applied again after the students had performed exercises related to the basic concepts of programming using three skills learned from the teaching process.

Data Analysis

The quantitative analysis of data at the end of the application was performed with the SPSS software package with the significance level of analysis accepted as 0.05.

Borg and Gall (1979 quoted in Cohen, Manion and Morrison, 2007) said that sample size in experimental research should be less than 15. Similarly, Büyüköztürk (2008) stated that according to the literature there are studies indicating that the use of parametrical tests causes no important

deviation in p significance levels when the sample size of sub-groups used in the research is 15 or more than 15. However, since a parametrical test should be used and the data should show normal distribution, a Kolmogorov-Smirnov Test was applied to determine whether the scores of critical thinking tendency, problem-solving skills, and levels of metacognitive awareness showed normal distribution (Table 3).

Table 3. *Distribution Scores of Research Data*

		N	P
Pretest Scores of Critical Thinking Disposition	Experimental	20	.20
	Control	19	.20
Posttest Scores of Critical Thinking Disposition	Experimental	20	.20
	Control	19	.20
Pretest Scores of Problem-Solving Skills	Experimental	20	.20
	Control	19	.173
Posttest Scores of Problem-Solving Skills	Experimental	20	.171
	Control	19	.20
Pretest Scores of Metacognitive Awareness	Experimental	20	.20
	Control	19	.20
Posttest Scores of Metacognitive Awareness	Experimental	20	.20
	Control	19	.20

As the significance scores of data for variables are higher than (p) .05, the data show normal distribution. Accordingly, it was decided to use two-factor variance analysis (ANOVA) to determine whether there was a significant difference between scores of the experimental and control groups for these three variables. Two-factor ANOVA indicates not only the impact of every independent variable but also the impact of two independent variables on each other (Cohen, Manion, and Morrison, 2007). In this study, data obtained from the experimental and control groups were compared to each other so not only in-group impacts but also intergroup impact became important.

Findings

Findings of the Critical Thinking Skill Scores

The results of two-factor ANOVA applied to determine whether there was a significant difference between scores of critical thinking disposition before and after the experiment are shown in Table 4:

Table 4. *ANOVA results of pretest-posttest scores of critical thinking disposition scale*

Source of the Variance		KT	sd	KO	F	P
Between-subjects	24521,75		38			
Group (Experimental/Control)		564,825	1	564,825	0,872	,356
Error		23956,925	37	647,484		
In-subjects	9925,952		39			
Measurement (Pretest-Posttest)		654,867	1	654,867	2,676	,110
Group*Measurement		**214,847**	**1**	**214,847**	**0,878**	**,355**
Error		9056,238	37	244,763		

Total	34447,702	77				

According to Table 4, student's critical thinking dispositions in the experimental and control groups did not show significant differences pre-experiment and post-experiment. That is, the Alice environment had no significant impact on critical thinking disposition, F(1,37)= ,878, p>0,05. In addition, there was no significant difference between the scores of the experimental and control groups in terms of critical thinking disposition pretest and posttest. This situation signifies that programming education using the Alice environment did not have significant impact on critical thinking disposition.

Findings of Problem-Solving Skill Scores

The result of the two-factor ANOVA test into whether there was a significant difference between changes observed in student scores in problem-solving before and after the experiment are given in Table 5:

Table 5. *ANOVA results of pretest-posttest scores of problem-solving scale*

Source of the Variance		KT	Sd	KO	F	P
Between-subjects	14254,949		38			
Group (Experimental/Control)		110,286	1	110,286	,289	,594
Error		14114,663	37	381,477		
In-subjects	3654,527		39			
Measurement (Pretest-Posttest)		574,912	1	574,912	,057	,012
Group*Measurement		**65,373**	**1**	**65,373**	**,802**	**,376**
Error		3014,242	37	81,466		
Total	17909,476		77			

It is evident from Table 5 that student problem-solving skills in the experimental and control groups did not show significant differences pre-experiment and post-experiment. That is, Alice had no significant impact on problem-solving skills, F(1,37)= ,802, p>0,05. Moreover, there was no significant difference between the scores of the experimental group and the control group in terms of problem-solving skills pretest and posttest. Consequently, programming learning using Alice did not impact on problem-solving skills.

