YAPILAN İŞ : Projede kullanılabilecek ögelerden biri olan kamera özelliği kullanılmıştır. Aynı zamanda kamera ile fotoğraf çekimi ve çekilen fotoğraları kaydetme özelliği kullanılmıştır. Q_INVOKABLE yapısı kullanılmıştır. Ayrıca ses ve resim ekleme özellikleri kullanılmıştır.

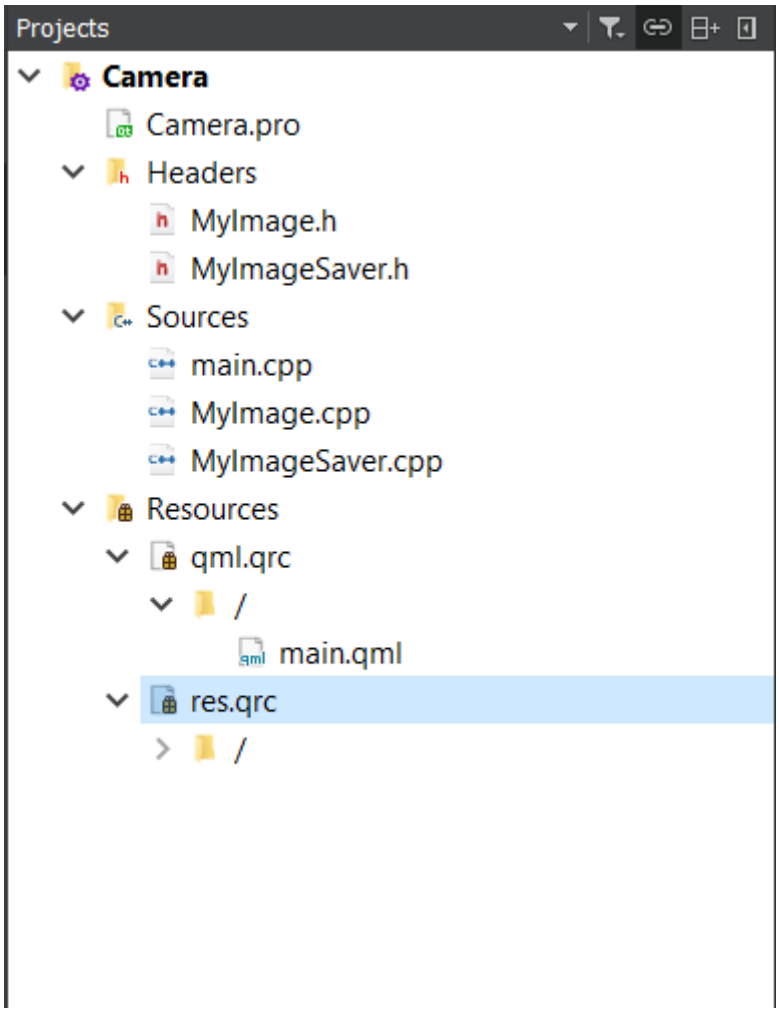
NOT : Bu Örnekte INVOKABLE Yapısı ile yazılmıştır. Bu tamamlandıktan sonra INVOKABLE yerine Property ile yazılmaya başlanmıştır.

Bu Örnekte çok fazla daha önce kullanılmayan yapı olduğundan bazı komutlar net bir şekilde anlatılamamıştır.

NOT 1 : İlk olarak QT Quick Application Empty projesi oluşturulmuştur. Ardından projede kullanılacak ses ve görüntüleri eklemek için gerekli JPG , PNG ve WAV dosyaları indirilmiştir.

NOT 2 : İndirilen dosyaları projede kullanılmak için Qt Resource File türünde res.qrc adlı dosya projeye eklenmiştir. Daha sonrasında "/" adında bir prefix eklenip bu prefixin içine dosyalar seçilmiştir. Buradaki önemli husus kullanılacak dosyalar proje klasörünün içinde bulunmalıdır.

NOT 3 : Daha sonrasında sırasıyla MyImage ve MyImageSaver adlı C++ Class'ları oluşturulmuştur. Bu oluşturmalar sonucunda proje dosyaları GÖRSEL 1 'deki gibi gözükmektedir.



GÖRSEL 1 -> Projedeki Dosyalar

```
#ifndef MYIMAGE_H
#define MYIMAGE_H

#include <QByteArray>
#include <QString>

class MyImage
{
public:
    MyImage(const QString &id = "" , const QByteArray &data = QByteArray());
    void setData (const QByteArray &data);
    void setId (const QString &id);
    QByteArray data() const;
    QString id ()const;
private:
    QByteArray mData;
    QString mId;
};

#endif // MYIMAGE_H
```

GÖRSEL 2 -> MyImage.h

NOT 4 : İlk olarak QByteArray ve QString kütüphaneleri include edildi. QByteArray kullanılmasının sebebi proje içerisinde fotoğraf işleme olacağından bunu byte'lara çevirmek gerekmektedir. Bu sebeple arrayların veya dosyaların byte dizisi olarak tutulması için bu kütüphane tanımlanmıştır.

