KARABÜK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

SENIOR PROJECT DERSİ

PROJE RAPORU

PYTHON PROGRAMLAMA DİLİNDE GÖRÜNTÜ İŞLEME YÖNTEMİYLE TEMASSIZ BİLGİSAYAR KONTROLÜ

1810205092 - ABDULLAH FURKAN YEĞİN

2020

İÇİNDEKİLER

BÖLÜM 1: GÖRÜNTÜ İŞLEME UYGULAMALARININ ARAŞTIRILMASI

Bu projede amaç klavye, fare veya başka bir giriş birimi kullanılmadan yalnızca kamera kullanılarak bilgisayar üzerinde istenilen işlemleri gerçekleştirmektir. Bu amaçla çeşitli görüntü işleme teknikleri kullanılarak kamera üzerinden alınan veriler işlenecek ve istenilen komutların bilgisayar tarafından anlaşılıp uygulanması sağlanmaya çalışılacaktır.

Araştırmalarım sonucu karşıma çıkan en yaygın 3 seçenekten birini tercih etmem gerekti. Bunlar Matlab, C++ ile OpenCV ve Python ile OpenCV. Bu üç seçenekten kaynak fazlalığı, kullanım kolaylığı ve görüntüleme teknikleri ve makine öğrenmesi üzerine özel frameworklerin bulunması ve Python'ın programlama dilleri arasındaki popülerliğinden dolayı Python ile OpenCV seçeneğini tercih ettim.

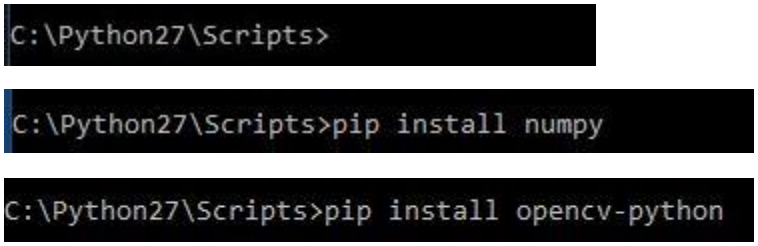
OpenCV (Open Source Computer Vision Library) genellikle gerçek zamanlı görüntüler üzerinden işlemler yapmaya odaklanan, kullanımı kolay, farklı programlama dillerini destekleyen, açık kaynak kodlu bir kütüphanedir. Bu kütüphane birçok kurum ve kuruluş tarafından da çeşitli amaçlarla kullanılmaktadır. Örneğin binalara izinsiz girişlerin tespit edilmesi, yüzme havuzlarındaki boğulma olaylarının kontrolü gibi olaylarda bu kütüphane yaygın olarak tercih edilmektedir.

Python üzerinde kullanılacak kütüphanelerden diğeri NumPy kütüphanesi. Bu kütüphane büyük ve çok boyutlu diziler ve matrisler tanımlayarak bu tanımlamaların üzerinde matematiksel işlemlerin yapılmasına olanak sağlamaktadır.

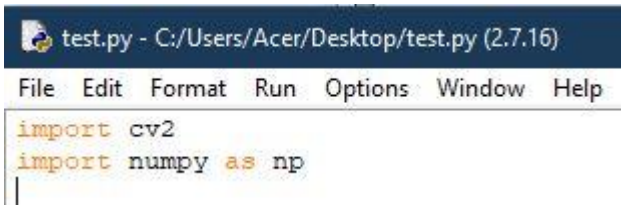
BÖLÜM 2: PYTHON ÜZERİNE GEREKLİ KÜTÜPHANELERİNİN KURULMASI VE ÇAĞIRILMASI

NumPy ve OpenCV kütüphaneleri Python üzerine komut istemi üzerinden yüklendi. Bunun için kurulum paketleri indirip kurmaya yarayan bir paket yönetim sistemi olan ‘pip’ (Package Installer for Python) kullanıldı.

Öncelikle Python’ın disk üzerinde kurulu olduğu klasöre komut istemi üzerinden ulaşarak ‘Scripts’ klasörüne giriş yapıldı. Daha sonra kurulacak kütüphaneler pip vasıtasıyla yüklendi.



Daha sonra başarıyla kurulan kütüphaneler ‘import’ komutuyla IDE içerisinde çağırılarak üzerinde program yazmaya hazır hale getirildi.



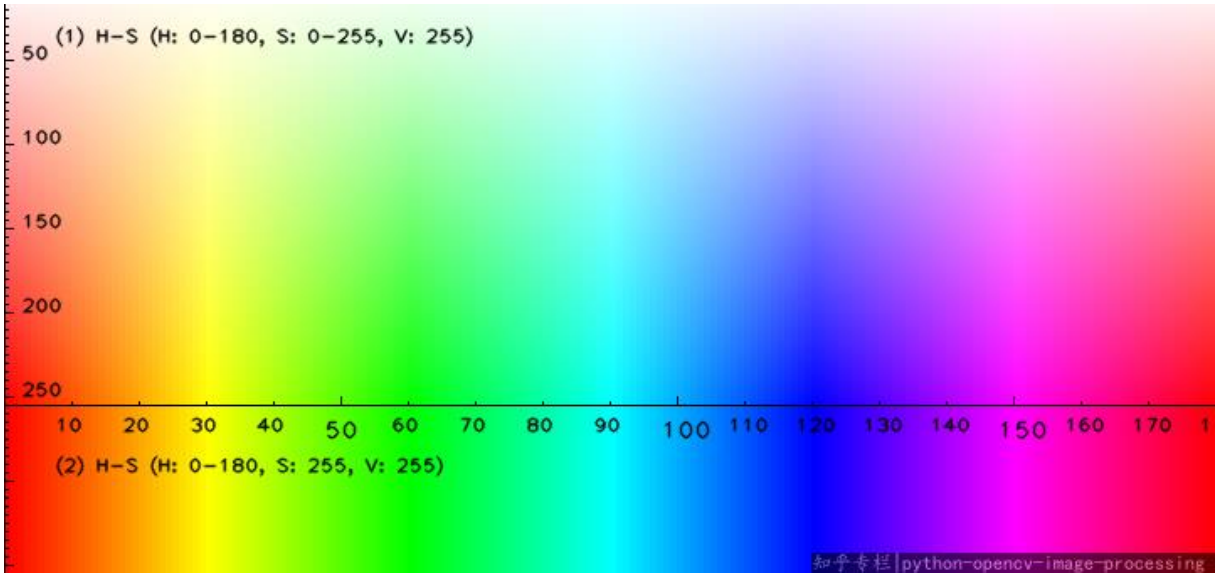
BÖLÜM 3: HSV RENK MODELİ KULLANILARAK İSTENİLEN RENK ARALIĞINA SAHİP NESNELERİN TESPİTİ

HSV renk modeli, renkleri RGB renk modeli gibi kırımızı, yeşil ve mavi renklerin birleşimi ile oluşturmak yerine her renge ait bir tanım bulundurmaktadır. Bu tanımlar üçe ayrılır:

1.kısım olan 'Hue' kısmı rengin tonunu belirlemek için kullanılır. OpenCV içerisinde kullanılan HSV renk modelinde Hue değeri 0 ile 180 arasında değer alır. Diğer uygulamalarda maksimum renk değeri 360'a kadar çıkabilmektedir.