Findings of Metacognitive Awareness Level Scores

The results of two-factor ANOVA applied to determine whether there was a significant difference between student scores of metacognitive awareness before and after the experiment are given in Table 6:

Table 6. *ANOVA results of pretest-posttest scores of metacognitive awareness scale*

Source of the Variance	KT	Sd	KO	F	p
Between-subjects	49254,718	38			
Group (Experimental/Control)	82,632	1	82,632	0,062	,804
Error	49172,086	37	1328,975		
In-subjects	4892,37	39			
Measurement (Pretest-Posttest)	280,421	1	280,421	2,335	,135
Group*Measurement	168,832	1	168,832	1,406	,243
Error	4443,117	37	120,084		
Total	54147,088	77			

According to Table 6, student metacognitive awareness levels in the experimental and control groups showed no significant difference pre-experiment and post-experiment. That is, the use of Alice did not create any significant impact on metacognitive awareness levels, $F(1,37)=1,406$, $p>0,05$. Similarly, there was no significant difference between cognitive awareness levels of the experimental and control groups pretest and posttest. These results indicate that programming learning using the Alice environment did not affect metacognitive awareness levels.

Discussion, Results, and Suggestions

Results of the research into the impact of Alice 3D visual software on teaching Introduction to Programming show that it does not have any significant impact on the variables of critical thinking, problem-solving, or metacognitive awareness. As the number of studies analyzing the impact of the Alice environment on these variables is so limited, we compared the data of similar environments or research results related to programming languages. Considering critical thinking, Grant (2003) compared student scores of critical thinking dispositions between those taking the course of traditional (procedure-based) programming and object-based programming. As a result, he concluded that there was no significant difference between the scores of these two groups or in their pretest-posttest scores. We produced parallel results considering that in our study PHP learning, which is a procedure-oriented programming language, was performed traditionally in one group and with the help of Alice, an object-oriented environment, in the other. While there was no significant difference between the impacts of procedure-based programming and object-based programming on critical thinking disposition, because the use of Alice as an object-based environment did not create any significant difference on critical thinking disposition, it might be said that the environment used and the type of language have no impact on critical thinking.

Other studies of problem-solving skills as the other variable had different results. Klassen (2006) reported that students with poor logical thinking skills could not develop their general problem-solving skills using Alice. That is, the reason why students' problem-solving skills did not develop

might be interpreted as because they have poor logical thinking skills. Furthermore, Carlisle et al. (2005) stated that teaching programming with the visual programming environment that they developed in their study, is more useful in developing problem-solving skills than teaching programming with non-visual and traditional languages. Moreover, Tsalapatas et al. (2012) stated that an environment known as cMinds, which is game-based visual programming with a visual learning environment, increases the capacity for problem-solving. While the research shows that two visual programming environments have positive effects on problem-solving skills, the reason why Alice does not create a significant difference on the variable might arise from the design features of the environment.

When comparing studies about the metacognitive awareness variable, it is clear that there is much research related to this variable. In the study carried out by Kiser (1989), it can be seen that just logo teaching consisting of problem-solving content does not increase student metacognitive skills. This might be because the method applied in programming education or the language or tools chosen had no impact on metacognitive skills.

Consequently, it might be said that visualization applications have no impact on the variables of critical thinking and metacognitive awareness. Their impact on problem-solving skills shows differences depending on the environment in general. In particular, the Alice environment has no impact on this variable.

Based on the results, some suggestions might be in order for future research. In particular, given the limited number of studies on the variables of critical thinking and metacognitive awareness, conducting studies similar to this one might provide generalizable results by supporting the results or disproving them. This study used the Alice environment to teach the PHP programming language and Alice and PHP were used successively in every lecture. The lecture in question forms the basis for some lectures in the following semester. It is a set curriculum that must be followed, so no changes could be made in the structure of the lecture. This situation is regarded as a limitation. Cliburn (2008) stated that when students learn Alice and another language in the same lecture, the aim of the lecture might be unclear to them. With this in mind, and within the limitations of the research, in future studies the preparatory class for one semester might begin by changing the use of the environment as suggested by some researchers (Klassen, 2006; Zaccone, Cooper and Dann, 2003). The Alice environment and the basic concepts of programming could be taught within the scope of this preparatory class. Studies examining the impact of Alice on learning programming language, which is the subject of the current introduction to programming lecture, could then be designed. This would avoid the negative situation created by students learning two different language structures consecutively.