NOT 5 : mData ve mId private olarak tanımlanmıştır. Daha sonrasında public olarak gerekli fonksiyon tanımlamaları yapılıp bu fonksiyonların içeriği MyImage.cpp dosyasında yazılmıştır.

NOT 6 : Constructor içinde data ve id erişimi sağlamak için tanımlamalar yapılmıştır. Bunun sonrasında data ve id methodları tanımlanmıştır.

```cpp
#include "MyImage.h"

MyImage::MyImage(const QString &id, const QByteArray &data)
{
    mId = id;
    mData = data;
}

void MyImage::setData(const QByteArray &data)
{
    mData = data;
}

void MyImage::setId(const QString &id)
{
    mId = id;
}

QByteArray MyImage::data()const
{
    return mData;
}

QString MyImage::id() const
{
    return  mId;
}
```

GÖRSEL 3-> MyImage.cpp

NOT 7 : MyImage.cpp dosyasında klasik olarak private olarak tanımlanam variable'lara erişim ve başka dosyalarda kullanmak için gerekli eşitlemeler ve return yapısı kullanılmıştır.

```cpp
#ifndef MYIMAGESAVER_H
#define MYIMAGESAVER_H

#include <QObject>
#include <QList>
#include "MyImage.h"

class MyImageSaver : public QObject
{
    Q_OBJECT

public:
    MyImageSaver(QObject *parent = nullptr);
    Q_INVOKABLE bool savePicture(const QString &id ,QObject *objectImage);
    Q_INVOKABLE bool writePictures();

private:
    QList<MyImage> mImages;
};

#endif // MYIMAGESAVER_H
```

GÖRSEL 4 -> MyImageSaver.h

NOT 8 **:** İlk olarak QObject ve QList kütüphaneleri tanımlanmıştır. Daha sonra private olarak QList kütüphanesi kullanılarak MyImage classı içindeki data yapısında göre bir liste oluşturulmuştur. Bu listeye de mImages referred edilmiştir.

NOT 9 : Burada savePicture ve writePictures methodları qml dosyalarında çağırılabilmesi için Q_INVOKABLE yapısı kullanılmıştır.

```cpp
#include "MyImageSaver.h"
#include <QFile>
#include <QImage>
#include <QDebug>
#include <QBuffer>
#include <QStandardPaths>
#include <QQuickItemGrabResult>

MyImageSaver::MyImageSaver(QObject *parent)
    : QObject {parent}
{

}
bool MyImageSaver::savePicture(const QString &id, QObject *objectImage)
{
    const int nImages = mImages.size();
    for (int ix = 0; ix< nImages; ++ix)
    {
        if(mImages.at(ix).id() == id)
        {
            return false;
        }
    }

    QQuickItemGrabResult *item = static_cast<QQuickItemGrabResult *>(objectImage);
    QByteArray imageData;
    QBuffer buffer(&imageData);
    if (item->image().save(&buffer, "PNG"))
    {
        MyImage image;
        image.setData(imageData);
        image.setId(id);
        mImages.append(image);
        return true;
    }
    return false;

}
```

GÖRSEL 5-> MyImageSaver.cpp (Satır 1-38)

NOT 10 : Satır 16 'da yapılan işlemde MyImageSaver.h dosyası içinde oluşturulan mImages adli listenin size'ına erişim sağlanmıştır. Liste içindeki eleman miktarı nImages adlı variable'a atanmıştır.

NOT 11 : Satır 17 de başlayan for döngüsündeki mantık ise eğer mImages adlı listenin herhangi bir elemanının id'si başka bir elemanın id'si ile aynı olursa bu id için false döndür yani kabul etme.

NOT 12 : Satır 25 ile 36 arasında image'i oluşturup kaydetme işlemi yapılmaktadır. Burada ilk olarak satır 25 'de object(item)kapsülleme işlemi yapılır. Daha sonra bunu image'i byte yapısına dönüştürmek için imageData adlı QByteArray class'ına ait variable oluşturulur. Daha sonra bu imagedata'ya erişim sağlayabilmek için bir ara birim yapısı olan buffer tanımlanır.