2.kısım olan 'Saturation' bölümünde rengin yoğunluğu belirlenir. 0 ile 255 arasında değer alır. 255 değeri rengin çok yoğun olduğunu ifade eder.

3.kısım olan 'Value' değeri parlaklığı belirtir. Yine 0 ile 255 arasında değer alır. 0 tamamen siyah, 255 tamamen beyaz demektir.



Python üzerinden istenilen renklere sahip cisimler belirlenirken direk renk değeri girilmek yerine belirli bir renk aralığı girilmesi tercih edilmektedir. Bu şekilde başarılı olma olasılığı artmaktadır. Bunun için program içerisinde istenilen renk aralığına dair alt ve üst limitler girilir. Örnekte kırmızı renk için hue değeri 0 ile 10 arasında, saturation değeri ise 100 ile 255 arasında, value değeri ise 100 ile 255 arasında tutulmuştur.

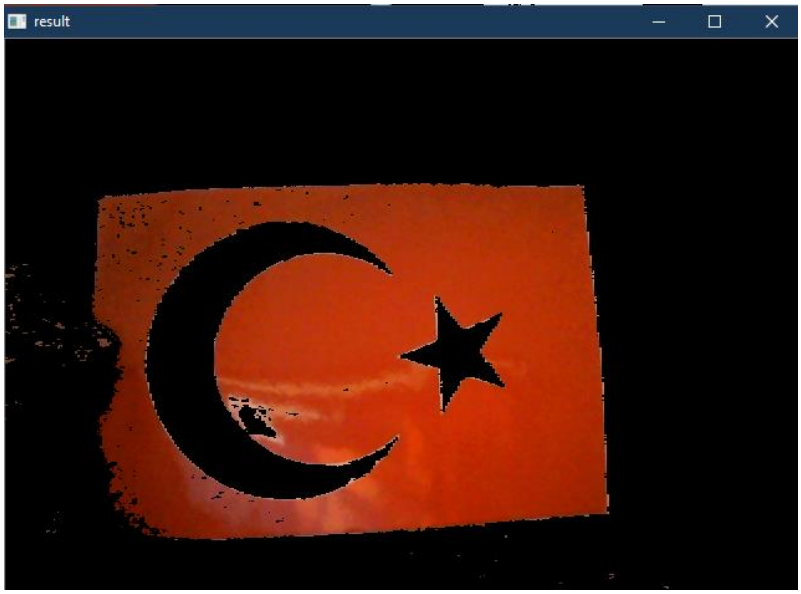
```
lowerLimit=np.array([0,100,100])
upperLimit=np.array([10,255,255])
```

OpenCV fonksiyonlarından ‘inRange’ ile bu aralıktaki renk değerleri tespit edilmiştir.

```
mask=cv2.inRange(hsv,lowerLimit,upperLimit)
```

Daha sonra iki görüntünün dizi bitlerini ‘and’ işlemine tabi tuttuğumuzda sadece istediğimiz renk aralığındaki cisimlerin orijinal hallerini görebilmekteyiz.

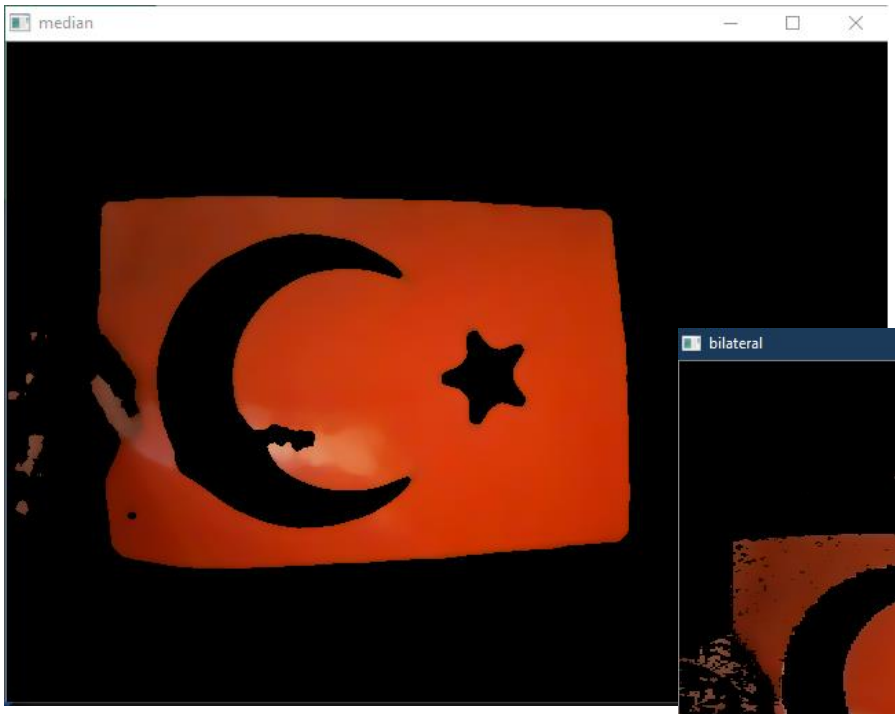
```
result=cv2.bitwise_and(frame,frame,mask=mask)
```

