# T.C.
# FIRAT ÜNİVERSİTESİ
## MÜHENDİSLİK FAKÜLTESİ
### BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ


## 2021-2022 BAHAR DÖNEMİ

### BİTİRME PROJESİ 3. ve 4. HAFTA RAPORU

**MEDYA OYNATICI ARAYÜZÜ VE**
**GERÇEK ZAMANLI EL TESPİTİ**

**185260012 – DİNÇER ŞİPKA**
**185260009 – EMİRHAN AKTAŞ**
**185260019 - SELİM CAN ERKAN**

## ÖZET

Bitirme projesi için 3. ve 4. Hafta basit bir medya oynatıcı uygulaması ve gerçek zamanlı olarak el tespiti gerçekleştirdik. El tespitini gerçekleştirdikten sonra örnek bir komut tespiti gerçekleştirdik.

# MEDYA OYNATICI ARAYÜZÜ

Arayüz geliştirme için PyQt5 kütüphanesini kullandık. PyQt, Qt araç setinin python için olan bir eklentisi. Python ile grafiksel kullanıcı arayüzlü programlar oluşturmamızı sağlıyor. İlk olarak kütüphanenin kurulumunu gerçekleştirdik. Kurulum için aşağıdaki pip kodunu kullandık.

Kütüphane Kurulumu İçin;

```
pip install pyqt5
```

Uygulama için bir python dosyası oluşturduk. Bu dosya içinde uygulama arayüzü için gerekli bileşenleri ve fonksiyonları oluşturduk.

```python
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QHBoxLayout, QVBoxLayout, QSlider, QFileDialog
from PyQt5.QtMultimedia import QMediaPlayer, QMediaContent
from PyQt5.QtMultimediaWidgets import QVideoWidget
from PyQt5.QtGui import QIcon, QPalette
from PyQt5.QtCore import Qt, QUrl
import sys


class Window(QWidget):
    def __init__(self):
        super().__init__()

        #Pencere'nin genel özellikleri
        self.setWindowTitle("Media Player")
        self.setWindowIcon(QIcon("./icons/app.ico"))
        self.setFixedSize(750, 500)

        #Arkaplan rengi ayarlamak için
        palette = self.palette()
        palette.setColor(QPalette.Window, Qt.black)
        self.setPalette(palette)
```

*Şekil 1 – Arayüz Constructor Metodu*

Şekil-1 üzerinde gösterilen kodda pencerenin genel özelliklerini tanımladık.

Medya oynatıcı için gerekli olan bileşenleri createPlayer metodu ile oluşturuyoruz. Kullandığımız bileşenler PyQt5 kütüphanesi altında bulunuyor. Bileşenlere işlevleri connect ile bağlıyoruz.

```
def createPlayer(self):
    self.mediaPlayer = QMediaPlayer(None, QMediaPlayer.VideoSurface)

    self.openButton = QPushButton("OPEN")
    self.openButton.clicked.connect(self.openFile)

    self.playButton = QPushButton()
    self.playButton.setEnabled(False)
    self.playButton.setIcon(QIcon("./icons/play.ico"))
    self.playButton.clicked.connect(self.playMedia)
    self.playButton.clicked.connect(self.mediaStateChanged)
```

*Şekil 2 - createPlayer Metodu*

Arayüz bileşenlerini Layout kullanarak ana penceremize ekledik. Kontrol paneli için yatay, medya oynatıcı ve kontrol panelini alt alta göstermek için dikey Layoutlar kullandık.

```
horizontalBox = QHBoxLayout()
horizontalBox.setContentsMargins(0,0,0,0)
horizontalBox.addWidget(self.openButton)
horizontalBox.addWidget(self.playButton)
horizontalBox.addWidget(self.timeSlider)
horizontalBox.addWidget(self.volumeButton)
horizontalBox.addWidget(self.volumeSlider)

videoWidget = QVideoWidget()

verticalBox = QVBoxLayout()
verticalBox.addWidget(videoWidget)
verticalBox.addLayout(horizontalBox)
```