NOT 13 : Satır 28 'de yapılan işlem mantık olarak Eğer üretilen item, image'e aitse bunun bufferını (yani datasını(imagedata)) kullanılarak PNG olarak kaydet. İf içinde ise image oluşturulur. Bunun imageDatası ve id 'si setlenir. Daha sonrasında oluşturulan image'i mImages listesine kaydet.

```cpp
bool MyImageSaver::writePictures()
{
    const int nImages = mImages.size();
    for (int ix= 0; ix < nImages; ++ix){
        QFile file;
        QString filename = mImages.at(ix).id().split("/").last() + ".png";
        QString directory = QStandardPaths::writableLocation(QStandardPaths::DesktopLocation);
        QString path = directory + "/" + filename;
        file.setFileName(path);
        if (file.open(QIODevice::WriteOnly))
        {
            if (file.write(mImages.at(ix).data()) > 0)
            {
                file.flush();
                file.close();
            }
            else
            {
                qDebug() << "Error" << file.errorString();
            }
        }
        else
            qDebug()<<"Error "<<file.errorString();
            return false;
    }
    return true;
}
```

GÖRSEL 6 -> MyImageSaver.cpp(Satır 39-65)

NOT 14 : writePictures methodunda ise ilk olarak tekrardan nImage oluşturulur. Daha sonrasında boş bir file oluşturulur. Ardından bu file'a id ve türü belirli olan bir filename oluşturulur. Daha sonrasında ise bunun oluşturulacağı directory ve path belirlenir.

NOT 15 : Dosya kaydetmesi yapıldıktan sonra bu dosyasının doğru olup olmadığı kontrol edilir.

```
1  import QtQuick 2.15
2  import QtQuick.Controls 2.5
3  import QtMultimedia 5.9
4  import QtQuick.Layouts 1.3
5  import QtQuick.Controls 1.4 as C1
6  import QtQuick.Controls.Material 2.2
7
8  ApplicationWindow {
9      id : root
10     width: 640
11     height: 480
12     visible: true
13     title:"Camera"
14
15     Material.theme: Material.Dark
16     Material.accent: Material.Green
17
18     property bool rounded: roundedSwitch.position === 1.0
19     property bool adapt: true
20     property var picturesModel: []
21     property string cameraState: turnOnSwitch.position === 1.0 ? "Camera Enable " : "CameraDisabled"
22
23     SoundEffect{
24         id : buttonSound
25         source: "qrc:/button.wav"
26     }
27
28     SoundEffect{
29         id : captureSound
30         source: "qrc:/cameraappCapture1.wav"
31     }
32
33     SoundEffect{
34         id : beepSound
35         source: "qrc:/beep.wav"
36     }
37
38     function addPicture(source) {
39         var image = {
40             "id" :source,
41             "source": source
42         };
43         picturesModel.push(image);
44         root.picturesModelChanged();
45     }
46     Camera {
47         id :camera
48         digitalZoom: zoomSlider.value
49         imageProcessing.whiteBalanceMode: CameraImageProcessing.WhiteBalanceFlash
50         exposure.exposureCompensation: -1.0
51         exposure.exposureMode: Camera.ExposurePortrait
52         flash.mode : Camera.FlashRedEyeReduction
53         imageCapture.onImageCaptured: {
54             addPicture((preview))
55         }
56
57     }
58     C1.SplitView{
59         anchors.fill: parent
60         orientation: Qt.Horizontal
61         Item{
62             id : cameraControls
63             width: 200
64             height: parent.height
65
66             GroupBox{
67                 id:controls
68                 label:Label{
69                     text: "CONTROLS"
70                     font.pointSize: 15
71                     font.bold: true
72                 }
73                 width: parent.width
74                 height: parent.height / 2
75                 Column {
76                     anchors.fill: parent
77                     spacing: 1
78                     Switch {
79                         id:turnOnSwitch
80                         position : 0.0
81                         text: "CAMERA"
82                         onPositionChanged:{
83                             buttonSound.play();
84                             if(position == 1.0){
85                                 camera.start();
86                             }
87                             else{
88                                 camera.stop();
89                             }
90
91                         }
92                     }
```

NOT 16 : Satır 18-22 arasında program içindeki bazı özellikler tanımlandı.

NOT 17 : SoundEffect komutları ile daha önce projeye eklenen ses dosyaları koda eklendi.

NOT 18 : addPicture fonksiyonunda image oluşturulur. Bu özelliklerde oluşturulan picturesModel listesine push edilir ve liste değiştirilir.

NOT 19 : Camera yapısı Qt documentation sitesinden alınmıştır.

NOT 20 : SplitView yapısı fotoğraf çekim sonucunda fotoğraflara ekranda erişememe durumu olmaması için yapılmıştır.