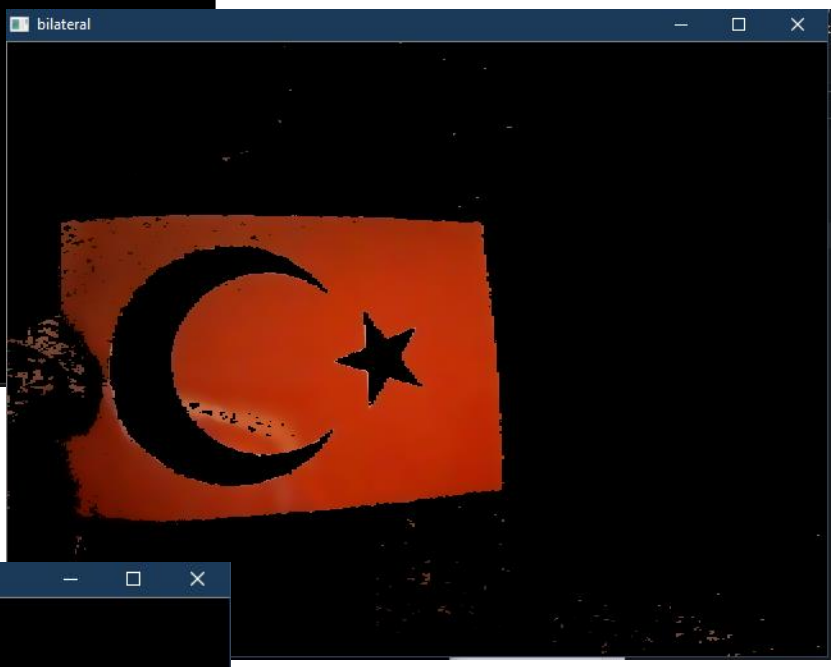


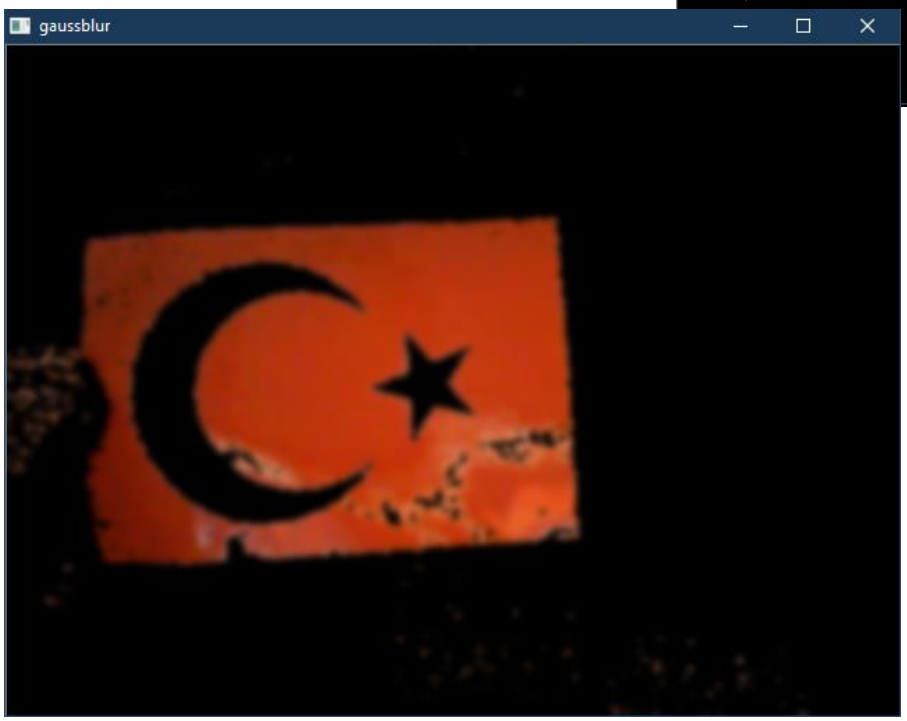
Daha sonra istenilirse sonuç üzerinde de tekrar değişiklikler yapılabilir.

Örnek olarak median, bilateral ve gaussianblur efektleri çıktı üzerine uygulanmıştır. Bu algoritmalar OpenCV kütüphanesi üzerinden çağırılmaktadır.

```
blur=cv2.GaussianBlur(result,(15,15),0)
median=cv2.medianBlur(result,15)
bilateral=cv2.bilateralFilter(result,15,75,75)
```







BÖLÜM 4: İSTENİLEN RENK ARALIĞINA SAHİP İKİ CİSİM İLE FARE KONTROLÜ

Bu aşamada Python IDE ile giriş aygıtları arasında bağlantıyı sağlayabilmek için Python'a ait 'pynput' kütüphanesi kullanıldı. Bu kütüphane sayesinde klavye ve fare girişleri yapılabilmekte veya yapılan girişlerin görüntülenebilmesi mümkün olmaktadır. Bu kütüphaneyi kullanabilmek için konsol üzerinden pip komutu ile yükleme işleminin gerçekleştirildi.

```
C:\Python27\Scripts>pip install wxpython

C:\Python27\Scripts>pip install pynput
```

Yükleme işleminden sonra Python üzerinden import edilir. Ayrıca ekran boyutunun elde edilebilmesi için wx kütüphanesi de eklendi.

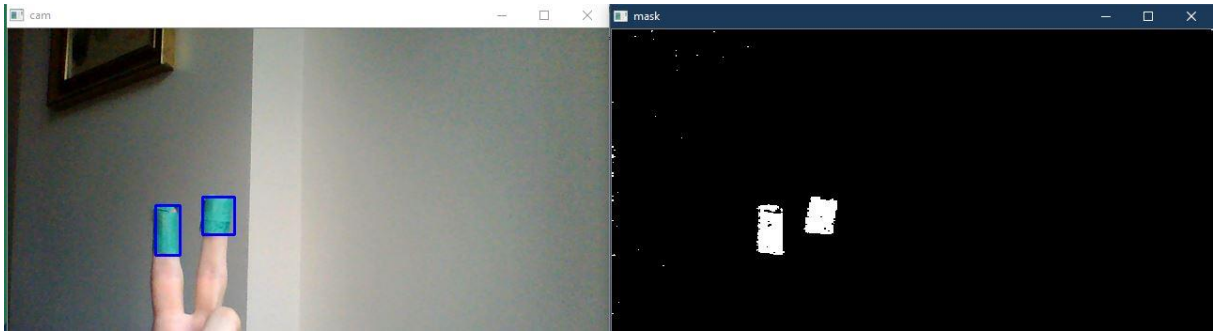
```
from pynput.mouse import Button,Controller
import wx

mouse=Controller()
app=wx.App(False)
(sx,sy)=wx.GetDisplaySize()
```

Ekranda istenen renk aralığına sahip iki nesne belirdiğinde bu nesnenin sınırlarının tespit edilebilmesi için findContours fonksiyonu kullanıldı. Ardından bu sınırların belirginleştirilmesi için rectangle fonksiyonuyla dikdörtgen içine alındı.

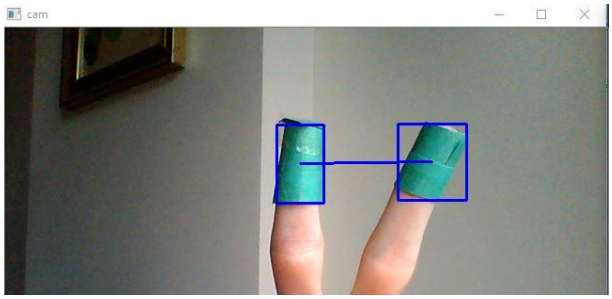
```
_,conts,h = cv2.findContours(maskFinal.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
if(len(conts)==2):

        x1,y1,w1,h1=cv2.boundingRect(conts[0])
        x2,y2,w2,h2=cv2.boundingRect(conts[1])
        cv2.rectangle(img,(x1,y1),(x1+w1,y1+h1),(255,0,0),2)
        cv2.rectangle(img,(x2,y2),(x2+w2,y2+h2),(255,0,0),2)
```



