SİNEMADA GERÇEK ZAMANLI GÖRSEL ETKİLER: UNITY OYUN MOTORU İLE GÖRSEL ETKİ ÜRETİMİ

Yüce SAYILGAN
Marmara Üniversitesi, Türkiye
yucesayilgan@gmail.com

ÖZ
Bilgisayar işlemcilerinin gelişmesi ile dijital oyun motorlarının kapasitesi artmış ve pek çok oyun motoru, geliştiricileri tarafından yeni medya tasarımcılarına bedava veya belirli bir komisyon karşılığı bir üretim aracı olarak sunulmuştur. Bu durum oyun motorlarının kullanım alanlarını genişletmiş ve büyük bir tasarımcı topluluğu oluşturmuştur. Bu çalışmada, film üretim sürecinde, yenil nesil oyun motorlarından Unity'nin görsel etki oluşturma amacıyla nasıl kullanılabileceği sorusuna, tasarım merkezli bir yaklaşımla, yanıt aranmıştır. Bu amaçla kamerayla görüntülenen objelerin, renk silme yöntemiyle gerçek zamanlı olarak çizilen sanal uzayın içine yerleştirilmesi örneği üzerinden gidilmiştir.

Anahtar Kelimeler: *Görsel Etki, Oyun Motoru, Renk Silme*

REAL TIME RENDERING VISUAL EFFECTS IN FILM: VISUAL EFFECTS PRODUCTION WITH UNITY GAME ENGINE

ABSTRACT
Capacity of digital game engines increased with evolution of microprocessors and many game engines became available by their own developers as a production tool presented for a particular commision or free. In this circumstance, using fields of game engines is expanded and a big developers community has been occured. In this research, I tried to answer the question how we use Unity as one of the next generation game engines for creating visual effects in film development process from a design centric approach. For that purpose, camera shots and real time rendered images composed with chroma key technique.

Keywords: *Visual Effect, Game Engine, Chroma Key*

GİRİŞ

Electronic Magazines dergisinin 19 Nisan 1965 tarihli sayısında Intel şirketinin kurucularından Gordon Moore çarpıcı bir tez öne sürdü. Ona göre, mikroişlemciler içindeki transistör sayısı her yıl iki katına çıkacaktı. 1975 yılında bu öngörüsünü, bir buçuk kat olarak güncelledi.(Moore, 1965) Mikroişlemcilerdeki transistör sayısının artması bilgisayarların işlem gücünün artmasına olanak tanıyordu. "Moore yasası" olarak anılan bu öngörü, günümüze kadar büyük oranda geçerliliğini korudu. Bilgisayar işlemcilerinin gelişimi yaşamın her alanını etkilediği ve dönüştürdüğü gibi, Moore yasası ortaya atılmadan yetmiş sene önce ortaya çıkmış ve nispeten yeni sayılan bir sanat dalı olan sinemayı da kökten değiştirdi.

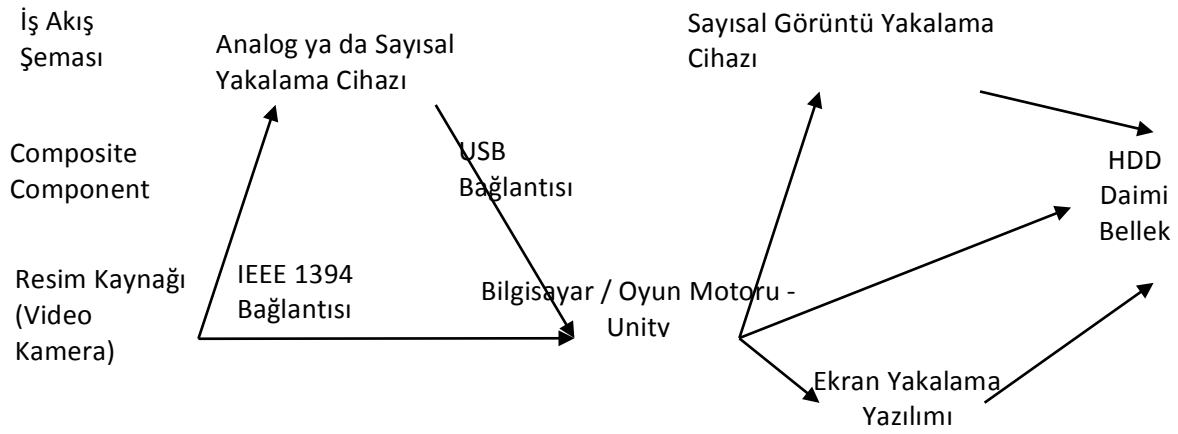
Bilgisayarların gelişimine paralel olarak, doksanların başından itibaren sinema ve televizyon alanında, yapım sonrası aşaması hızlı bir şekilde bilgisayarların hâkimiyetine girdi. Görsel etkiler, büyük oranda sayısal ortamda yaratılmaya başladı. Algıladığı ışık bilgisini sayısal veriye çeviren ve kaydeden sayısal film kameralarının ortaya çıkması ve yaygınlaşması da kurgu ve görsel etki yaratımında bilgisayarların kullanımını hızlandırdı. Günümüzde, üretilen filmlerin tamamına yakını sayısal kameralarla çekiliyor ve bilgisayar ortamında kurgulanıyor, renk düzeltmesi yapılıyor ve görsel etkileri hazırlanıyor.