NOT 21 : Satır 68 de başlık olarak konulan CONTROL Label'ı bulunmaktadır.

NOT 22 : Column yapısı içindeki Switch kendi içinde Camera adında bir labelı bulunmaktadır. Temel işlevi pozisyonu değiştirildiğinde camerayı açıp kapamayı kontrol etmektir Ayrıca pozisyon değiştiğinde buttonSound'ı çalmaktır.

GÖRSEL 7 -> main.qml (Satır 0-92)

```
                Switch{
                    id :roundedSwitch
                    text:"CIRCULAR"
                    position: 1.0
                    onPositionChanged: {
                        buttonSound.play();
                    }
                }
                Image {
                    source: "qrc:/save.png"
                    sourceSize.width: 55
                    sourceSize.height: 55
                    fillMode: Image.PreserveAspectCrop
                    width: 55
                    height: 55
                    MouseArea{
                        anchors.fill: parent
                        onClicked: {
                            beepSound.play();
                            MyImageSaver.writePictures();
                        }
                    }
                }
            }
        }
        VideoOutput{
            anchors.left: parent.left
            anchors.right: parent.right
            anchors.top:controls.bottom
            height: parent.height / 2 -50
            source: camera
            visible: true
          Rectangle{
              id : captureButton1
              width: 55
              height: 55
              radius: 50
              color: "red"
              border.width: 2
              border.color: "lime"
              opacity: 0.6
              anchors.horizontalCenter: parent.horizontalCenter
              anchors.bottom: parent.bottom
              MouseArea{
                  anchors.fill :parent
                  onPressed: {
                      captureButton1.color = "lime"
                      camera.imageCapture.capture();
                      captureSound.play();
                  }
                  onReleased: {
                      captureButton1.color = "red";
                  }
              }
          }
          Rectangle{
              anchors.fill: parent
              color: "black"
              visible: cameraState === "CameraDisabled"
              Image{
                  source: "qrc:/disabled.jpg"
                  sourceSize.width: parent.width / 2
                  sourceSize.height: parent.height /2
                  width: parent.width /2
                  height: parent.height /2
                  fillMode: Image.PreserveAspectFit
                  anchors.centerIn: parent
              }
          }
        }
        Slider{
            id:zoomSlider
            orientation: Qt.Horizontal
            from:0
            to:camera.maximumDigitalZoom /10
            value: 1.0
            anchors.left:parent.left
            anchors.right: parent.right
            anchors.bottom: parent.bottom
        }
    }
```

GÖRSEL 8 -> main.qml (Satır 93-174)

NOT 22 : Satır 93 ‘de başlayan Switch yapısı ekranda çekilen fotoğrafların üstüne Mouse ile gelindiğinde ortada circular oluşturmak için düzenlenmiştir.

NOT 23 : Satır 101 ‘de oluşturulan Image yapısı içinde öncelikle save.png eklemesi yapılır. Ardından Mouse Area ile bu resme tıklandığında fotoğrafı kaydetme işlemi yapmak için writePictures() fonksiyonu çağırılır. Ayrıca beepSound play edilir.

NOT 24 : VideoOutput yapısı ekranda kameradaki durumu aktif olarak göstermek için yazılmıştır. (Source olarak camera gösterilmiştir.)

NOT 25 : Mouse Area ile fotoğraf çekimi gerçekleştirilmektedir. Basıldığında çember lime renginde olup imageCapture çağırılmaktadır. Ayrıca captureSound çağrılmaktadır.

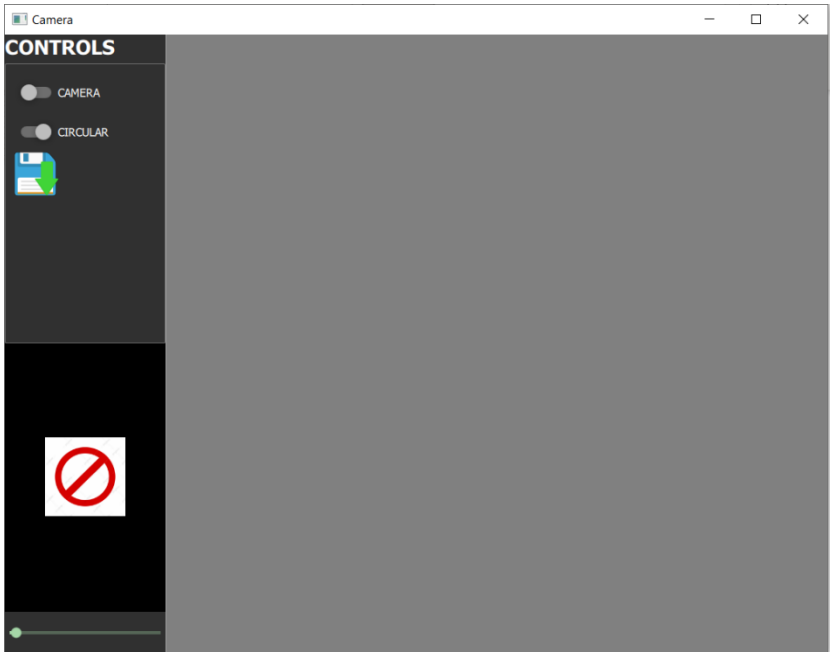
NOT 26 : Satır 149’daki Rectangle yapısında ise daha önce oluşturduğumuz Camera Slider yapısının pozisyonu 0.0 olduğunda Camera Disable edilecek ve resim olarak disabled.jpg gösterilecektir.