*Şekil 3 - Layout Kodları*

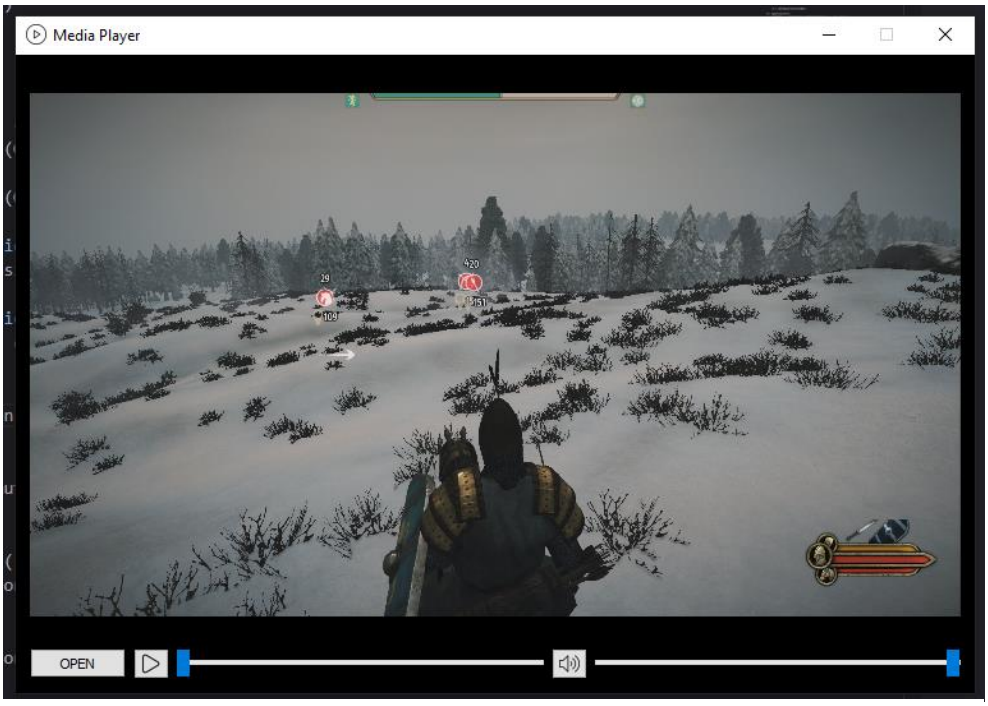
Videoları arayüz içerisinden dosya seçici ile seçtik. Seçtiğimiz dosyanın yolunu kullanarak arayüzde bulunan mediaPlayer bileşinine QMediaContent ile medya ekledik.

```
def openFile(self):
    fileName, _ = QFileDialog.getOpenFileName(self, "Open Media")

    if fileName != '':
        self.mediaPlayer.setMedia(QMediaContent(QUrl.fromLocalFile(fileName)))
        self.playButton.setEnabled(True)
```

*Şekil 4 - Dosya açma metodu*

Uygulamayı çalıştırdıktan sonra bir arayüz bileşenlerini ve bu bileşenlerin işlevlerini bir video ile test ettik. Kodların tamamını Github üzerine yükledik.




*Şekil 5 - Medya Oynatıcı Arayüz*

# GERÇEK ZAMANLI EL TAKİBİ

Gerçek Zamanlı El Takibi yapabilmek için geçtiğimiz hafta kurmuş olduğumuz OpenCV ve MediaPipe kütüphanelerinin kullanımına devam edildi. Import ettiğimiz cv2 ve mediapipe kütüphaneleriyle el takibinin kameradan belirlenmesi sağlanacaktır.

Kamerayı kullanmak için VideoCapture sınıfında nesne oluşturduk. Daha sonrasında kamera görüntüsünü göstereceğimiz pencere için ekran boyutu ayarladık. Daha önce el tespiti için kullandığımız MediaPipe ile oluşturduğumuz sınıftan bir nesne oluşturduk.