Submit Date: 11.04.2017, Acceptance Date: 10.06.2017, DOI NO: 10.7456/10703100/007

Bu çalışma, sinema alanında eğitim gören öğrencilere görsel etki için aydınlatma ve görsel etki iş akışı konuları anlatılırken, ortaya çıkan sonuçların gerçek zamanlı olarak izlenebilme ihtiyacının karşılanması, ayrıca düşük maliyetli bir sanal stüdyo çözümüyle öğrencilerin kısıtlı mekânlarda daha özgürce çekimler yapabilmeleri amacıyla ortaya çıktı. Bu konudaki yazılı kaynakların yetersizliğinden dolayı, doğrudan doğruya deneyimleme yoluyla bir çalışma gerçekleştirildi. Yeni nesil oyun motorları incelendikten sonra, ücretsiz bir oyun motoru olan Unity seçildi ve standart çözünürlüklü bir video kamera ve profesyonel aydınlatma araçları kullanılarak stüdyo ortamında gerçekleştirilen çekimlerle ortaya konulan iş akışı yaratıldı ve sınandı.

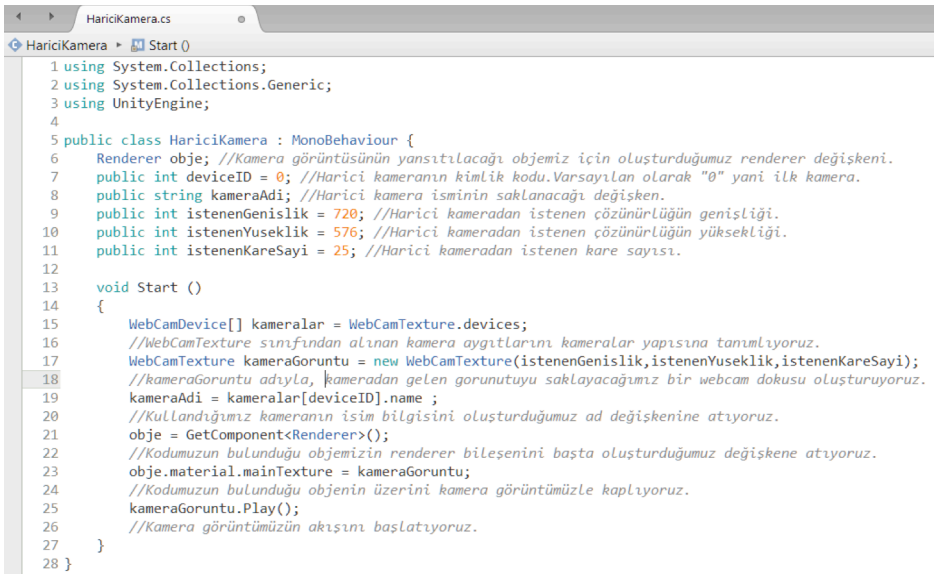
Görsel etkiler, çekilmesi mümkün olmayan, tehlikeli ya da fazla maliyetli görsellerin yaratılmasında kullanılan tekniklerdir.(Okun ve Zwerman, 2010:2) Birden fazla gerçek çekimin gerçekliği manipüle edecek şekilde birleştirilmesi, çekimlerin manipüle edilmesi ya da bilgisayarda oluşturulmuş görüntülerin gerçek görüntülerle birleştirilmesi ile oluşturulabilir. Görsel etkiler hazırlanırken sıklıkla birçok kaynaktan gelen görsel elemanlar birleştirilerek kullanılır.(Bu işlem yaygın olarak "compositing" terimiyle tanımlanır.) Bu görsel elemanlar gerçek çekimler olabileceği gibi, bilgisayarda hazırlanmış canlandırma ya da resimler olabilir. Profesyonel görsel etki iş akışı içerisinde, bilgisayarda yaratılan bu görsel elemanlar, sıklıkla önceden hazırlanmış ve bir resim ya da video formatında kaydedilmiş olur.

Birleştirmeyi yapan tasarımcının elinde, önceden hazırlanmış ve kaydedilmiş pek çok, gerçek ya da bilgisayarda oluşturulmuş görsel malzeme bulunabilir. Bilgisayarda hazırlanan görsel malzemelerin önceden çizilerek kaydedilmesi azami görüntü kalitesinin elde edilmesinin hedeflenmesidir. Özellikle, bilgisayarda hazırlanan üç boyutlu görseller iki boyutlu olarak çizilirken gerçekçiliği yakalamak için, yüksek işlemci gücü gerekir. Bu durum üç boyutlu görsellerin istenilen kalitede çizilmesinin gerçek zamanlı olarak yapılmasını güçleştirir. Örneğin; gerçek bir oyuncu, gerçekçi sanal bir mekânın içine yerleştirilecekse, görüntü arka fonunun silinmesine uygun olarak çekilir ve kaydedilir. Mekân üç boyutlu olarak bilgisayar ortamında oluşturulur ve bilgisayar tarafından iki boyutlu olarak çizilmesi sağlanarak kaydedilir. Daha sonra bu iki görsel eleman yazılımlar aracılığıyla birleştirilerek tekrar çizilir ve kaydedilir. Bu yöntem oluşturulan görsellerde yüksek kalite elde edildiği için özellikle film üretiminde tercih edilirken, arzulanan sonucun gerçek zamanlı olarak görülebilmesinin mümkün olmaması olumsuz yanı olarak kabul edilebilir. Görsel etkilerin gerçek zamanlı olarak kaliteli olarak yaratılabilmesi, görsel etki yaratım sürecini, oldukça kolaylaştıracak ve maliyetleri düşürecektir.