NOT 27 : Satır 164 ‘deki Slider’ın amacı program açıldığında kameranın altında bulunan slider kameranın zoom yapmasını sağlayacaktı ancak zoom olayı yapılamamaktadır.

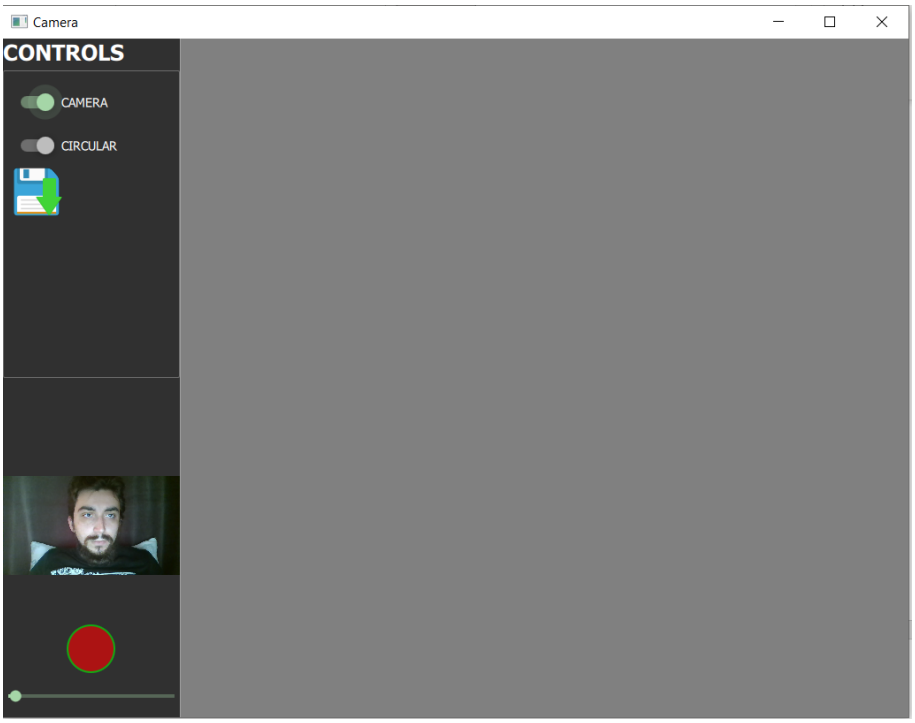
```
175     Item{
176         id:pictures
177         height: parent.height
178         Rectangle{
179             anchors.fill: parent
180             color: "grey"
181             GridView{
182                 id: grid
183                 anchors.fill:parent
184                 cellWidth: parent.width /5
185                 cellHeight: parent.height /5
186                 model :picturesModel
187                 delegate: Rectangle{
188                     property bool showPicture: true
189
190                     id:rect
191                     width: grid.cellWidth
192                     height: grid.cellHeight
193                     color: showPicture ? "transparent":"white"
194                     radius: rounded ? 200:0
195
196                     Text {
197                         anchors.centerIn: parent
198                         visible:!rect.showPicture
199                         font.pointSize: 20
200                         text: {
201                             var txt = modelData.id;
202                             txt = txt.substring(txt.indexOf("_")+1);
203                             return txt
204                         }
205                     }
206                     Image {
207                         id: image
208                         visible: rect.showPicture
209                         anchors.centerIn: parent
210                         source: modelData.source
211                         sourceSize.width: 512
212                         sourceSize.height: 512
213                         width: parent.width * 0.95
214                         height: parent.height * 0.95
215                         fillMode: Image.PreserveAspectCrop
216
217                         layer.enabled: rounded
218                         layer.effect: ShaderEffect{
219                             property real adjustX: adapt ? Math.max(width/height, 1) : 1
220                             property real adjustY: adapt ? Math.max(1/ width/height,1) : 1
221
222                         }
223                         Component.onCompleted: {
224                             image.grabToImage(function(result){
225                                 MyImageSaver.savePicture(modelData.id,result);
226
227                             });
228                         }
```

NOT 28 : Satır 175 ‘deki İtem yapısı ekranda fotoğrafların gösterilmesi amacıyla yazılmıştır.

