Лабораторная работа №1. Цветовые модели.

Выполнил Белоус Артем

1. Постановка задачи:

   Изучить цветовые модели: RGB, CMYK, HSV, преобразования между ними. Создать приложение/веб-приложение, позволяющее пользователю выбирать, а затем интерактивно менять цвет, показывая при этом его составляющие в трех моделях одновременно.

2. Входные данные:

   Цветовые модели RGB, CMYK, HSV.

2. Листинг программы с комментариями:

   Рассмотрим функции, используемые в ходе выполнения программы.

   Класс MainWindow содержит в себе все графические объекты программы и реализует ее логику. Конструктор этого класса имеет вид:

```
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Color converter")
        self.setMinimumSize(600,600)
        self.setMaximumSize(1500,1000)

        self.color_display = QLabel()

        # Основной вертикальный layout
        self.layout = QVBoxLayout()

        # Слайдеры для RGB
        self.RGB_layout = QVBoxLayout()
```

```
        self.red_slider = self.CreateSliderLayout(Colors.R
ED, 255)
        self.green_slider = self.CreateSliderLayout(Color
s.GREEN, 255)
        self.blue_slider = self.CreateSliderLayout(Colors.
BLUE, 255)

        self.RGB_layout.addLayout(self.red_slider)
        self.RGB_layout.addLayout(self.green_slider)
        self.RGB_layout.addLayout(self.blue_slider)

        #Слайдеры для CMYK
        self.CMYK_layout = QVBoxLayout()
        self.cyan_slider = self.CreateSliderLayout(Colors.
CYAN, 99)
        self.magenta_slider = self.CreateSliderLayout(Colo
rs.MAGENTA, 99)
        self.yellow_slider = self.CreateSliderLayout(Color
s.Yellow, 99)
        self.key_slider = self.CreateSliderLayout(Colors.K
EY, 99)

        self.CMYK_layout.addLayout(self.cyan_slider)
        self.CMYK_layout.addLayout(self.magenta_slider)
        self.CMYK_layout.addLayout(self.yellow_slider)
        self.CMYK_layout.addLayout(self.key_slider)


        # Слайдеры для HSV
        self.HSV_layout = QVBoxLayout()
        self.hue_slider = self.CreateSliderLayout(Colors.H
UE, 359)
        self.saturation_slider = self.CreateSliderLayout(C
olors.SATURATION, 99)
        self.value_slider = self.CreateSliderLayout(Color
s.VALUE, 99)
```

```
        self.HSV_layout.addLayout(self.hue_slider)
        self.HSV_layout.addLayout(self.saturation_slider)
        self.HSV_layout.addLayout(self.value_slider)

        self.CreateOutputFieldAndSelectButton()

        # Добавляем все элементы в layout
        self.layout.addLayout(self.output_layout)
        self.layout.addLayout(self.RGB_layout)
        self.layout.addLayout(self.CMYK_layout)
        self.layout.addLayout(self.HSV_layout)


        self.setLayout(self.layout)
```

Функция CreateSliderLayout создает слайдер и текстовое поле для изменения цвета с заданным именем, а также соединяет сигналы от текстового поля и слайдера с соответствующими слотами:

```
    def CreateSliderLayout(self, color_name, upper_bound):
        slider = QSlider(Qt.Horizontal)
        slider.setRange(0, upper_bound)
        slider.setValue(0)
        line_edit = QLineEdit("0")
        line_edit.setMaximumSize(50, 20)

        if color_name == Colors.RED or color_name == Color
s.GREEN or color_name == Colors.BLUE:
            slider.valueChanged.connect(self.EditedRGB)
            rx = QtCore.QRegExp("^(25[0-5]|2[0-4][0-9]|[0
1]?[0-9][0-9]?)?$")
            val = QRegExpValidator(rx)
            line_edit.setValidator(val)
```

```
            line_edit.textChanged.connect(self.EditedRGB)

        elif color_name == Colors.HUE or color_name == Col
ors.SATURATION or color_name == Colors.VALUE:
            slider.valueChanged.connect(self.EditedHSV)
            if color_name == Colors.HUE:
                rx = QtCore.QRegExp("^(3[0-5][0-9]|[1-2][0
-9]{2}|[0-9]{1,2})?$")
            else:
                rx = QtCore.QRegExp("^(99|[1-9][0-9]?|0)?
$")

            val = QRegExpValidator(rx)
            line_edit.setValidator(val)
            line_edit.textChanged.connect(self.EditedHSV)
        else:
            slider.valueChanged.connect(self.EditedCMYK)
            rx = QtCore.QRegExp("^(99|[1-9][0-9]?|0)?$")
            val = QRegExpValidator(rx)
            line_edit.setValidator(val)
            line_edit.textChanged.connect(self.EditedCMYK)

        label = QLabel(color_name.value + ":")

        h_layout = QHBoxLayout()
        h_layout.addWidget(label)
        h_layout.addWidget(line_edit)
        h_layout.addWidget(slider)

        return h_layout
```