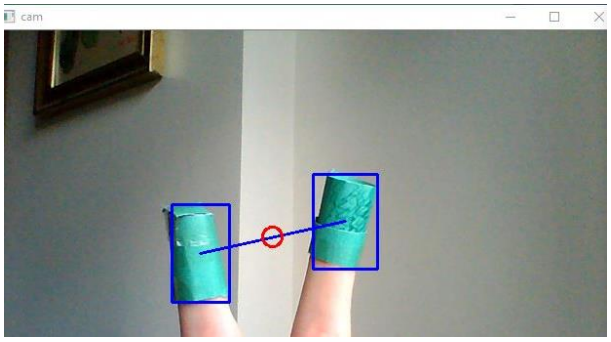
Uygulamada amaç iki nesnenin birbirine yakınlığı kullanılarak tıklama yapılması olduğundan iki nesnenin arasında bir çizgi oluşturularak bağlantı kuruldu.

```
cx1=x1+w1/2
cy1=y1+h1/2
cx2=x2+w2/2
cy2=y2+h2/2
cx=(cx1+cx2)/2
cy=(cy1+cy2)/2
cv2.line(img,(cx1,cy1),(cx2,cy2),(255,0,0),2)
```



Daha sonra imleç hareketlerinde merkez olarak kullanılmak üzere çizginin ortasına bir daire eklendi.

```
cv2.circle(img, (cx,cy),2,(0,0,255),2)
```



Fare imlecinini hareket ettirmek için daha önce ‘GetDisplaySize’ ile elde edilen boyutu kamera çözünürlüğüne bölüp bu değeri pynput özelliği olan mouse.position’a atayarak farenin konumu değiştirildi.

```
mouse.position=(sx-(mouseLoc[0]*sx/camx),mouseLoc[1]*sy/camy)
while mouse.position!=(sx-(mouseLoc[0]*sx/camx),mouseLoc[1]*sy/camy):
    pass
```

Son olarak fare tıklama hareketi için yine pynput kütüphanesi kullanılır. Bu fonksiyonların çalışma koşulu iki nesne arasındaki uzaklığa bağlıdır. Tıklama yapılması için ne kadar mesafe olması gerektiği belirlenerek if koşulu eklendi. Bu koşulların while döngüsü içine alınarak sürekli sorgulanması sağlandı.

```
while True:

if(abs((w*h-openw*openh)*100/(w*h))<22):
        pinchFlag=1
        mouse.press(Button.left)
        openx,openy,openw,openh=(0,0,0,0)



  mouse.press(Button.left)
```

Koşullar sağlanırsa fare tıklaması sağlanır. Bu durumu belli etmek için tıklama gerçekleştiğinde büyük bir daire ile tıklama yapıldığı gösterildi.

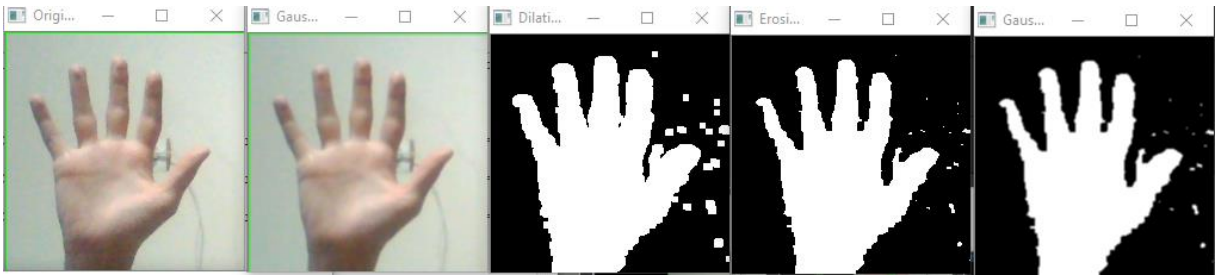
```
cv2.circle(img, (cx,cy),(w+h)/4,(0,0,255),2)
```

İki nesne birbirinden uzaklaştığında if koşulu false değerini gönderir ve fare release komutuyla tıklama sona erdirilerek serbest bırakılır.

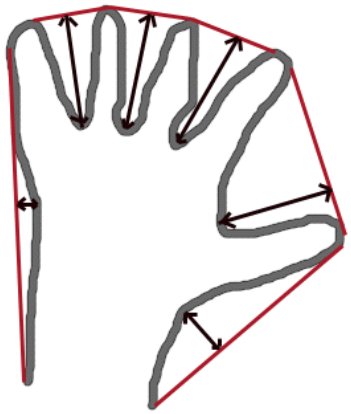
```
mouse.release(Button.left)
```

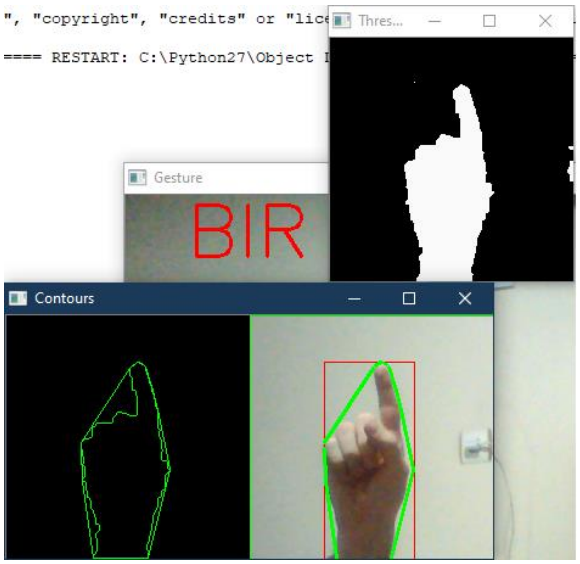
BÖLÜM 5 : UYGUN IŞIK KOŞULLARI ALTINDA ELİN TESPİT EDİLMESİ

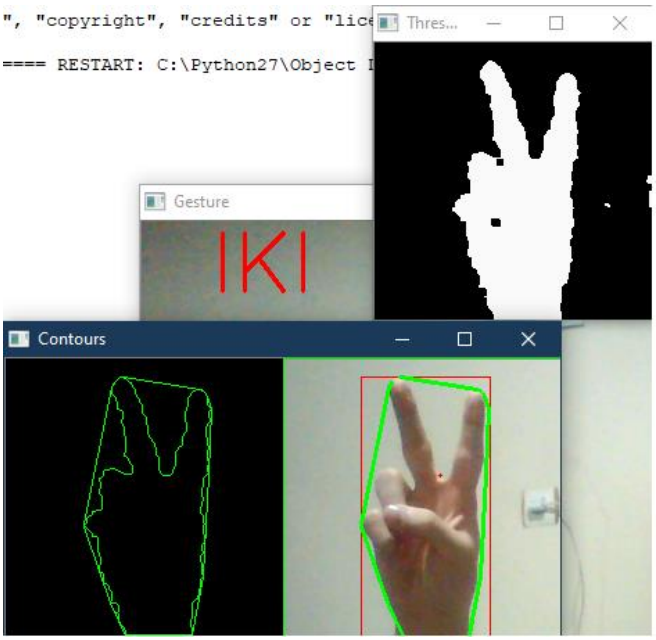
Daha önce kullanılan iki noktanın birbirine uzaklığına göre oluşturulan konsept yerine elin tamamını algılayan bir konsepte geçiş yapıldı. Kameranın eli tanıması için ten rengine ait HSV renk değerleri programa eklendi. Ancak gündüz saatlerinde ortamdaki ışıklandırmanın değişiklik göstermesinden dolayı istenilen sonuç elde edilemedi. Bundan dolayı akşam sabit ışıklandırma ile el tespiti yapıldı. El tespiti yapıldıktan sonra oluşan gürültüyü azaltmak için çeşitli filtreler kullanıldı.