The other limitation of the research is that the programming language in the lecture is PHP. Since changes could not be made to the curriculum, teaching of the PHP programming language was used in the study. PHP is a procedural programming language that offers an object-based environment. Given that the studies conducted by Price (2003), Moskal et al. (2004) show that the Alice environment improves the learning of object-oriented programming languages, it might be argued that the programming language is the reason why there are no significant results in this study. For this reason, the comparison and repetition of similar studies of introduction to programming lectures in which programming languages are taught will contribute to the literature.

A finding of the research was that Alice does not have similar results to other visualization tools that have positive impacts on the skills in question, according to results related to problem-solving skills. The reasons for this could be researched with qualitative studies that examine the features of the Alice environment in detail.

References

Adán-Coello, J. M., Tobar, C. M., Faria, E. J. , Menezes, W. S., and Freitas, R. L. (2011). Forming groups for collaborative learning of introductory computer programming based on students' programming skills and learning styles. *International Journal of Information and Communication Technology Education*, *7*(4), 34-46.

Akın, A., Abacı, R. ve Çetin, B. (2007). The validity and reliability of the Turkish version of the metacognitive awareness inventory. *Educational Sciences: Theory & Practice, 7*(2), 671-678.

Baldwin, L. P. and Kuljis, J. (2001, January). Learning programming using program visualization techniques. Paper presented at the *34th annual Hawaii international conference on system sciences*, Maui, HI, USA.

Ben—Ari, M. (2013). Visualization of programming. In Kadijevich, D. M., Angeli, C. ve Schulte, C. (Ed.), *Improving Computer Science Education (*ss. 61-74*).* New York: Routledge.

Berge, O., Borge, R.E., Fjuk, A., Kaasboll, J., and Samuelsen, T. (2003, November). Learning object oriented programming. *Paper presented at the Norsk informatik konferanse (Norwegian informatics conference),* Oslo, Norway.

Borg, W. R. and Gall, M. D. (1979). *Educational Research: An Introduction (third edition)*. London: Longman.

Brusilovsky, P., Kouchnirenko, A., Miller, P., and Tomek, I. (1994, June). Teaching programming to novices: a review of approaches and tools. *Proceedings of ED-MEDIA 94--World conference on educational multimedia and hypermedia*, Vancouver, British Columbia, Canada, pp. 103-110.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., and Miller, P. (1997). Mini-languages: a way to learn programming principles. *Education and Information Technologies, 2*, 65-83.

Bucci, P., Long, T. J., and Weide, B. W. (2001). Do we really teach abstraction?. *ACM SIGCSE Bulletin, 33*(1), 26-30.

Büyüköztürk, Ş. (2008). *Sosyal Bilimler için Veri Analizi El Kitabı,* 9.Baskı. Ankara: Pegem Akademi.

Carlisle, M.C., Wilson, T. A., Humphries, J. W., and Hadfield, S. M. (2005). RAPTOR: A visual programming environment for teaching algorithmic problem solving. *ACM SIGCSE Bulletin 37*(1), 176-180.

Cervesato, I. (2008). *On teaching programming languages using a wiki*. Report, Computer Science Department, Carnegie Mellon University, USA. Retreived from http://repository.cmu.edu/compsci/2

Cliburn, D. C. (2008, October). Student opinions of Alice in CS1. *Paper Presented at the 38th Annual Frontiers in Education Conference*, Saratoga Springs, NY, USA, doi: 10.1109/FIE.2008.4720254

Cohen, L., Manion, L., and Morrison, K. (2007). *Research Methods in Education* (Sixth Edition). London: Routledge.

Cooper, S., Dann, W., and Pausch, R. (2000a). Alice: A 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Small Colleges, 15*(5), 107-116.