Функция EditedRGB отвечает за изменение статуса всех слайдеров и текстовых полей, вызывается при изменении одного из слайдеров или

текстовых полей RGB палитры в результате действия пользователя. Аналогично реализованы функции EditedCMYK и EditedHSV:

```
    def EditedRGB(self):
        if isinstance(self.sender(), QLineEdit):
            redText = self.red_slider.itemAt(1).widget().t
ext();
            red = int(redText) if redText != '' else 0
            greenText = self.green_slider.itemAt(1).widget
().text()
            green = int(greenText) if greenText != '' else
0
            blueText = self.blue_slider.itemAt(1).widget
().text()
            blue = int(blueText) if blueText != '' else 0
            col = QColor(red, green, blue)
            self.red_slider.itemAt(2).widget().setValue(re
d)
            self.green_slider.itemAt(2).widget().setValue
(green)
            self.blue_slider.itemAt(2).widget().setValue(b
lue)
        else:
            red = self.red_slider.itemAt(2).widget().value
()
            green = self.green_slider.itemAt(2).widget().v
alue()
            blue = self.blue_slider.itemAt(2).widget().val
ue()
            col = QColor(red, green, blue)
            self.red_slider.itemAt(1).widget().setText(str
(red))
            self.green_slider.itemAt(1).widget().setText(s
tr(green))
            self.blue_slider.itemAt(1).widget().setText(st
r(blue))
```

```
        self.DisconnectSMYK()
        self.DisconnectHSV()

        self.UpdateCMYK(col)
        self.UpdateHSV(col)
        self.color_display.setStyleSheet(f"background-colo
r: rgb({red}, {green}, {blue});")

        self.ConnectSMYK()
        self.ConnectHSV()

    def EditedCMYK(self):
        if isinstance(self.sender(), QLineEdit):
            cyanText = self.cyan_slider.itemAt(1).widget
().text()
            cyan = int(cyanText) if cyanText != "" else 0
            magentaText = self.magenta_slider.itemAt(1).wi
dget().text()
            magenta = int(magentaText) if magentaText !=
"" else 0
            yellowText = self.yellow_slider.itemAt(1).widg
et().text()
            yellow = int(yellowText) if yellowText != "" e
lse 0
            keyText = self.key_slider.itemAt(1).widget().t
ext()
            key = int(keyText) if keyText != "" else 0
            col = QColor.fromCmykF(cyan/99, magenta/99, ye
llow/99, key/99)

            self.cyan_slider.itemAt(2).widget().setValue(c
yan)
            self.magenta_slider.itemAt(2).widget().setValu
e(magenta)
            self.yellow_slider.itemAt(2).widget().setValue
```

```
(yellow)
            self.key_slider.itemAt(2).widget().setValue(ke
y)
        else:
            cyan = self.cyan_slider.itemAt(2).widget().val
ue()
            magenta = self.magenta_slider.itemAt(2).widget
().value()
            yellow = self.yellow_slider.itemAt(2).widget
().value()
            key = self.key_slider.itemAt(2).widget().value
()
            col = QColor.fromCmykF(cyan/99, magenta/99, ye
llow/99, key/99)
            self.cyan_slider.itemAt(1).widget().setText(st
r(cyan))
            self.magenta_slider.itemAt(1).widget().setText
(str(magenta))
            self.yellow_slider.itemAt(1).widget().setText
(str(yellow))
            self.key_slider.itemAt(1).widget().setText(str
(key))

        self.DisconnectHSV()
        self.DisconnectRGB()

        self.UpdateHSV(col)
        self.UpdateRGB(col)
        col = col.toRgb()
        self.color_display.setStyleSheet(f"background-colo
r: rgb({col.red()}, {col.green()}, {col.blue()});")

        self.ConnectHSV()
        self.ConnectRGB()

    def EditedHSV(self):
```

```
        if isinstance(self.sender(), QLineEdit):
            hueText = self.hue_slider.itemAt(1).widget().t
ext()
            hue = int(hueText) if hueText != "" else 0
            saturationText = self.saturation_slider.itemAt
(1).widget().text()
            saturation = int(saturationText) if saturation
Text != "" else 0
            valueText = self.value_slider.itemAt(1).widget
().text()
            value = int(valueText) if valueText != "" else
0
            col = QColor.fromHsvF(hue / 359, saturation /
99, value / 99)

            self.hue_slider.itemAt(2).widget().setValue(hu
e)
            self.saturation_slider.itemAt(2).widget().setV
alue(saturation)
            self.value_slider.itemAt(2).widget().setValue
(value)
        else:
            hue = self.hue_slider.itemAt(2).widget().value
()
            saturation = self.saturation_slider.itemAt(2).
widget().value()
            value = self.value_slider.itemAt(2).widget().v
alue()
            col = QColor.fromHsvF(hue / 359, saturation /
99, value / 99)
            self.hue_slider.itemAt(1).widget().setText(str
(hue))
            self.saturation_slider.itemAt(1).widget().setT
ext(str(saturation))
            self.value_slider.itemAt(1).widget().setText(s
tr(value))
```

```
        self.DisconnectSMYK()
        self.DisconnectRGB()

        self.UpdateCMYK(col)
        self.UpdateRGB(col)
        col = col.toRgb()
        self.color_display.setStyleSheet(f"background-colo
r: rgb({col.red()}, {col.green()}, {col.blue()});")

        self.ConnectSMYK()
        self.ConnectRGB()
```