Resim 1:Unity Oyun Motorunda Görsel etki İş Akışı

Submit Date: 11.04.2017, Acceptance Date: 10.06.2017, DOI NO: 10.7456/10703100/007



```
HariciKamera.cs
HariciKamera ▸ Start ()
 1 using System.Collections;
 2 using System.Collections.Generic;
 3 using UnityEngine;
 4
 5 public class HariciKamera : MonoBehaviour {
 6     Renderer obje; //Kamera görüntüsünün yansıtılacağı objemiz için oluşturduğumuz renderer değişkeni.
 7     public int deviceID = 0; //Harici kameranın kimlik kodu.Varsayılan olarak "0" yani ilk kamera.
 8     public string kameraAdi; //Harici kamera isminin saklanacağı değişken.
 9     public int istenenGenislik = 720; //Harici kameradan istenen çözünürlüğün genişliği.
10     public int istenenYuseklik = 576; //Harici kameradan istenen çözünürlüğün yüksekliği.
11     public int istenenKareSayi = 25; //Harici kameradan istenen kare sayısı.
12
13     void Start ()
14     {
15         WebCamDevice[] kameralar = WebCamTexture.devices;
16         //WebCamTexture sınıfından alınan kamera aygıtlarını kameralar yapısına tanımlıyoruz.
17         WebCamTexture kameraGoruntu = new WebCamTexture(istenenGenislik,istenenYuseklik,istenenKareSayi);
18         //kameraGoruntu adıyla, kameradan gelen gorunutuyu saklayacağımız bir webcam dokusu oluşturuyoruz.
19         kameraAdi = kameralar[deviceID].name ;
20         //Kullandığımız kameranın isim bilgisini oluşturduğumuz ad değişkenine atıyoruz.
21         obje = GetComponent<Renderer>();
22         //Kodumuzun bulunduğu objemizin renderer bileşenini başta oluşturduğumuz değişkene atıyoruz.
23         obje.material.mainTexture = kameraGoruntu;
24         //Kodumuzun bulunduğu objenin üzerini kamera görüntümüzle kaplıyoruz.
25         kameraGoruntu.Play();
26         //Kamera görüntümüzün akışını başlatıyoruz.
27     }
28 }
```

Resim 2.Harici Görüntü Aygıtından Görüntü Almak İçin Kullanılan Unity 5.5 C# Kodu

Üç boyutlu grafiklerin gerçek zamanlı olarak çizimi, dijital oyunların yaygınlaşması ile hızla gelişme göstermiştir. Özellikle, fiziksel tabanlı çizim yapan ve foto-gerçekçiliği hedefleyen Unreal Engine, Unity gibi oyun motorlarının yeni sürümlerinin kullanılmaya başlanması ile beraber gerçeğe çok yakın grafikler oyun motorları ile gerçek zamanlı olarak çizilebilmektedir. Ayrıca bu oyun motorlarının belli yazılım dilleriyle rahatlıkla kodlanabilmeye açık olması ve mobil platformları destekledikleri için dolayı dijital kameraların özelliklerine erişmeye olanak tanımaları, oyun motorlarını görsel etki oluşturma sürecinde, hatta renk düzeltme sürecinde etkin olarak kullanabileceğimiz araçlar haline getirmektedir. Oyun motorlarını tamamen canlandırma grafiklerin çiziminde kullanabileceğimiz gibi, gerçek oyuncuları sanal mekânların içine yerleştirmek için; ya da sanal oyuncuları gerçek mekânların içine yerleştirmek için kullanabiliriz.

Unity, Unity Technologies tarafından geliştirilen Haziran 2005'de ilk kararlı sürümü yayınlanan yeni nesil bir oyun motorudur. Tamamen ücretsiz sürümü bulunan oyun motoru, gerçek zamanlı grafikleri yüksek kalitede çizmeye olanak sağlamaktadır. Kodlama kütüphanesinde yer alan kodlarla harici bir video kameradan alınan görsel verinin oyun motorunu içine gerçek zamanlı olarak alınması ve resim sekansı olarak çıktı alınması ya da yardımcı ekran yakalama yazılımları ve aygıtları ile video olarak kaydedilmesi mümkündür.