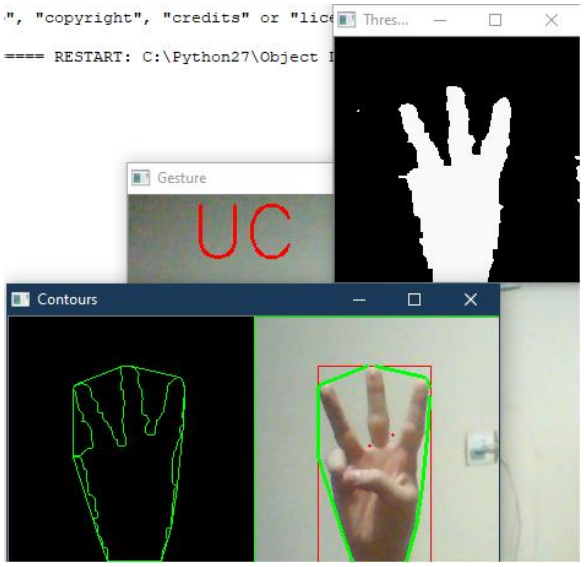


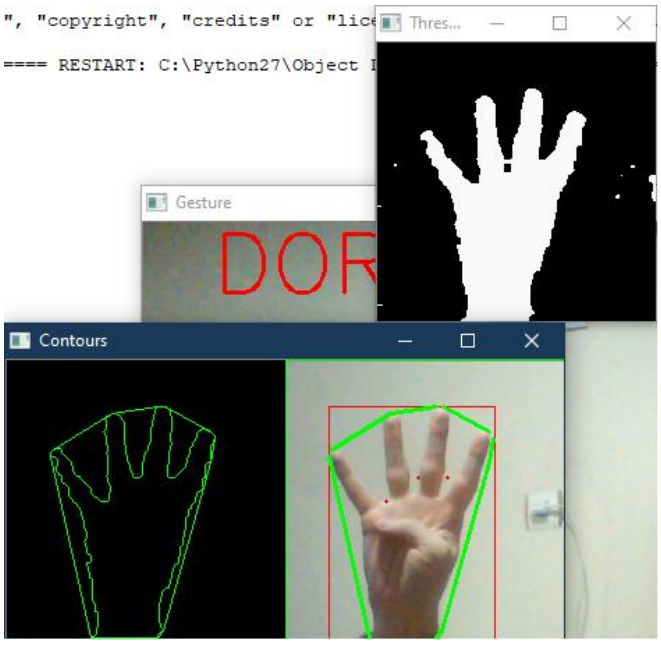
Daha sonra ele fonksiyon atayabilmek için elin mevcut şeklini tanıyıp ona göre çıktı verebilen kod yazıldı. Böylece parmaklar arasındaki mesafe kullanılarak bilgisayarın el hareketlerine tepki verebilmesi sağlanmış oldu.

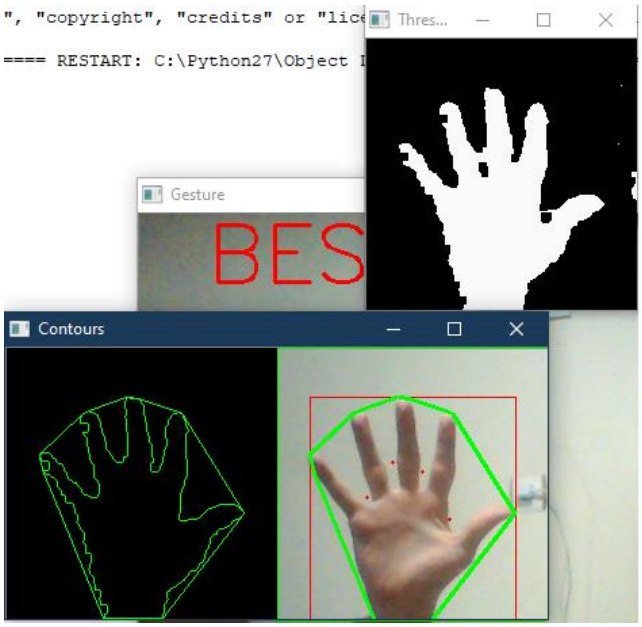



", "copyright", "credits" or "lic
==== RESTART: C:\Python27\Object
Thres...
Gesture
BIR
Contours


", "copyright", "credits" or "lic
==== RESTART: C:\Python27\Object
Thres...
Gesture
IKI
Contours


", "copyright", "credits" or "lic
==== RESTART: C:\Python27\Object
Thres...
Gesture
UC
Contours


", "copyright", "credits" or "lic
==== RESTART: C:\Python27\Object
Thres...
Gesture
DOR
Contours


", "copyright", "credits" or "lic
==== RESTART: C:\Python27\Object
Thres...
Gesture
BES
Contours

Bir sonraki aşamada her bir durum için farklı eylemler atandı. Örneğin 5 durumunda ekranın ve elin ortasında birer nokta oluşturuldu. Bu iki nokta arasındaki konum farkı ile (sol-üstünde, sol-altında, sağ-üstünde, sağ-altında bulunması) fare imlecinin hareketi kesintili de olsa sağlandı. Ardından 2 durumuna fare sol tıklama komutu atandı ve yaklaşık olarak istenilen konuma getirilen fare imleci ile tıklama gerçekleştirildi.