Функция UpdateRGB отвечает за изменение всех текстовых полей и слайдеров, связанных с палитрой RGB, согласно новому цвету, предаваемому в функцию в качестве параметра. Аналогично реализованы функции UpdateCMYK и UpdateHSV:

```
    def UpdateCMYK(self, col):
        color = col.toCmyk()
        self.cyan_slider.itemAt(2).widget().setValue(round
(color.cyanF() * 99))
        self.magenta_slider.itemAt(2).widget().setValue(ro
und(color.magentaF() * 99))
        self.yellow_slider.itemAt(2).widget().setValue(rou
nd(color.yellowF() * 99))
        self.key_slider.itemAt(2).widget().setValue(round
(color.blackF() * 99))

        self.cyan_slider.itemAt(1).widget().setText(str(ro
und(color.cyanF() * 99)))
        self.magenta_slider.itemAt(1).widget().setText(str
(round(color.magentaF() * 99)))
        self.yellow_slider.itemAt(1).widget().setText(str
```

```
(round(color.yellowF() * 99)))
        self.key_slider.itemAt(1).widget().setText(str(rou
nd(color.blackF() * 99)))

    def UpdateHSV(self, col):
        color = col.toHsv()
        self.hue_slider.itemAt(2).widget().setValue(round
(max(color.hueF() * 359, 0)))
        self.saturation_slider.itemAt(2).widget().setValue
(round(color.saturationF() * 99))
        self.value_slider.itemAt(2).widget().setValue(roun
d(color.valueF() * 99))

        self.hue_slider.itemAt(1).widget().setText(str(rou
nd(max(color.hueF() * 359, 0))))
        self.saturation_slider.itemAt(1).widget().setText
(str(round(color.saturationF() * 99)))
        self.value_slider.itemAt(1).widget().setText(str(r
ound(color.valueF() * 99)))

    def UpdateRGB(self, col):
        color = col.toRgb()
        self.red_slider.itemAt(2).widget().setValue(round
(color.redF() * 255))
        self.green_slider.itemAt(2).widget().setValue(roun
d(color.greenF() * 255))
        self.blue_slider.itemAt(2).widget().setValue(round
(color.blueF() * 255))

        self.red_slider.itemAt(1).widget().setText(str(rou
nd(color.redF() * 255)))
        self.green_slider.itemAt(1).widget().setText(str(r
ound(color.greenF() * 255)))
        self.blue_slider.itemAt(1).widget().setText(str(ro
und(color.blueF() * 255)))
```

Функция SelectColor отвечает за выбор цвета с помощью диалогового окна при нажатии на кнопку выбора цвета:

```
    def SelectColor(self):
        col = QColorDialog.getColor()

        self.DisconnectRGB()
        self.DisconnectSMYK()
        self.DisconnectHSV()

        self.color_display.setStyleSheet(f"background-colo
r: rgb({col.red()}, {col.green()}, {col.blue()});")
        self.UpdateRGB(col)
        self.UpdateCMYK(col)
        self.UpdateHSV(col)

        self.ConnectRGB()
        self.ConnectSMYK()
        self.ConnectHSV()
```