KAMERADAN GELEN GÖRÜNTÜNÜN OYUN MOTORUNUN İÇİNE ALINMASI

Genel olarak, günümüz görsel etki iş akışında, oyuncular yeşil ya da mavi perde önünde çekilir ve çekim sonrasında yeşil ya da mavi renk silinerek sanal bir mekânın içerisinde yerleştirilir. Bu tarz bir iş akışında oyuncuların yerleştirilecekleri sanal dünyayı gerçek zamanlı olarak görmeleri ya da sanal uzaydaki nesnelerle etkileşime girmeleri mümkün olmaz. Oyuncular ve film ekibi hayal güçlerini kullanarak ve referans görüntüleri esas alarak yaratacakları sanal dünya varmış gibi film çalışmasını gerçekleştirirler. Gerçek mekânlarla sanal karakterlerin birleştirildiği çalışmalarda da aynı yöntem izlenir. Kamerayla görüntülenen fiziksel dünyanın sanal uzayın içine gerçek zamanlı olarak getirilmesi ve oyun motorunun sanal dünyayı ya da karakterleri gerçek zamanlı çizebilme olanağından yararlanılması bu süreci oyuncular ve set ekibi tarafından kolaylaştıracaktır. Polygon.com'un 1 Mart 2017 tarihli "Star Wars: Rogue One'ın en iyi karakteri gerçek zamanlı çizildi, sinemada bir ilk" başlıklı haberinde görsel etki süpervizörü John Knoll, görsel etkileri son halinin anında görülebilmesinin filmin yapımına yardım ettiğini açıklamıştır. (Polygon.com, 2017)

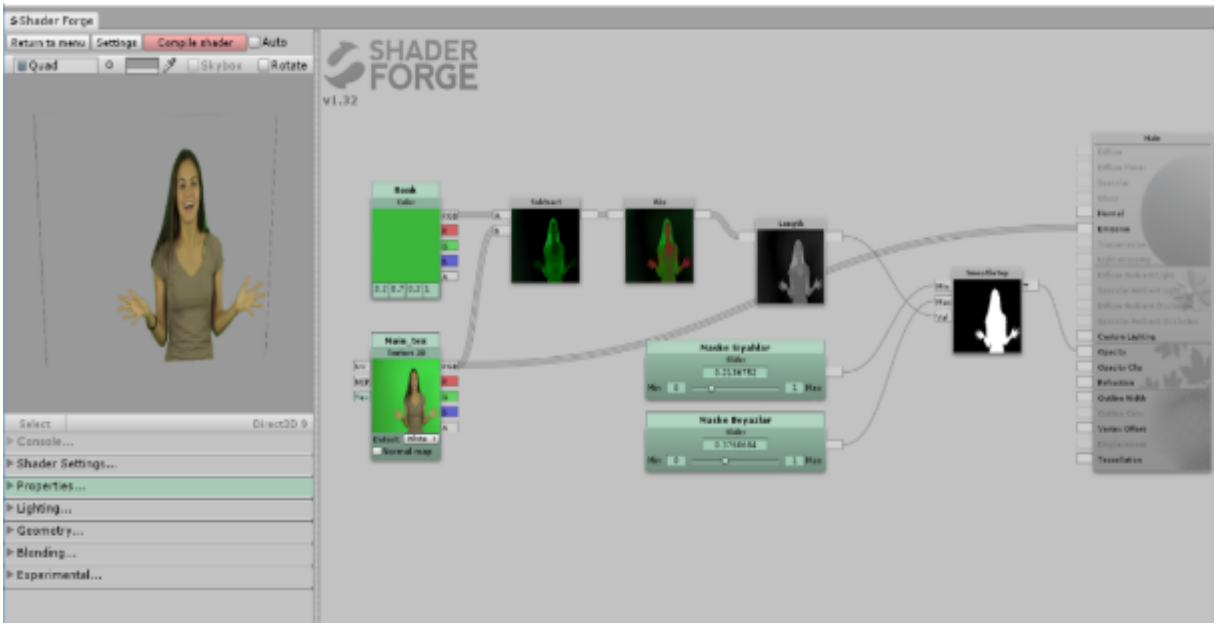
Star Wars: Rogue One filminde kullanılan Unreal Engine 4 gibi Unity de yeni nesil bir oyun motorudur ve benzer özellikler içerir. Unity, JavaScript ya da C# programlama dili kullanılarak tüm özelliklerine erişilebilen bir yazılımdır. Unity uygulama programlama arayüzü (Unity API) kullanıcıya Unity oyunları ve yazılımları oluşturması için birçok araç sunar. Unity oyun motorunun içine harici bir

Submit Date: 11.04.2017, Acceptance Date: 10.06.2017, DOI NO: 10.7456/10703100/007