Cooper, S., Dann, W., and Pausch, R. (2000b). Developing algorithmic thinking with Alice. In D. Colton (Ed.), *Proceedings of the 17th Information Systems Education Conference (ISECON 2000)* (pp. 506-539), Pensilvania, USA, Vol. 17.

Cooper, S., Dann, W., and Pausch, R. (2003). Using animated 3D graphics to prepare novices for CS1. *Computer Science Education*, *13* (1), 3-30.

Costa, C. J., Aparicio, M., and Cordeiro, C. (2012, June). A solution to support student learning of programming. *Proceedings of The Workshop on Open Source and Design of Communication* (pp. 25-29). ACM.

Dann, W., Cooper, S., and Pausch, R. (2000, July). Making the connection: programming with animated small worlds. *Proceedings of the 5th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE 2000) (pp. 41-44), Helsinki, Finland.

Dass, N., Kim, J., Ford, S., Agarwal, S., and Chau, D. H. P. (2018, April). Augmenting coding: augmented reality for learning programming. *Proceedings of the Sixth International Symposium of Chinese CHI (*pp: 156-159*)*, Montreal, Canada.

Doherty, L. and Kumar, V. (2009, August). Teaching programming through games. Paper presented at *International Workshop on Technology for Education*. Bangalore, India.

Driscoll, M. (1994). Gagne's theory of instruction. *Psychology of Learning for Instruction*. Boston, MA, Allyn and Bacon, 329-358.

Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. In R. T. Mittermeir (Ed.), *Proceedings of Evolution and Perspectives: 2nd International Conference in Informatics in Secondary Schools* (ISSEP 2006) (pp. 159 – 168). Berlin: Springer.

Gagné, R. M. (1985). *The conditions of learning and theory of instruction*. New York: Holt, Rinehart and Winston.

Gallant, R. J. and Mahmoud, Q. H. (2008, March). Using greenfoot and a moon scenario to teach java programming in CS1. Paper presented at the *46th Annual Southeast Regional Conference on XX*, Auburn, Alabama, USA, New York: ACM.

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

Grant, N. S. (2003, October). A study on critical thinking, cognitive learning style, and gender in various information science programming classes. *Proceedings of the 4th Conference on Information Technology Curriculum (*pp. 96-99*),* Lafayette, Indiana, doi:10.1145/947121.947142

Gundurao, H.K., Manjunath, N. S., and Nachappa, M. N. (2010). *Computer technology and computer programming*. IND: Global Media.

Hansen, M. R. and Kristensen, J. T. (2008). Experiences with functional programming in an introductory curriculum. In Bennedsen, J., Caspersen, M. E. and Kölling, M. (Ed.), *Reflections on the teaching of programming methods and implementations* (ss. 30-46). Berlin: Springer-Verlag.

Helminen, J. and Malmi, L. (2010, October). Jype - a program visualization and programming exercise tool for Python. *Proceedings of the 5th International Symposium on Software Visualization (*pp. 153-162*)*, Utah, USA, doi:10.1145/1879211.1879234

Heppner, P. P. and Petersen, C. H. (1982). The development and implications of a personal problem-solving inventory. *Journal of Counseling Psychology, 29*(1), 66-75.

Hernandez, C. C., Silva, L., Segura, R. A., Schimiguel, J., Ledon, M. F. P., Bezerra, L. N. M., and Silveira, I. F. (2010). Teaching programming principles through a game engine. *Clei Electronic Journal, 13*(2), 1-8.

Howard, E. V., Evans, D., Courte, J., and Bishop-Clark, C. (2006). A qualitative look at Alice and pair-programming. *Information Systems Education Journal, 7*(80).

Ibrahim, R., Rahim, N. Z. A., Ten, D. W. H., Yusoff, R., Maarop, N., and Yaacob, S. (2018). Student's opinions on online educational games for learning programming ıntroductory. *International Journal of Advanced Computer Science and Applications*, *9*(6), 332-340.

Karsten, R., Kaparti, S., and Roth, R. M. (2005). Teaching programming via the web: a time-tested methodology. *College Teaching Methods and Styles Journal, 1*(3), 73- 79.

Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. H., and Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education, 10*(2), 109-128.

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

Kiser, S. S. (1989). *Logo programming, metacognitive skills in mathematical problem solving and mathematics achievement*. Unpublished Doctoral Dissertation, North Carolina University, USA.

Klassen, M. (2006, July). Visual approach for teaching programming concepts. *Proceedings of 9th International Conference on Engineering Education* (ICEE 2006) (pp. 23-28), San Juan, Puerto Rico.

Kosurkar, V., Zade, A., Tikas, A., Prasad, S., and Sure, B. (2017). Learning programming language with game-like elements integrated on a web-based platform and assessment using formative feedback. *International Journal of Engineering Science*, *7*(3), 5268-5271.

Kökdemir, D. (2003). *Belirsizlik durumlarında karar verme ve problem çözme*. Yayınlanmamış Doktora Tezi (Unpublished doctoral dissertation), Social Science Institute, Ankara University, Ankara, Turkey.

Kölling, M. (2008). Greenfoot: a highly graphical ide for learning object-oriented programming. *ACM SIGCSE Bulletin*, *40*(3), 327-327.

Lawrence, A., Badre, A., and Stasko, J. (1994, October). Empirically evaluating the use of animations to teach algorithms. *Proceedings of 1994 IEEE Symposium on Visual Languages (*pp. 48-54*)*, St. Louis, MO, USA, doi: 10.1109/VL.1994.363641

Lin, H. and Kuo, T. (2010, June). Teaching programming technique with edutainment robot construction. Paper presented at *The 2nd International Conference on Education Technology and Computer* (ICETC), Shanghai, China, doi: 10.1109/ICETC.2010.5529557

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., and Resnick, M. (2004, January). Scratch: a sneak preview. Paper presented at *the 2nd International Conference On Creating, Connecting And Collaborating Through Computing*, Kyoto, Japan, pp. 104-109. doi: 10.1109/C5.2004.1314376

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 16, 1-15.

Miyadera, Y., Kurasawa, K., Nakamura, S., Yonezawa, N., and Yokoyama, S. (2007). A real-time monitoring system for programming education using a generator of program animation systems. *Journal of Computers, 2*(3), 12-20.

Mohorovicic, S. and Tijan, E. (2011). Blende learning model of teaching programming in higher education. *The Journal of Knowledge and Learning, 7*(1), 2, 86-98.

Moskal, B., Lurie, D., and Cooper, S. (2004). Evaluating the effectiveness of a new instructional approach. *ACM SIGCSE Bulletin, 36*(1), 75-79.

Nascimento, M. R., Mendonça, A. P., Guerrero, D. D. S., and Figueiredo, J. C. A. (2010, October). Teaching programming for high school students: a distance education experience. Paper presented at *the 40th ASEE/IEEE Frontiers in Education Conference,* Washington, DC, USA, 27-30 October 2010, doi:10.1109/FIE.2010.5673642.

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., and Lahmine, S. (2015). Learning basic programming concepts by creating games with scratch programming environment. *Procedia-Social and Behavioral Sciences*, *191*, 1479-1482.

Partovi, H. and Sahami, M. (2013). The hour of code is coming!. *ACM SIGCSE Bulletin, 45*(4), 5-5.

Pausch, R. (head), Burnette, T., Capeheart, A. C., Conway, M., Cosgrove, D., DeLine, R., Durbin, J., Gossweiler, R., Koga, S., and White, J. (1995). Alice: rapid prototyping system for virtual reality. *IEEE Computer Graphics and Applications, 15*(3), 8-11.

Price, K. W. (2003). *Using visual technologies in the ıntroductory programming courses for computer science majors.* Doctoral dissertation, Nova Southeastern University Graduate School of Computer and Information Sciences, USA.

Radosevic, D., Orchovacki, T., and Lovrencic, A. (2009, September). New approaches and tools in teaching programming. Paper presented at *the 20th Central European Conference on Information And Intelligent Systems*, Varazdin, Croatia, pp.49-57. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2505767

Reigeluth, C. M. (1987). *Instructional theories in action. Lessons illustrating selected theories and models*. Hillsdale, NJ: Lawrence Elrbaum Associates.