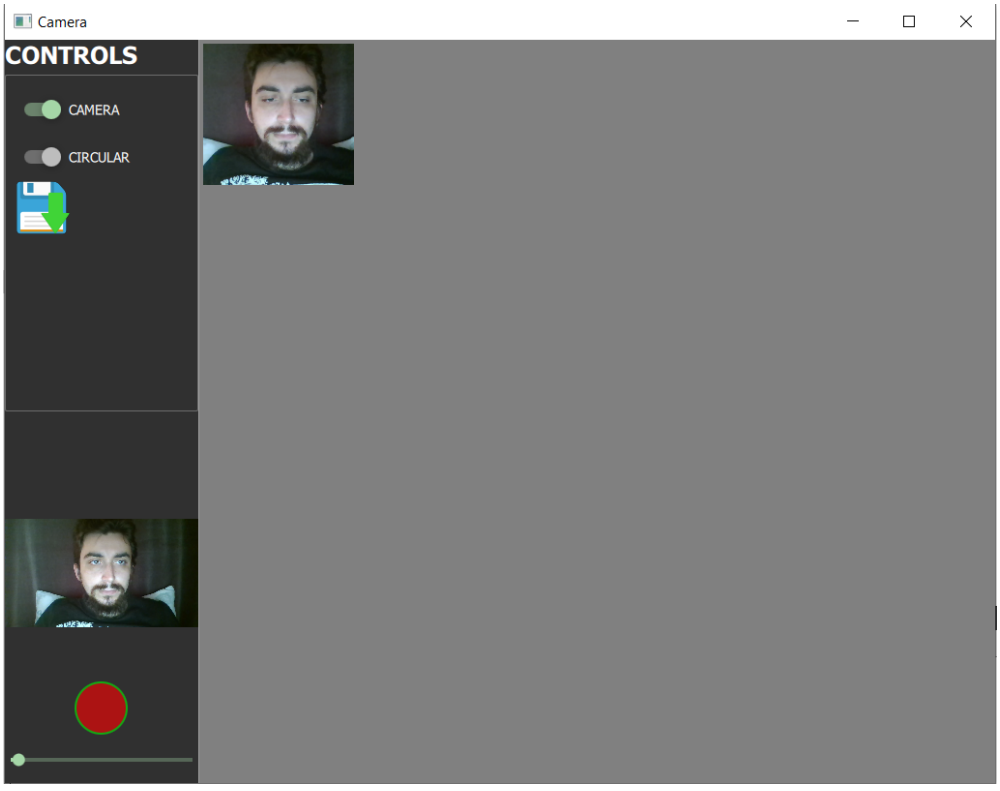
GÖRSEL 9 -> main.qml (Satır 175-228)