kameradan görüntü çağırabilmek için UnityEngine kütüphanesinde yer alan "WebCamTexture" sınıfını kullanılır.(Unity Technologies, 2017) Bu sınıfın içinde yer alan elementlerle kameradan gelen görüntüyü bir doku dosyası haline getirip istediğimiz öğeye atamamız mümkün olmaktadır. Resim 2'de, harici kameradan görüntüyü almamızı sağlayacak C# programlama diliyle yazılmış bir kod örneği bulunmaktadır. Burada yer alan bazı değişkenler çıkartılarak kod daha da sadeleştirilebilir, ya da Unity'nin diğer kütüphane öğelerinden de yararlanılarak başlat-durdur tuşları gibi ek özellikler katılabilir. Bu koddan yararlanabilmemiz için öncelikle sahnemizde üç boyutlu bir obje oluşturmamız gerekir. Örneğin; oyun motorunda hazır olarak sunulan üç boyutlu objelerin içerisindeki "plane" objesi kullanılabilir. Bu objenin kullanımı bize görüntüyü yansıtacak düz bir yüzey sağlar. Sonrasında oluşturduğumuz objeye, oluşturduğumuz kodu, bileşen olarak eklememiz gerekir. Kod dosyasını objenin üzerine sürükleyerek ekleriz. Unity editöründe oyun ön izlemesini başlattığımızda kameramızdan görüntümüz akmaya başlayacaktır, durdurduğumuzda ise görüntümüz kaybolacak objemiz malzemesinin ön tanımlı rengine bürünecektir. Objemizin büyüklüğünü değiştirerek görüntümüzün oranını istediğimiz gibi değiştirebiliriz ya da objemizin konumunu değiştirerek görüntümüzü üç boyutlu sahnenin içinde istediğimiz gibi konumlandırabiliriz.

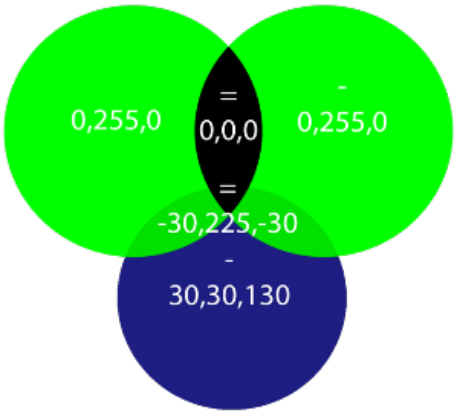
SHADER FORGE EKLENTİSİ İLE RENK SİLMESİNE UYGUN GÖLGELENDİRİCİ HAZIRLANMASI

Unity oyun motoru içerisinde, üç boyutlu ve iki boyutlu sahne unsurlarına malzeme tanımlanabilir. Malzemeler, gölgelendirici terimiyle tanımlanan kod dizileriyle çalışırlar ve kullandıkları gölgelendirici ile görsel özellikleri belirlenir. Malzemeler nesneler, dokular ve ışıklar arasında bağı kurar. Bu bağın nasıl olacağı ise, örneğin nesnenin ışığa karşı nasıl davranacağı gibi, gölgelendiricilere göre belirlenir. "Tamamen beyaza boyalı bir küp hayal edelim. Bütün yüzeylerde aynı renk kullanılsa bile, ışığın geliş yönüne ve bakış açımıza göre hepsi beyazın farklı bir tonu olacaktır. Üç boyutlu grafiklerde bu ek gerçekçilik, ışığın davranışını taklit eden gölgelendiriciler ile sağlanır."(Zucconi, Lammers, 2016: 36) Örneğin, kameradan aldığımız görüntü herhangi bir malzemeye tanımlandığında nasıl görüneceği, o malzemenin gölgelendiricisi tarafından belirlenir. Bu özelliği kullanarak kameradan aldığımız görüntü üzerinde neredeyse sonsuz oynama imkânımız bulunur. Gölgelendirici yazılımlarını yazmak için çeşitli programlama dilleri geliştirilmiştir. Unity pek çok gölgelendirici programlama dilini destekler. Unity projemiz için, HLSL ya da Open GL gibi programlama diliyle gölgelendiriciler oluşturabileceğimiz gibi Shader Forge gibi bize görsel programlama imkânı sunan bir eklenti ile de gölgelendiricimizi oluşturabiliriz.



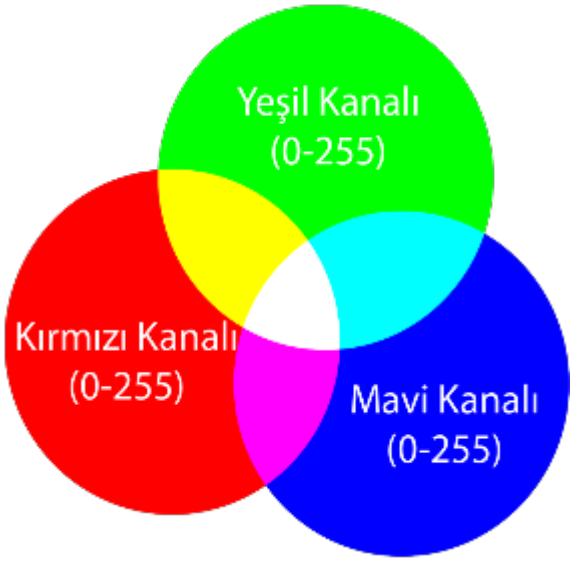
Resim 3: Unity Shader Forge Eklentisi Renk Silme Gölgelendiricisi Şeması

Submit Date: 11.04.2017, Acceptance Date: 10.06.2017, DOI NO: 10.7456/10703100/007



Resim 4: *Katmalı renk karışımına göre çeşitli renk değerleri arasındaki işlemler.*