Repenning, A. and Sumner, T. (1995). Agentsheets: A medium for creating domain-oriented visual languages. *IEEE Computer, 28*(3), 17-25.

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

Rogalski, J. and Samurcay, R. (1990). Acquisition of programming knowledge and skills. İçinde Hoc, J. M., Green, T. R. G., Samurcay, R. ve Gillmore, D. J. (Ed.). *Psychology of programming* (ss. 157-174). London: Academic Press.

Sagisaka, T. and Watanabe, S. (2008, October). Toward the development of adaptive learning system --- investigations of beginners in programming course based on learning strategies and programming test with gradual levels. Paper presented at *IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2008),* Freiburg, Germany.

Sanders, D. and Dorn, B. (2003). Jeroo: a tool for introducing object-oriented programming. *ACM SIGCSE Bulletin, 35*(1), 201-204.

Schraw, G. and Dennison, R. S. (1994). Assessing metacognitive awareness. *Contemporary Educational Psychology, 19*(4), 460-475.

Shaft, T.M. (1995). Helping programmers understand computer programs: the use of metacognition. *ACM SIGMIS Database, 26*(4), 25-46.

Shneiderman, B. and Mayer, R. (1979). Syntactic/semantic interactions in programmer behavior: a model and experimental results. *International Journal of Computer and Information Sciences, 8*(3), 219-238.

Şahin, N., Şahin, N. H., and Heppner, P. P. (1993). Psychometric properties of the problem solving inventory in a group of Turkish university students. *Cognitive Therapy and Research, 17*(4), 379-396.

Şendağ, S. and Başer, Y. G. (2013). Öğretim teorileri ve öğretim teknolojileri. İçinde Çağıltay, K. ve Göktaş, Y. (Ed.), *Öğretim teknolojilerinin temelleri: teoriler, araştırmalar, eğilimler* (ss. 133-151). Ankara: Pegem Akademi.

Tekdal, M. (2013). The effect of an example-based dynamic program visualization environment on students' programming skills. *Journal of Educational Technology & Society*, *16*(3), 400-410.

Tsalapatas, H., Heidmann, O., Alimisi, R., and Houstis, E. (2012). Game-based programming towards developing algorithmic thinking skills in primary education. *Scientific Bulletin of the Petru Maior University of Targu Mures, 9*(1), 56-63.

Vihtonen, E., Alaoutinen, S., and Kaarna, A. (2001, October). Computer supported learning environment for c programming language. *Proceedings of the First Annual Finnish / Baltic Sea Conference On Computer Science Education (*pp:27-32*),* Koli, Finland. Retrieved from https://www.kolicalling.fi/old_cms/archive/2001/koli_proc_2001.pdf#page=31

Wang, T., Mei, W., Lin, S., Chiu, S., and Lin, J. M. (2009, October). Teaching programming concepts to high school students with Alice. Paper presented at the *39th ASEE/IEEE Frontiers in Education Conference*, San Antonio, TX, USA, doi:10.1109/FIE.2009.5350486.

Wilson, C. (2015). Hour of code---a record year for computer science. *ACM Inroads, 6*(1), 22-22.

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

Zaccone, R., Cooper, S., and Dann, W. (2003, November). Using 3d animation programming in a core engineering course seminar. Paper presented at the *3rd ASEE/IEEE Frontiers in Education Conference*, Westminster, CO, USA. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1264764

Zapušek, M., and Rugelj, J. (2013). Learning programming with serious games. *EAI Endorsed Trans. Game Based Learn*, *13*(01), e6.

Zhao, D., Chis, A. E., Muntean, G. M., and Muntean, C. H. (2018, July). A large-scale pilot study on game-based learning and blended learning methodologies in undergraduate programming courses. Paper presented at *EDULEARN Conference, Palma de Mallorca, Spain*. Retrieved from http://www.newtonproject.eu/wp-content/uploads/2018/08/4.-Zhao_EDULEARN_CameraReady.pdf