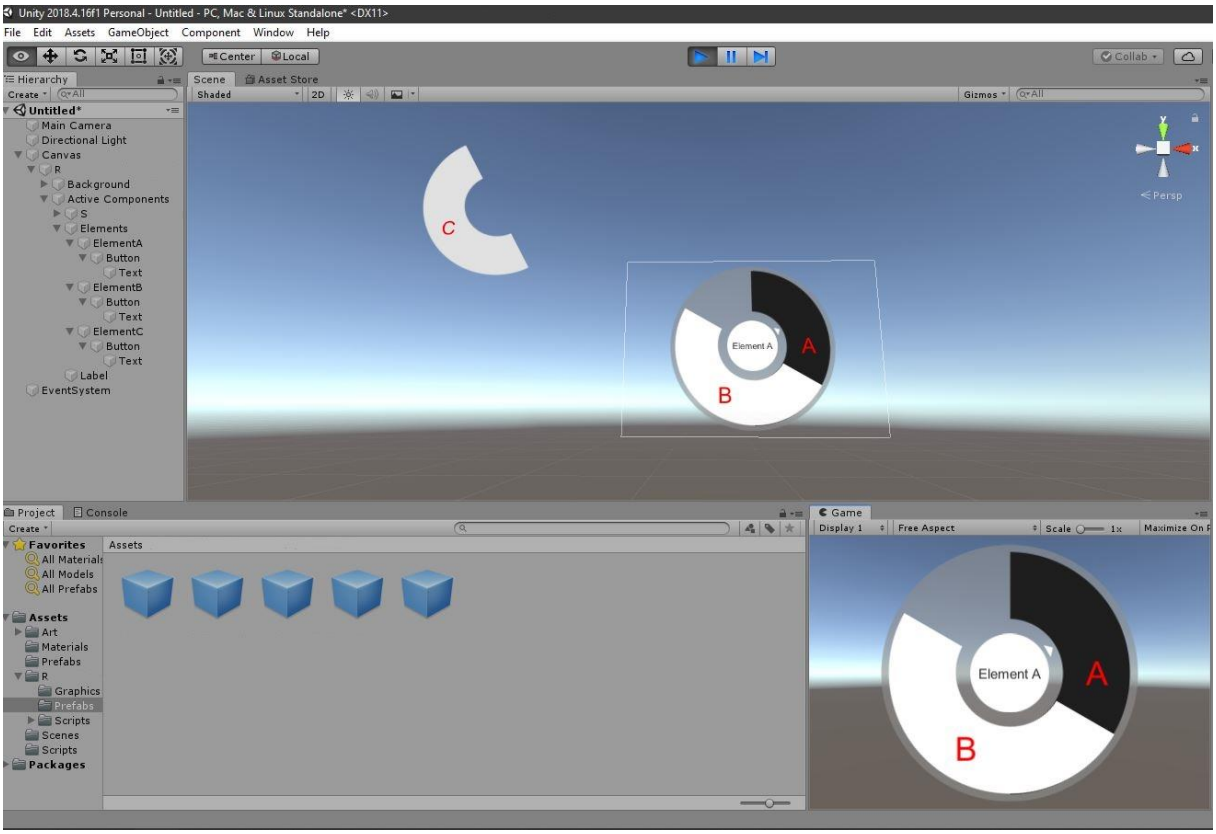
Farklı bir uygulamada ise 1-2-3-4 durumlarına yön tuşları atanarak sanal ortamda temassız olarak araba sürüldü. Ancak ani refleks gerektiren durumlarda bu atamanın pek de başarılı olmadığı gözlendi. Ayrıca zaman zaman eli tanıyamama gibi sorunlar ortaya çıktı.

Sonuç olarak farklı ışık koşullarında farklı sonuçlar elde edildiğinden ten renginin bilgisayarın rahatça algılayabileceği bir renk olmadığı ve kameranın farklı ışık koşullarında mevcut renkleri, olduğundan farklı renkte veya tonda algılayıp istenmeyen komutları istemsiz bir şekilde çalıştırabildiğinden kameralar tarafından en net algılanabilen renk olan yeşil rengin bir önceki yöntemle kullanılmaya devam edilmesine karar verildi.

BÖLÜM 6: UNITY OYUN MOTORU KULLANILARAK ARAYÜZ TASARIMININ YAPILMASI

Fare hareketlerinin başarılı bir şekilde gerçekleştirilmesine rağmen çözünürlüğü yüksek olan bilgisayarlarda kullanımın zor olacağı ve bilgisayar kullanım bilgisi yetersiz olan kişilerin de bilgisayarı kullanabilmeleri için bir arayüz tasarlanması gerektiği düşünüldü. Bu arayüz tasarımını yapabilmek için birçok seçenek değerlendirilmiş olup en sonunda Unity oyun motoru kullanılmasına karar verildi.

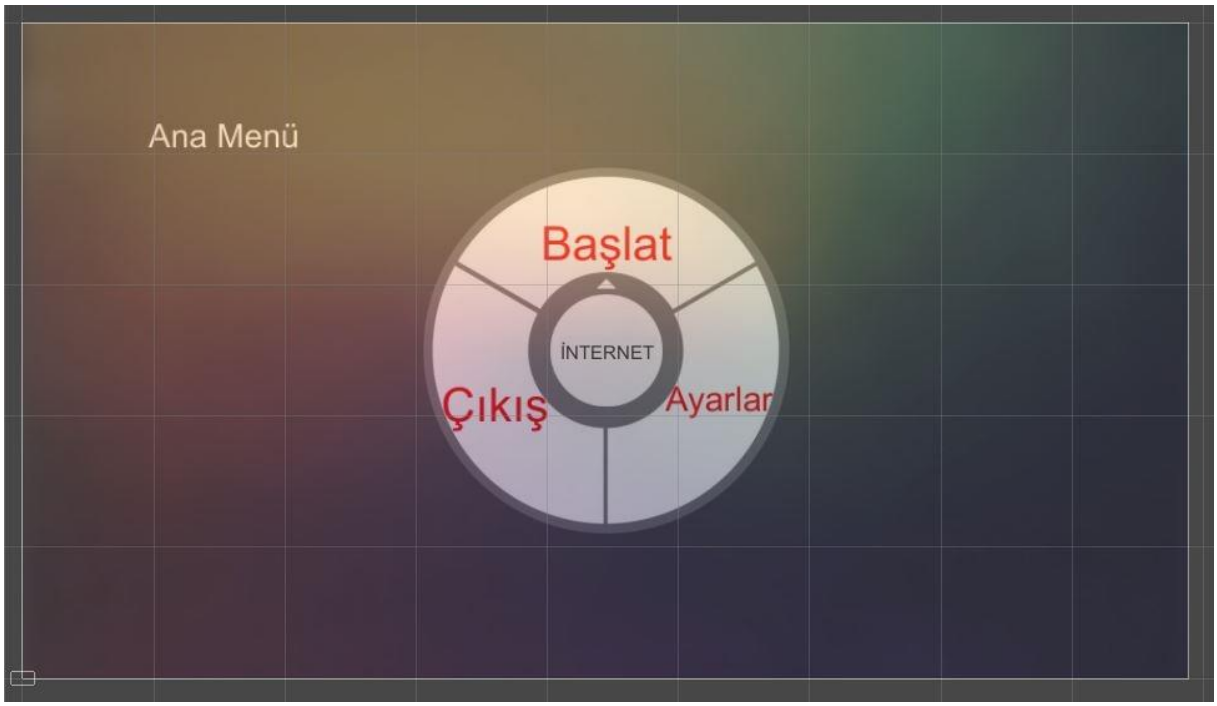
Bu oyun motoru genelde kompleks oyunlar oluşturulmak için kullanılmakta olup tarafımca sadece menü tasarımlarında kullanılan kısmı kullanılacaktır.