Resim 3'de görülen Shader Forge eklentisinde hazırlanan bir renk silme gölgelendiricisinin grafiğidir. Burada görülen her unsur aslında gölgelendirici programlama dilindeki çeşitli fonksiyonların görsel karşılığıdır. Pek çok farklı şekilde rengi silecek gölgelendiriciler hazırlanabilir. Her gölgelendirici yazılımının giriş değerleri ve çıkış değerleri bulunur. Bu grafikte iki giriş değeri bulunmaktadır. Birincisi kameradan aldığımız görüntüyü atayacağımız "Main Tex" girişimiz, ikincisi ise kamera görüntüsünden çıkararak saydamlaştırmak istediğimiz rengi seçtiğimiz "Renk" girişi. Shader Forge'da bu girişler "Properties(Özellikler)" olarak isimlendirilir. Bu girişleri oluşturmak için, kamera görüntümüzün yer alacağı "Texture2D" düğümü(node) ve sileceğimiz rengi seçeceğimiz bir "Color" düğümü eklememiz gerekir. Shader Forge düğüm tabanlı (node based) bir yazılımdır. Bu tür yazılımlarda her bir öğe, düğüm olarak isimlendirilir ve düğümlerin birden fazla girişi ve çıkışı bulunabilir.



Resim 5: Katmalı Renk Karışımı

Girişi bulunmayan özellik düğümleri haricindekiler, girişten aldığı veriyi düğümün özelliğine göre işleyerek çıkış ya da çıkışlara verir. Texture2D ve Color düğümü girişi olmayan düğümlerdir. Buradaki giriş, nesnenin malzemesi tarafından yapılır. Buradaki amaç her seferinde gölgelendirici koduyla oynamadan, malzeme arayüzünü kullanarak veri girişi yapmaktır.

Submit Date: 11.04.2017, Acceptance Date: 10.06.2017, DOI NO: 10.7456/10703100/007

Oluşturacağımız gölgelendirici, fiziksel tabanlı bir gölgelendirici, aydınlatılabilen(lit) ya da aydınlatılamaz(unlit) bir gölgelendirici olabilir. Bu örnekte, aydınlatılamaz bir gölgelendirici oluşturulmuştur ve renk kanalı yoktur. Bunun yerine kendi ışığını yaydığı için "Emission" kanalı bulunur. Girişten çıkışa iki ayrı bilgiyi taşımamız gereklidir. Birincisi "Emission" kanalına taşıyacağımız renk bilgisi, ikincisi ise bir maske oluşturarak "Opacity" kanalına taşıyacağımız saydamlık bilgisi. Bunun için tamamen siyah ve beyazın tonlarından oluşturduğumuz bir maske yaratmamız gereklidir. Buradaki beyaz görünür alanları, siyah da saydam alanları belirtir. Siyah ve beyaz haricindeki gri tonlar resmin yarı saydam görünmesine yol açar. Girişlerimiz için kullanılan resimlerimiz ve renklerimiz RGB renk modelinde oluşturulmuştur ve 4 adet 8 bitlik kanal içerir. Bunlardan katmalı renk karışımına göre üç temel renk olan kırmızı, yeşil ve mavi için kullanılır. "Katmalı karışım, renklerin ışık olarak karışımıdır. Renkli televizyon, resim ve baskı tekniğinde olduğu gibi bazı renklerin birbirleriyle karıştıkları zaman değişik renkler meydana getirmesi olayına dayanmaktadır. Mavi, yeşil ve kırmızının şiddetleri eşit kalmak üzere azaltıldıkça beyaz grileşir, gri koyu gri olur neticede siyah görünür. Örneğin, bu ışıklardan yeşille kırmızı belli oranlarda birbirlerine karıştırıldıklarında sarı renk elde edilir."(Vardar, 2006:39) Dördüncü kanal ise alfa kanalıdır.

Başlangıçta video görüntümüzün yer alacağı düğümümüzün birleşik RGB çıkışını, gölgelendiricimizin "emission" girişine bağlamamız gerekir. Böylece videomuz görüntüde belirebilir. Diğer işlemler tamamen videodaki istenmeyen bölgeyi silmek için siyah-beyaz bir maske yaratmak amacıyladır. Bunun için yapılması gereken ilk işlem silmek istediğimiz renkten, video görüntümüz renk değerlerini çıkarmaktır. Yani RGB değerleri arasında çıkarma işlemi yapmaktır. 8 bitlik RGB görüntüde her kanal 0-255 arası değer alabilir. Örneğin; R:0, G:255, B:0 değerleri almışsa bu tam anlamıyla yeşildir. Görüntümüzdeki her pikselin 0-255 arası bir kırmızı, bir yeşil, bir mavi değeri bulunur ve renkler katmalı renk karışımına göre oluşturulur. 0,0,0 değeri siyahtır ve 255,255,255 değeri beyazdır. Bu değerler 0 ya da 255'ten daha büyük ya da daha küçük olsa dâhi renksel olarak bir karşılıkları olamaz, eksi değerler siyah, 255 üstü değerler ise beyaz görünür.