Функция DisconnectRGB предназначена для временного разъединения слотов и сигналов, связанных с палитрой RGB. Функция ConnectRGB, в свою очередь, предназначена для обратного присоединения слотов и сигналов. Эти функции вызываются с целью избегания зацикливания вызова функций EditedRGB. Аналогично реализованы функции DisconnectCMYK, ConnectCMYK, DisconnectHSV, ConnectHSV:

```
    def DisconnectSMYK(self):
        self.cyan_slider.itemAt(2).widget().valueChanged.d
isconnect(self.EditedCMYK)
        self.cyan_slider.itemAt(1).widget().textChanged.di
sconnect(self.EditedCMYK)
        self.magenta_slider.itemAt(2).widget().valueChange
```

```
d.disconnect(self.EditedCMYK)
        self.magenta_slider.itemAt(1).widget().textChange
d.disconnect(self.EditedCMYK)
        self.yellow_slider.itemAt(2).widget().valueChange
d.disconnect(self.EditedCMYK)
        self.yellow_slider.itemAt(1).widget().textChanged.
disconnect(self.EditedCMYK)
        self.key_slider.itemAt(2).widget().valueChanged.di
sconnect(self.EditedCMYK)
        self.key_slider.itemAt(1).widget().textChanged.dis
connect(self.EditedCMYK)

    def ConnectSMYK(self):
        self.cyan_slider.itemAt(2).widget().valueChanged.c
onnect(self.EditedCMYK)
        self.cyan_slider.itemAt(1).widget().textChanged.co
nnect(self.EditedCMYK)
        self.magenta_slider.itemAt(2).widget().valueChange
d.connect(self.EditedCMYK)
        self.magenta_slider.itemAt(1).widget().textChange
d.connect(self.EditedCMYK)
        self.yellow_slider.itemAt(2).widget().valueChange
d.connect(self.EditedCMYK)
        self.yellow_slider.itemAt(1).widget().textChanged.
connect(self.EditedCMYK)
        self.key_slider.itemAt(2).widget().valueChanged.co
nnect(self.EditedCMYK)
        self.key_slider.itemAt(1).widget().textChanged.con
nect(self.EditedCMYK)

    def DisconnectHSV(self):
        self.hue_slider.itemAt(2).widget().valueChanged.di
sconnect(self.EditedHSV)
        self.hue_slider.itemAt(1).widget().textChanged.dis
connect(self.EditedHSV)
        self.saturation_slider.itemAt(2).widget().valueCha
```

```
nged.disconnect(self.EditedHSV)
        self.saturation_slider.itemAt(1).widget().textChan
ged.disconnect(self.EditedHSV)
        self.value_slider.itemAt(2).widget().valueChanged.
disconnect(self.EditedHSV)
        self.value_slider.itemAt(1).widget().textChanged.d
isconnect(self.EditedHSV)

    def ConnectHSV(self):
        self.hue_slider.itemAt(2).widget().valueChanged.co
nnect(self.EditedHSV)
        self.hue_slider.itemAt(1).widget().textChanged.con
nect(self.EditedHSV)
        self.saturation_slider.itemAt(2).widget().valueCha
nged.connect(self.EditedHSV)
        self.saturation_slider.itemAt(1).widget().textChan
ged.connect(self.EditedHSV)
        self.value_slider.itemAt(2).widget().valueChanged.
connect(self.EditedHSV)
        self.value_slider.itemAt(1).widget().textChanged.c
onnect(self.EditedHSV)

    def DisconnectRGB(self):
        self.red_slider.itemAt(2).widget().valueChanged.di
sconnect(self.EditedRGB)
        self.red_slider.itemAt(1).widget().textChanged.dis
connect(self.EditedRGB)
        self.green_slider.itemAt(2).widget().valueChanged.
disconnect(self.EditedRGB)
        self.green_slider.itemAt(1).widget().textChanged.d
isconnect(self.EditedRGB)
        self.blue_slider.itemAt(2).widget().valueChanged.d
isconnect(self.EditedRGB)
        self.blue_slider.itemAt(1).widget().textChanged.di
sconnect(self.EditedRGB)
```

```
    def ConnectRGB(self):
        self.red_slider.itemAt(2).widget().valueChanged.co
nnect(self.EditedRGB)
        self.red_slider.itemAt(1).widget().textChanged.con
nect(self.EditedRGB)
        self.green_slider.itemAt(2).widget().valueChanged.
connect(self.EditedRGB)
        self.green_slider.itemAt(1).widget().textChanged.c
onnect(self.EditedRGB)
        self.blue_slider.itemAt(2).widget().valueChanged.c
onnect(self.EditedRGB)
        self.blue_slider.itemAt(1).widget().textChanged.co
nnect(self.EditedRGB)
```

Класс Colors представляет собой перечислений всех использующихся цветов для создания моделей:

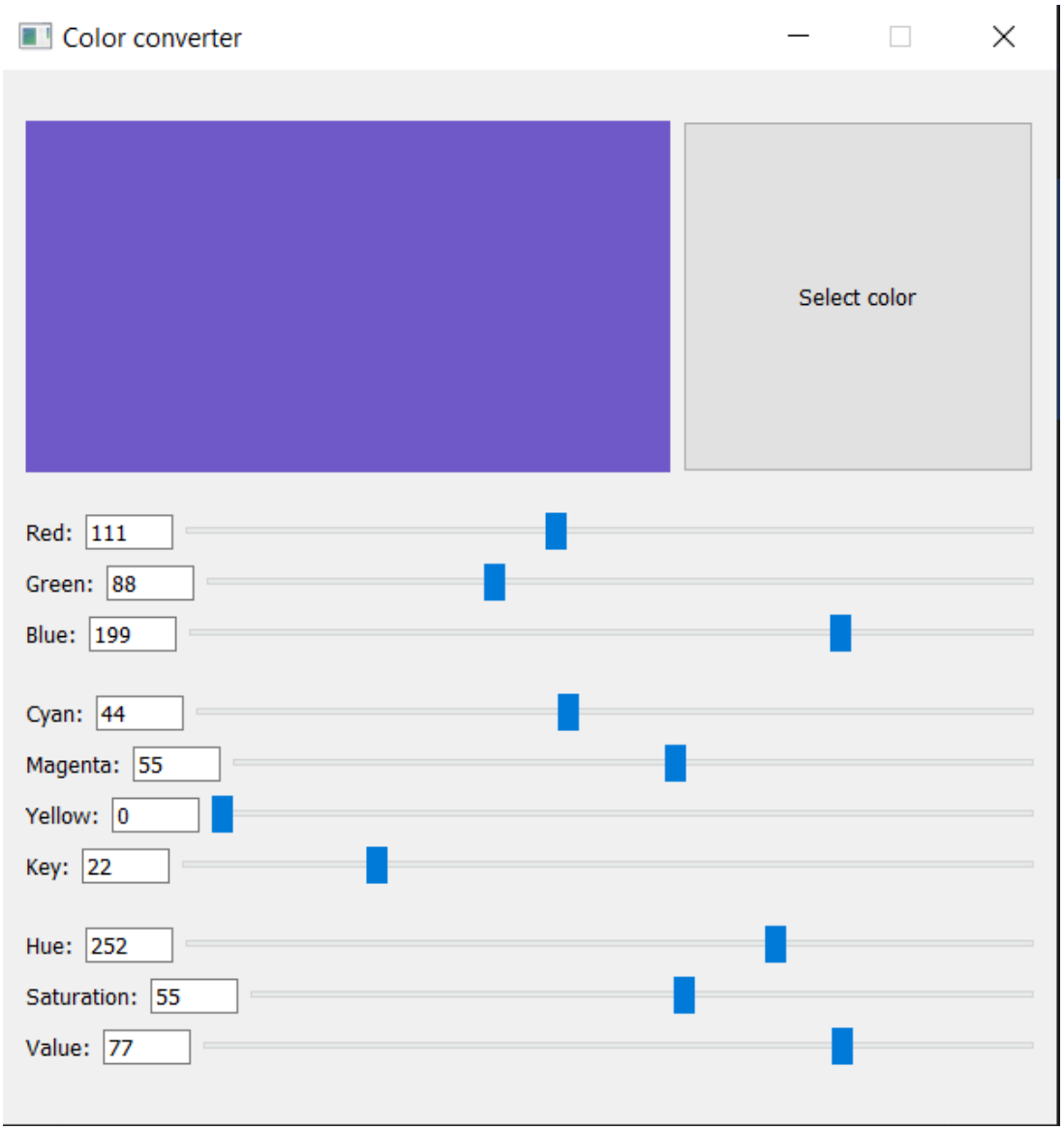
```
class Colors(Enum):
    RED = 'Red'
    GREEN = 'Green'
    BLUE = 'Blue'
    CYAN = 'Cyan'
    MAGENTA = 'Magenta'
    Yellow = 'Yellow'
    KEY = 'Key'
    HUE = 'Hue'
    SATURATION = 'Saturation'
    VALUE = 'Value'
```

Наконец, функция main является точкой входа программы и создает оконное приложение

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

3. Выходные данные.

   Выходными данными является программа для преобразования цвета с помощью различных цветовых моделей:



4. Вывод

   В ходе выполнения лабораторной работы я изучил цветовые модели RGB, CMYK и HSV путем написания программы с удобным графическим интерфейсом для преобразования цветов в данных моделях.