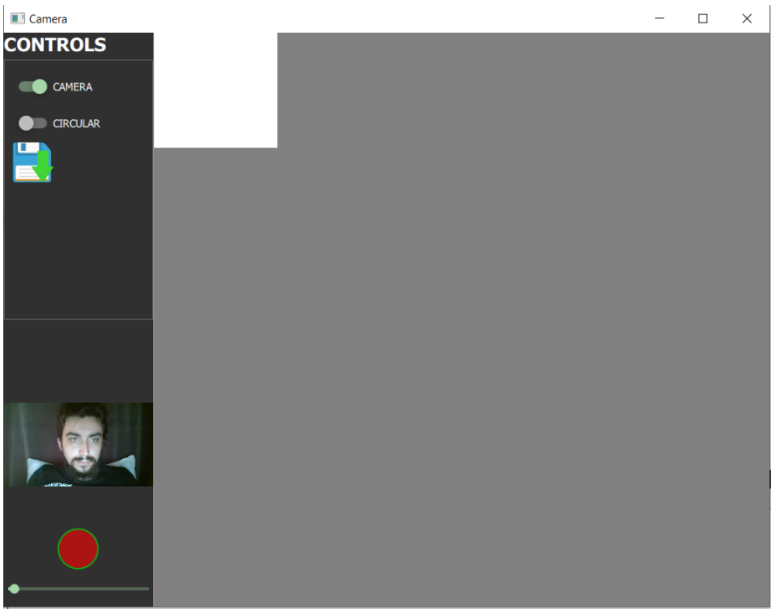
GÖRSEL 10 -> Program Çalıştırıldığında



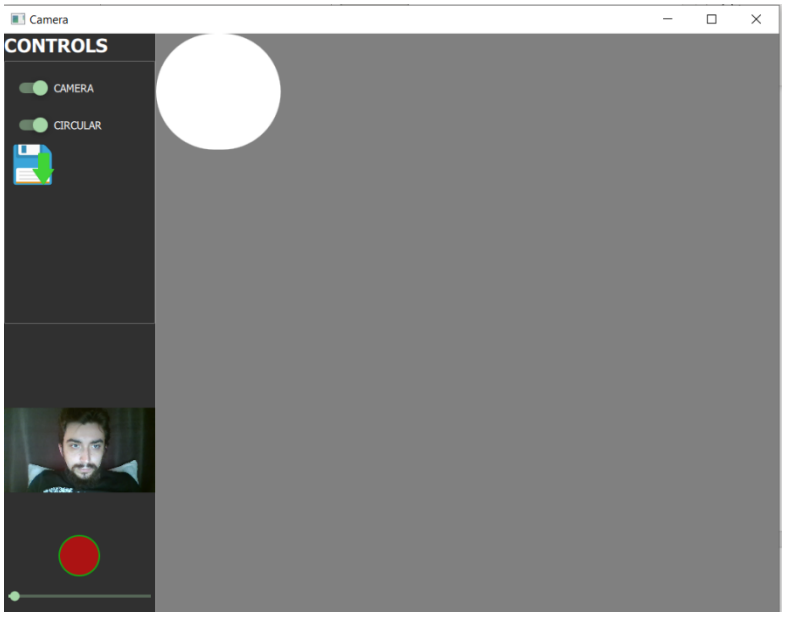
GÖRSEL 11-> Kamera Aktifleştiğinde



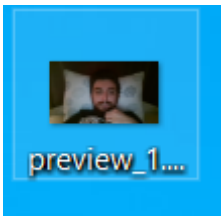
GÖRSEL 12 -> Fotoğraf Çekimi yapıldığında



GÖRSEL 13 -> Circular Açılmadığında



GÖRSEL 14 -> Circular Açıldığında



GÖRSEL 15 -> Save image'ine basıldığında

YAPILAN İŞ : Projede kullanılmak üzere daha önce belirlenen özelliklerden biri olan saati analog olarak gösterme uygulamasıdır.(Digital Saat Uygulaması Daha Önce Yapılmıştır.)

```
1  import QtQuick 2.0
2
3  Item {
4      id: id_root
5      property int value: 0
6      property int granularity: 60
7
8      Rectangle {
9          width: 1
10         height: id_root.height * 0.6
11         color: "red"
12         anchors {
13             horizontalCenter: id_root.horizontalCenter
14         }
15         antialiasing: true
16         y: id_root.height * 0.05
17     }
18
19     rotation: 360/granularity * (value % granularity)
20     antialiasing: true
21 }
22
```

GÖRSEL 1 -> SecondNeedle.qml

NOT 1 : Bu dosyada analog saatteki saniye imleci ayarlanmıştır.

NOT 2 : Özellik olarak başlangıç değeri ataması olarak value = 0 ve granularity(adet olarak düşünülebilir) = 60 yapılmıştır

NOT 3 : Ardından imleci oluşturmak için Rectangle komutu yazılmıştır. Bunun içindeki yükseklik id_root 'a göre ayarlanmıştır. Fakat id_root tüm dosyaların id'si olduğu için aslında ekran boyutuna göre ayarlanmışıtr da denilebilir.

NOT 4 : Rotation kısmında yazılan araştırmalar sonucunda bulunmuştur.

```
1  import QtQuick 2.0
2
3  Item {
4      id: id_root
5      property int value: 0
6      property int granularity: 60
7
8      Rectangle {
9          width: 2
10         height: id_root.height * 0.4
11         color: "darkslategray"
12         anchors {
13             horizontalCenter: id_root.horizontalCenter
14             bottom: id_root.verticalCenter
15         }
16         antialiasing: true
17     }
18
19     rotation: 360/granularity * (value % granularity)
20     antialiasing: true
21 }
22
```

GÖRSEL 2 -> MinuteNeedle.qml

NOT 5 : MinuteNeedle.qml dosyası SecondNeedle.qml dosyasına göre düzenlenmiştir.

NOT 6 : Satır 14 'deki bottom'ı verticalCenter'a eşitlenmezse dakika imleci saatin köşelerine dayanmaktadır.