Bu özellikten yararlanarak "Color" düğümümüzden, yani görüntüden çıkarmak istediğimiz rengimizden, "Main tex" düğümümüzün renk değerlerini çıkardığımızda benzer değerler birbirini sıfırlayacağından silmek istediğimiz renkte pikseller sıfır ya da sıfıra yakın değer alacaktır. Bu çıkarma işlemini "Substract" düğümünü kullanarak yapabiliriz. Ayrıca seçtiğimiz renk hariç diğer tüm renkler de sıfır ya da sıfırın altına düşecektir. Bu değerleri geri getirip artı değerlere çevirmek durumundayız ki, maskemizdeki pozitif değerleri oluşturabilelim. Bunun için "Abs" düğümünü kullanıyoruz ve tüm eksi değeleri değerleri artıya çeviriyoruz. Daha sonra aldığımız değerleri "Length" düğümüne giriyoruz. Length düğümü aldığı değerlerin birbirine olan uzunluğunu hesaplar ve en büyük uzaklık değerlerini beyaz, en küçük uzaklık değerlerini ise siyah olarak görselleştirir. Böylece sadece siyah, beyaz ve grinin tonlarından oluşan bir resim oluşturmuş oluruz.

Ancak maskemizdeki gri değerleri yarı saydam görürüz, bu yüzden gri değerleri siyaha ya da beyaza yaklaştırmak durumundayız. Bunun için "Smooth Step" düğümünü kullanabiliriz. Bu düğümde "Min", "Max" ve "Value" olmak üzere üç giriş bulunur. Value girişine length düğümüyle oluşturduğumuz parlaklık değerlerinden oluşan veriyi gireriz. Smooth Step düğümü, "Value" girişine girdiğimiz resmin, belirlenen min değerinden düşük değerlerini siyah, max değerinden büyük değerlerini de beyaz haline getirir. Bunun için "Siyahlar" ve "Beyazlar" olarak isimlendiriğimiz iki slider kullanabiliriz. Bu yöntemle çok yüksek karşıtlıklı, maskelemeye uygun bir siyah beyaz resim oluşturmak mümkündür. Burada alacağımız çıkışı gölgelendirici düğümümüzün "Opacity" girişine girmemiz gerekir.

BİRLEŞTİRİLEN GÖRÜNTÜLERİN KAYDEDİLMESİ

Oyun motorunda birleştirilen gerçek ve sanal görüntüler pek çok yöntemle kaydedilebilir. Unity oyun motorunun standart kütüphanelerinde video kaydetmeye uygun komutlar bulunmamaktadır. Ancak Unity Engine kütüphanesi içinde yer alan "Application.CaptureScreenshot" komutu ile resim sekansları kaydetmek mümkündür. Resim 6'daki komutlar kullanılarak kolaylıkla resim sekansları

Submit Date: 11.04.2017, Acceptance Date: 10.06.2017, DOI NO: 10.7456/10703100/007

oluşturulabilir. Bu yöntem oldukça pratik olmakla beraber kameradan gelen görüntünün kare sayısı oyun motoru tarafından sınırlandırıldığında gerçek görüntülerdeki hareketlerde zamanın bozulması sonucu hızlanmalar yaşanmaktadır. Bu ancak çok basit sahneler oluşturarak ya da kaydetme boyutunu küçülterek aşılabilir. Bu yüzden yüksek çözünürlüklü resimler oluşturulmak istenen durumlarda kullanım alanları sınırlıdır. Bu yöntemde yaşanan hız sorunları sistemden sisteme değişiklik gösterebilir.

Bir başka yöntem, ekran görüntüsünü kaydeden bir kaydedici yazılım ya da donanım kullanmaktır. Örneğin; Open Broadcaster Software ücretsiz ve açık kaynak kodlu bir ekran kaydetme yazılımıdır. İşletim sisteminde açık olan herhangi bir pencere ya da ekranın tümü bu yazılım vasıtasıyla kaydedilebilir. Oluşturulan görüntünün bu yöntemle kaydedilmesi daha yüksek çözünürlüklerle kayıt yapmak ve doğrudan video olarak kayıt yapabilmek adına daha avantajlıdır.

Fraps gibi, Camtasia gibi, pek çok ücretli ve ücretsiz ekran yakalama yazılımı mevcuttur. Ekranı kaydetmek için harici bir yöntem de HDMI arayüzü üzerinden kayıt yapabilen kaydedicilerdir. Bilgisayarın HDMI portundan harici kaydedicilere giriş yapılarak ekran görüntüsünün yüksek bit hızlarında video olarak kaydedilmesi mümkündür. Ancak bu yöntem ek donanım ve çoğunlukla ek bir bilgisayar sistemi gerektirdiğinden daha maliyetli bir yöntemdir.