HandControl.py > ...

```
1   import cv2, HandModule
2
3   capture = cv2.VideoCapture(0)
4
5   camWidth, camHeight = 640, 480
6
7   capture.set(3, camWidth)
8   capture.set(4, camHeight)
9
10  detector = HandModule.HandDetector(detectionCon=0.65, maxHands=1)
```

*Şekil 6 - VideoCapture Nesnesi*

Gerçek zamanlı görüntü yakalamak için sonsuz döngü oluşturduk. Sonsuz döngü içinde ilk önce oluşturduğumuz VideoCapture nesnesi ile görüntüyü yakaladık. Daha sonra el tespiti ve elimizdeki indekslerin pozisyonlarını bulduk.

```
13    while True:
14        success, img = capture.read()
15        img = cv2.flip(img, 1)
16        img = detector.findHands(img)
17        lmList, bbox = detector.findPosition(img, draw=False)
```

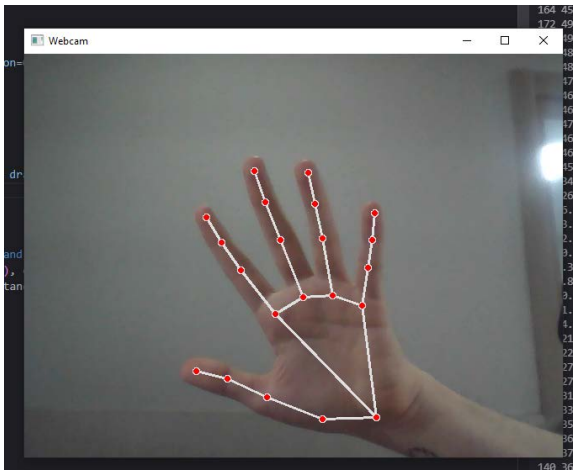
*Şekil 7 - El tespiti ve El indeksleri*

Ses kontrolünü sağlamak için oluşturacağımız komut için örnek bir kontrol mekanizması gerçekleştirdik. Bu kontrole göre işaret parmağı ve baş parmak haricindeki parmaklarımız aşağıya dönük olduğu takdirde ses kontrolü komutu aktif oluyor. Daha sonra bu geliştirmeleri test ettik.

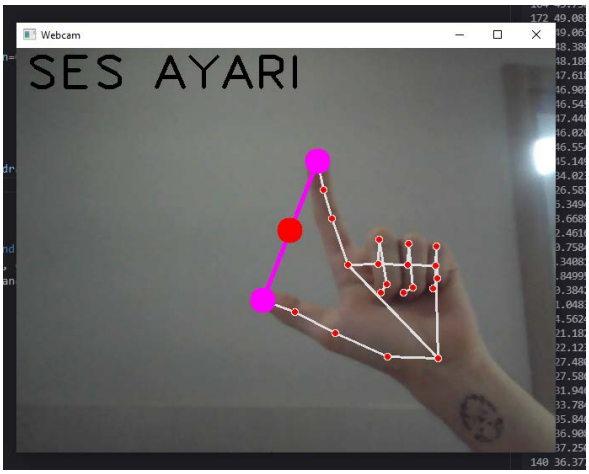
```
if len(lmList) != 0:
    length, img, array = detector.findDistance(lmList[4],lmList[8], img)

    if length < 35:
        cv2.putText(img, "KAPALI", (10, 50), cv2.FONT_HERSHEY_PLAIN, 4, (0, 0, 0), 3)
    else:
        cv2.putText(img, "ACIK", (10, 50), cv2.FONT_HERSHEY_PLAIN, 4, (0, 0, 0), 3)
```

*Şekil 8 - Ses kontrolü mekanizması*




*Örnek 1: - El ve el indekslerinin tespiti*




*Örnek 2: - Ses kontrolü*

Proje kodlarına aşağıda bulunan Github sayfasından ulaşabilirsiniz.

https://github.com/Bitirme-Projesi-Grubu/El-Hareketleriyle-Medya-Oynatici-Kontrolu