NOT 7 : SecondNeedle.qml dosyasına da Satır 14 sonradan eklenmiştir.

```qml
import QtQuick 2.0

Item {
    id: id_root
    property int value: 0
    property int valueminute: 0
    property int granularity: 12

    Rectangle {
        width: 2
        height: id_root.height * 0.3
        color: "orangered"
        anchors {
            horizontalCenter: id_root.horizontalCenter
            bottom: id_root.verticalCenter
        }
        antialiasing: true
    }

    rotation: 360/granularity * (value%granularity) + 360/granularity * (valueminute / 60)
    antialiasing: true
}

```

Görsel 3 -> HourNeedle.qml

NOT 8 : HourNeedle dosyası da diğerlerinde uygulanan mantık ile yazılmıştır. Sadece rotation kısmında değişiklik olmuştur. Bunun sebebi ise saat imlecinin diğer imleçlere göre haraket hızının düşük olmasıdır.

```qml
import QtQuick 2.0

Item {
    id: id_root

    property color color: "darkturquoise"

    property int hours: currentDate.getHours()
    property int minutes: currentDate.getMinutes()
    property int seconds: currentDate.getSeconds()
    property var currentDate: new Date()

    Timer {
        id: timer
        repeat: true
        interval: 1000
        running: true

        onTriggered: id_root.currentDate = new Date()
    }

    Rectangle {
        id: id_plate

        anchors.centerIn: parent
        height: Math.min(id_root.width, id_root.height)
        width: height
        radius: width/2
        color: id_root.color
        border.color: "gold"
        border.width: 4

        Repeater {
            model: 12

            Item {
                id: hourContainer

                property int hour: index
                height: id_plate.height/2
                transformOrigin: Item.Bottom
                rotation: index * 30
                x: id_plate.width/2
                y: 0

```

NOT 9 : Satır 6 ‘daki özellik saatin arka rengini belirtir.

NOT 10 : Satır 8-9-10 güncel saat dakika ve saniye bilgisini alır.

NOT 11 : Satır 11 ‘de Date ve Clock özelliği aktif edilir.

NOT 12 : Satır 22 ‘deki Rectangle Saatin yuvarlaklığı ve boyutu, diğer özellikler belirlenir.

NOT 13 : Satır 33’dek Repeater ile 1’den 12 ‘ ye kadar saatler yerleştirmek için kullanılmıştır.

GÖRSEL 4 -> Clock.qml (Satır 1-45)

```
            Rectangle {
                height: id_plate.height*0.05
                width: height
                radius: width/2
                color: "ivory"
                anchors.horizontalCenter: parent.horizontalCenter
                anchors.top: parent.top
                anchors.topMargin: 4
            }

            Text {
                anchors {
                    horizontalCenter: parent.horizontalCenter
                }
                x: 0
                y: id_plate.height*0.06
                rotation: 360 - index * 30
                text: hourContainer.hour == 0 ? 12 : hourContainer.hour
                font.pixelSize: id_plate.height*0.1
                font.family: "Comic Sans MS"
                color: "darkslateblue"
            }
        }
    }
}
Rectangle {
    id: id_center

    anchors.centerIn: parent
    height: id_plate.height*0.05
    width: height
    radius: width/2
    color: "grey"
}
SecondNeedle {
    anchors {
        top: id_plate.top
        bottom: id_plate.bottom
        horizontalCenter: parent.horizontalCenter
    }
    value: id_root.seconds
}
MinuteNeedle {
    anchors {
        top: id_plate.top
        bottom: id_plate.bottom
        horizontalCenter: parent.horizontalCenter
    }
    value: id_root.minutes
}
HourNeedle {
    anchors {
        top: id_plate.top
        bottom: id_plate.bottom
        horizontalCenter: parent.horizontalCenter
    }
    value: id_root.hours
    valueminute: id_root.minutes
}
}
```

NOT 14 : Satır 46 ‘ daki Rectangle yapısı her saat üstündeki yuvarlak cisimleri oluşturmak için kullanılmıştır.

NOT 15 : Text kısmı ise 1’den 12’ ye kadar olan sayıları yerleştirmek için kullanılmıştır.

NOT 16 : Satır 71 ‘deki Rectangle yapısı saatin merkezinde imleçlerin uç noktasını merkez alan bir daire oluşturulmuştur.

NOT 17 : Son olarak ise oluşturulan SecondNeedle,MinuteNeedle ve Hour Needle yapıları çağırılıp yerleştirmeleri yapılmıştır.

GÖRSEL 5 -> Clock.qml (Satır 46-105)

```
import QtQuick 2.0

Clock {
    id: clock
    width:  400
    height: 400
}
```

GÖRSEL 6 -> main.qml