ResimSekansiKaydet.cs

ResimSekansiKaydet ▸ Update ()

```
1  using UnityEngine;
2  public class ResimSekansiKaydet : MonoBehaviour
3  {
4      //Resim sekanlarını hangi adla kaydedeceğiz?
5      public string klasorAdi = "Plan";
6      //En fazla kaydedilecek kare hızı.
7      public int kareHizi = 25;
8      //resimlerimizin boyutunu büyütmek için kullanacağımız rakam.
9      public int boyutCarpani = 1;
10     //Birden fazla klasör olduğunda kullanacağımız klasör isimleri.
11     private string gercekKlasor = "";
12     void Start()
13     {
14         //Bu komut oyun motorunu gerçek zamandan bağımsız,
15         //belli bir kare hızında çalışmaya zorluyor.
16         Time.captureFramerate = kareHizi;
17         // Bu kısımda kullandığımız klasör adında başka bir klasör varsa
18         //klasör adının sonuna gittikçe büyüyen rakamlar koyuyoruz.
19         gercekKlasor = klasorAdi;
20         int sayi = 1;
21         while (System.IO.Directory.Exists(gercekKlasor))
22         {
23             gercekKlasor = klasorAdi + sayi;
24             sayi++;
25         }
26         // Resimleri kaydedeceğimiz klasörü oluşturuyoruz.
27         System.IO.Directory.CreateDirectory(gercekKlasor);
28     }
29     void Update()
30     {
31         // Resimleri isimlendiriyoruz.
32         var isim = string.Format("{0}/kare {1:D04}.png", gercekKlasor, Time.frameCount);
33         // Resimleri kaydediyoruz.
34         Application.CaptureScreenshot(isim, boyutCarpani);
35     }
36 }
```

Resim 6: Oyun penceresinin resim sekansı olarak kaydedilmesi için gerekli C# kodu, Unity 5.5 İçin.

SONUÇ

Yeni nesil oyun motorları ve Unity oyun motoru görsel etki yapım sürecinin henüz çok yeni bir aracıdır. Bu araştırmada kullanılan kod kütüphaneleri görsel etki üretim sürecinde işlevsel olsa da film üretim sürecinde kullanılmak üzere hazırlanmamışlardır. Bu sebeple görsel etki ekipleri Star Wars: Rogue One örneğinde olduğu gibi kendi yazılım geliştirmelerini yapmaları halinde daha iyi sonuçlar alabilirler. Bununla beraber Unity oyun motoru pahalı sanal stüdyo uygulamalarına ücretsiz bir alternatif olarak yardımcı araçlarla ve geliştirmelerle birlikte, özellikle küçük bütçeli yapımlarda rahatlıkla kullanılabilir. Yine çekim sırasında ön görselleştirme ve referans oluşturacak bir görselleştirme için de kullanılabilir.

Submit Date: 11.04.2017, Acceptance Date: 10.06.2017, DOI NO: 10.7456/10703100/007

Ücretsiz ve kolay ulaşılabilir yeni nesil oyun motorlarıyla gerçekleştirilebiliyor oluşu, oyun motorları programlamaya açık olduğundan ihtiyaçlara cevap verecek güncellemelerin kolaylıkla yapılabilir oluşu, gerçek zamanlı çizilen mekân ya da karakterler sayesinde yaratıcı ekip ve oyuncuların görsel etkileri çekim sırasında görebilmesi, sanal gerçeklik ve stereoskopik film üretimini de mümkün kılması bu yöntemin olumlu yanları olarak sıralanabilir. Olumsuz yanlarını ise; oyun motorlarının film üretimi için henüz yeni bir araç olması ve bu yönde programlama kütüphanelerinin kısıtlı olması, görüntü girişi ve çıkışında kullanılacak kamera hareketlerini oyun motoruna aktaracak profesyonel donanımın oldukça kısıtlı olması olarak sıralayabiliriz.

Bu çalışmaya ek olarak, harici kameradan gelen görüntülerin ve kamera hareketlerinin hareket algılayıcılar ile üç boyutlu sanal uzaya daha kaliteli ve daha özelleştirilebilir bir şekilde aktarılması için çalışmalar yapılması iş akışının profesyonelleşmesi için faydalı olacaktır. Yine oyun motorunda oluşturulan görsellerin sıkıştırmasız, yüksek dinamik aralıklı olarak ve performans sorunları yaşanmadan kaydedilebilmesi için çalışmalar yapılması bu tarz bir görsel etki sürecinin standartlarını yükseltecektir ve yaygınlaşmasını sağlayacaktır. Bu tarz bir görsel etki sürecinin sinemaya estetik yönden yansımaları, Star Wars: Rogue One filmindekine benzer kullanımların artmasıyla yeni araştırma alanları açacaktır.

KAYNAKÇA

Moore, Gordon E. , (1965), "Cramming More Components onto Integrated Circuits", Electronics Magazine, 19 Nisan.
Okun, J.A., Zwerman, S.(2010). The VES Handbook of Visual Effects. Oxford: Focal Press
Polygon.com (2017), http://www.polygon.com/2017/3/1/14777806/gdc-epic-rogue-one-star-wars-k2so Erişim tarihi: 01.03.2017
Tüm web site: www.polygon.com
Tüm web site: www.unity3d.com
Unity Technologies (2017), https://docs.unity3d.com/ScriptReference/WebCamTexture.html Erişim Tarihi: 25.02.2017
Vardar., B. (2006). Sinema ve Televizyon Görüntüsünün Temel Öğeleri. İstanbul: Beta.
Zucconi, A., Lammers K.(2016). Unity 5.x Shaders and Effects Cookbook, Birmingham: Packt.

Submit Date: 11.04.2017, Acceptance Date: 10.06.2017, DOI NO: 10.7456/10703100/007