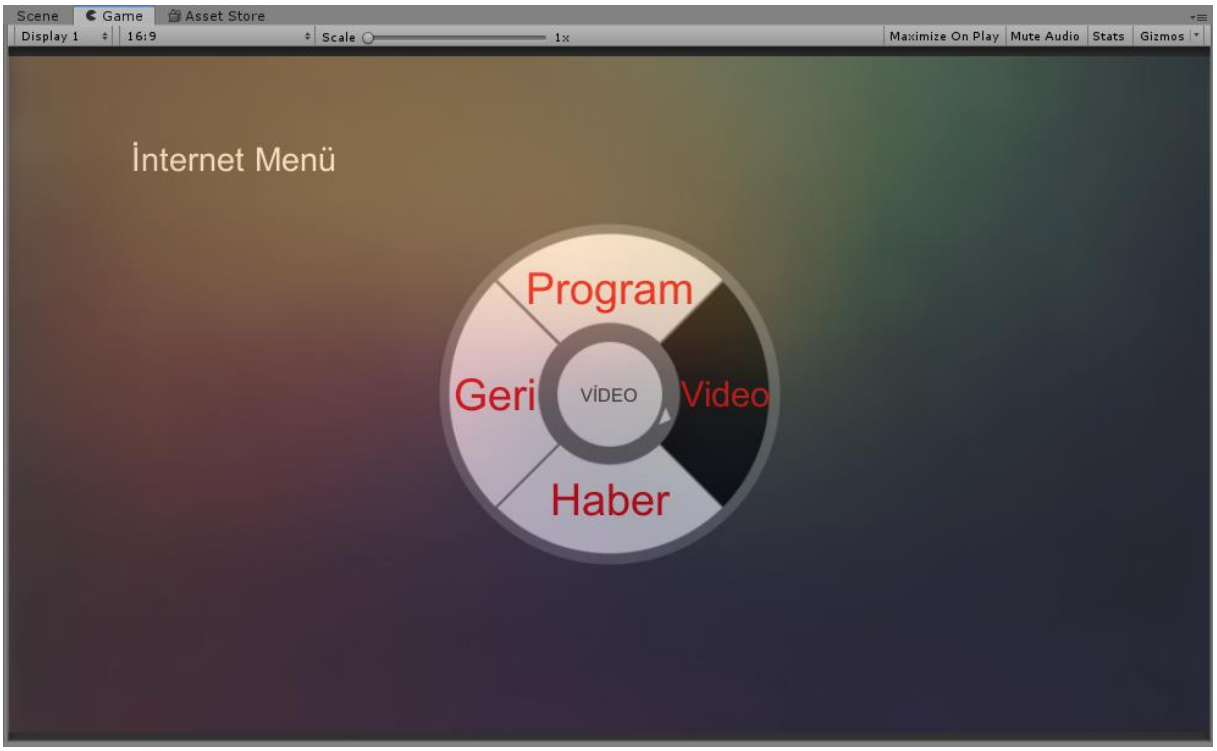


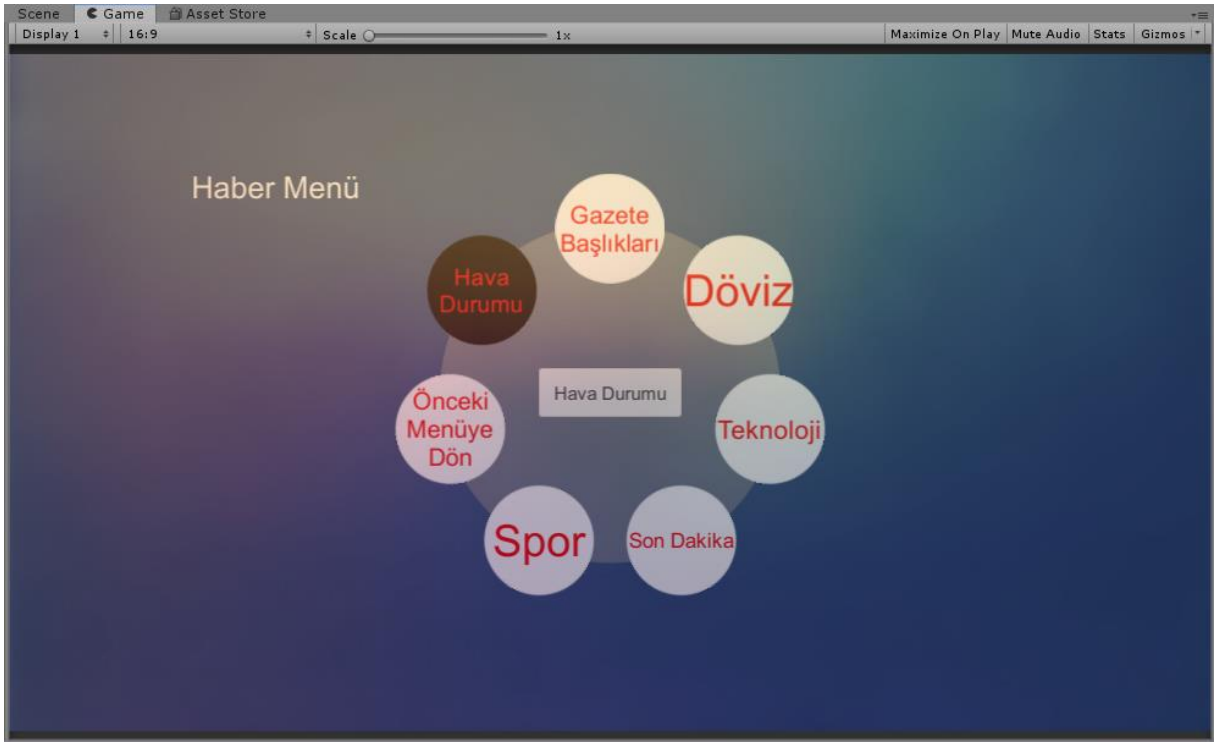
Bu tasarımda radial menü assetleri kullanılmıştır.

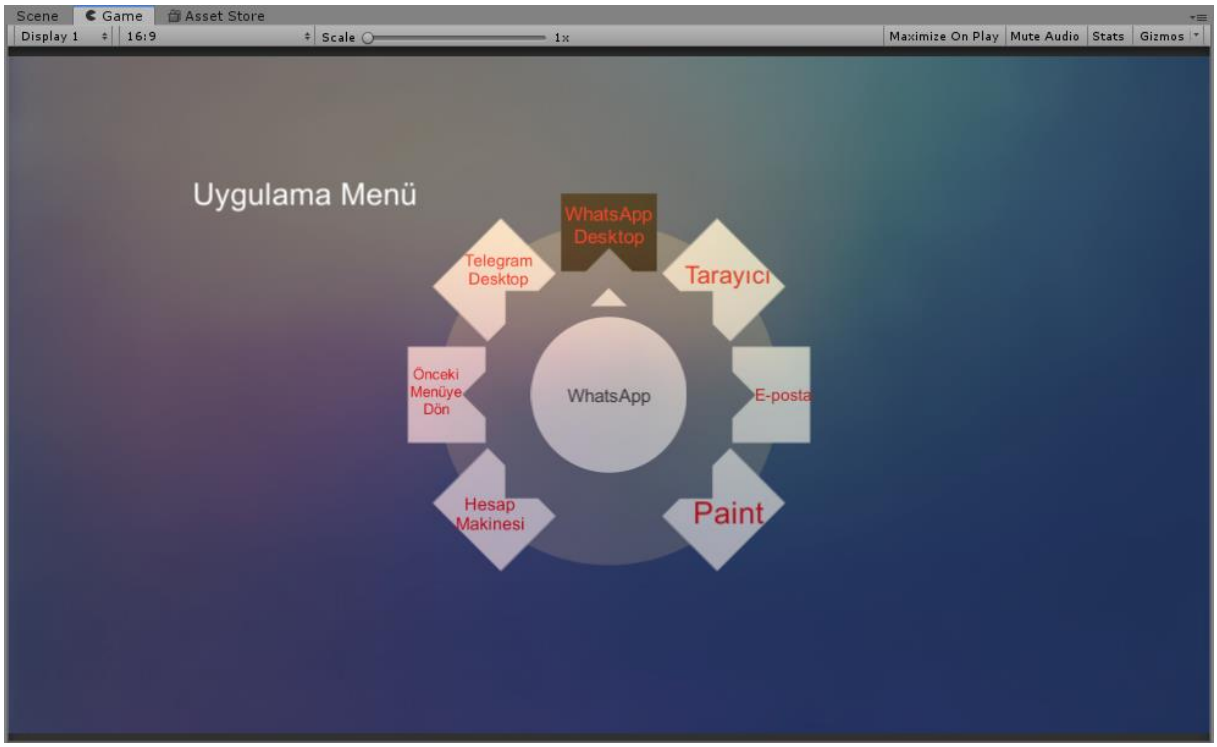
Unity C# uygulama dilini kullanır. Bu dil kullanılarak yönlendirmeler ve sahneler arası geçiş gerçekleştirilmiştir. Diğer görsel tasarımlar Unity'nin sahip olduğu arayüzden faydalanılarak hazırlanmıştır.

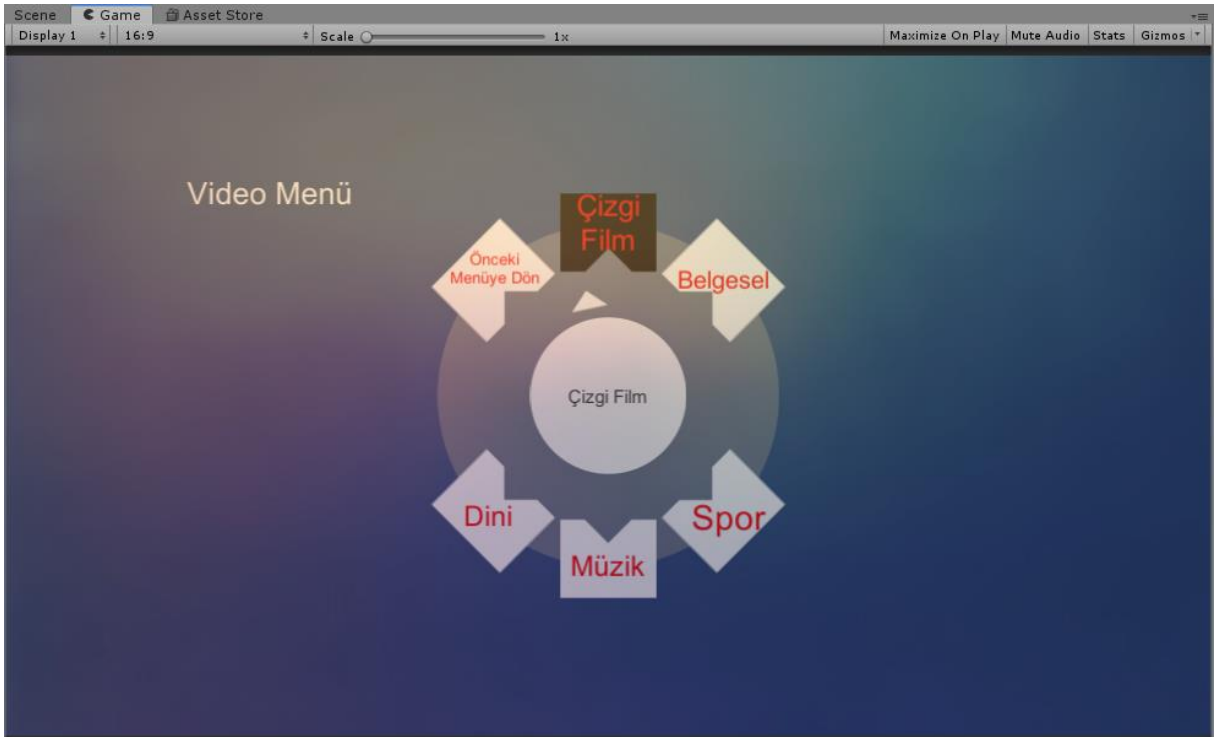
Bu menülerdeki butonlar kullanılarak istenilen uygulamaya, web sayfasına veya ayarlara erişilmesi hedeflenmiştir.




Scene
Game
Asset Store
Display 1
16:9
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos
İnternet Menü
Program
Geri
VİDEO
Video
Haber


Scene
Game
Asset Store
Display 1
16:9
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos
Haber Menü
Gazete Başlıkları
Hava Durumu
Döviz
Hava Durumu
Önceki Menüye Dön
Teknoloji
Spor
Son Dakika





Sonuç olarak kamera üzerinden bilgisayara komut atanmış ve oluşturulan arayüz sayesinde istenilen uygulama veya web sayfasına ulaşım sağlanmıştır. Seçeneklerin yetersiz gelmesi durumunda ekran klavyesi yardımıyla istenilen sayfaya ulaşım sağlanabilmektedir.

KAYNAKÇA

https://pythonprogramming.net/blurring-smoothing-python-opencv-tutorial/ (Haziran/2020)

https://stackoverflow.com

https://www.geeksforgeeks.org/python-detect-polygons-in-an-image-using-opencv/ (Haziran/2020)

https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html (Haziran/2020)

https://www.learnopencv.com/convex-hull-using-opencv-in-python-and-c/ (Haziran/2020)

https://www.pyimagesearch.com/2016/02/08/opencv-shape-detection/ (Haziran/2020)

https://pysource.com/2018/04/05/object-tracking-with-mean-shift-opencv-3-4-with-python-3-tutorial-29/ (Haziran/2020)

https://alloyui.com/examples/color-picker/hsv.html (Haziran/2020)

https://assetstore.unity.com/packages/tools/gui/radial-menu-framework-50601

https://www.youtube.com/watch?v=kdLM6AOd2vc&list=PLS1QulWo1RIa7D1O6skqDQ-JZ1GGHKK-K (Haziran/2020)

https://pyautogui.readthedocs.io/en/latest/ (Haziran/2020)

https://www.elektrikport.com/universite/opencv-nedir/21537