**Lab Exercise 4 – Basic Image Manipulation (Scaling, Resizing) using PySide6**



**Objective**

In this lab, you will learn how to manipulate images using **PySide6** by implementing **scaling and resizing** features. You will create a GUI application that allows users to **load an image, scale it up or down, and reset it to its original size**.

**Prerequisites**

* Python installed (Python 3.7+ recommended)
* PySide6 installed (pip install PySide6)

**Task 1: Setting Up the GUI**

Create a basic PySide6 application with:

* A QLabel to display the image
* A QPushButton to load an image
* A QSlider to adjust the image size

**Steps**

1. **Create a QWidget window**
2. **Use a QVBoxLayout to arrange widgets**



1. **Add a QLabel for displaying the image**



1. **Add a QPushButton to open image files**



1. **Add a QSlider to scale the image size dynamically**



**Task 2: Implementing Image Scaling**

* Use QPixmap.scaled() to scale the image while maintaining the aspect ratio.
* Link the QSlider to dynamically resize the image.

**Task 3: Implementing Reset Functionality**

* Add a reset button to restore the image to its original size.



**Complete Code**

import sys

from PySide6.QtWidgets import QApplication, QWidget, QLabel, QPushButton, QVBoxLayout, QFileDialog, QSlider, QHBoxLayout

from PySide6.QtGui import QPixmap

from PySide6.QtCore import Qt

class ImageEditor(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Basic Image Manipulation - PySide6")

self.setGeometry(200, 200, 700, 600)

# Layout

self.layout = QVBoxLayout()

# QLabel to display image

self.label = QLabel(self)



self.label.setText("No image loaded")

self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)

self.label.setStyleSheet("QLabel { font-size: 16px; color: gray; }")

# QPushButton to load image



self.load\_button = QPushButton("Load Image")

self.load\_button.clicked.connect(self.load\_image)

# QSlider for resizing the image



self.slider = QSlider(Qt.Orientation.Horizontal)

self.slider.setMinimum(10) # Minimum scale 10%

self.slider.setMaximum(200) # Maximum scale 200%

self.slider.setValue(100) # Default 100% (original size)

self.slider.setTickInterval(10)

self.slider.setTickPosition(QSlider.TickPosition.TicksBelow)

self.slider.valueChanged.connect(self.resize\_image)

# Reset Button



self.reset\_button = QPushButton("Reset Image")

self.reset\_button.clicked.connect(self.reset\_image)

# Horizontal layout for slider

slider\_layout = QHBoxLayout()

slider\_layout.addWidget(QLabel("10%"))

slider\_layout.addWidget(self.slider)

slider\_layout.addWidget(QLabel("200%"))

# Add widgets to layout

self.layout.addWidget(self.label)

self.layout.addWidget(self.load\_button)



self.layout.addLayout(slider\_layout)

self.layout.addWidget(self.reset\_button)

self.setLayout(self.layout)

self.pixmap = None # Store the original pixmap

def load\_image(self):

""" Opens a file dialog to select an image and displays it in QLabel """

image\_path, \_ = QFileDialog.getOpenFileName(self, "Open Image File", "", "Images (\*.png \*.jpg \*.jpeg \*.bmp)")

if image\_path:

self.pixmap = QPixmap(image\_path)

self.label.setPixmap(self.pixmap)

self.label.setScaledContents(True) # Scale to fit label size

self.label.setFixedSize(500, 400) # Set fixed label size

def resize\_image(self):

""" Resizes the image based on the slider value """

if self.pixmap:

scale\_factor = self.slider.value() / 100 # Convert to percentage

new\_width = int(self.pixmap.width() \* scale\_factor)

new\_height = int(self.pixmap.height() \* scale\_factor)

resized\_pixmap = self.pixmap.scaled(new\_width, new\_height, Qt.AspectRatioMode.KeepAspectRatio)

self.label.setPixmap(resized\_pixmap)

def reset\_image(self):

""" Resets the image to its original size """

if self.pixmap:

self.slider.setValue(100) # Reset slider to default

self.label.setPixmap(self.pixmap) # Restore original image

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

window = ImageEditor()

window.show()

sys.exit(app.exec())

**How It Works**

1. **Load an Image** using the "Load Image" button.
2. **Resize Image** by moving the slider (10% to 200%).
3. **Reset Image** to its original size using the "Reset Image" button.

import sys  
from PySide6.QtWidgets import (  
 QApplication, QWidget, QLabel, QPushButton, QVBoxLayout, QFileDialog,  
 QSlider, QHBoxLayout  
)  
from PySide6.QtGui import QPixmap, QImage, QTransform  
from PySide6.QtCore import Qt  
  
class ImageEditor(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 self.setWindowTitle("Advanced Image Editor - PySide6")  
 self.setGeometry(200, 200, 700, 600)  
  
 *# Layout* self.layout = QVBoxLayout()  
  
 *# QLabel to display image* self.label = QLabel(self)  
 self.label.setText("No image loaded")  
 self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)  
 self.label.setStyleSheet("QLabel { font-size: 16px; color: gray; }")  
  
 *# QPushButton to load image* self.load\_button = QPushButton("Load Image")  
 self.load\_button.setFixedSize(150, 80)  
 self.load\_button.clicked.connect(self.load\_image)  
  
 *# QSlider for resizing the image* self.slider = QSlider(Qt.Orientation.Horizontal)  
 self.slider.setMinimum(10) *# Minimum scale 10%* self.slider.setMaximum(200) *# Maximum scale 200%* self.slider.setValue(100) *# Default 100% (original size)* self.slider.setTickInterval(10)  
 self.slider.setTickPosition(QSlider.TickPosition.TicksBelow)  
 self.slider.valueChanged.connect(self.resize\_image)  
  
 *# Rotate Button* self.rotate\_button = QPushButton("Rotate 90°")  
 self.rotate\_button.setFixedSize(100, 50)  
 self.rotate\_button.clicked.connect(self.rotate\_image)  
  
 *# Grayscale Button* self.grayscale\_button = QPushButton("Grayscale")  
 self.grayscale\_button.setFixedSize(100, 50)  
 self.grayscale\_button.clicked.connect(self.convert\_to\_grayscale)  
  
 *# Save Button* self.save\_button = QPushButton("Save Image")  
 self.save\_button.setFixedSize(100, 50)  
 self.save\_button.clicked.connect(self.save\_image)  
  
 *# Reset Button* self.reset\_button = QPushButton("Reset Image")  
 self.reset\_button.setFixedSize(100, 50)  
 self.reset\_button.clicked.connect(self.reset\_image)  
  
 *# Layouts for controls* slider\_layout = QHBoxLayout()  
 slider\_layout.addWidget(QLabel("10%"))  
 slider\_layout.addWidget(self.slider)  
 slider\_layout.addWidget(QLabel("200%"))  
  
 button\_layout = QHBoxLayout()  
 button\_layout.addWidget(self.rotate\_button)  
 button\_layout.addWidget(self.grayscale\_button)  
 button\_layout.addWidget(self.save\_button)  
  
 *# Add widgets to layout* self.layout.addWidget(self.label,alignment=Qt.AlignmentFlag.AlignCenter)  
 self.layout.addWidget(self.load\_button,alignment=Qt.AlignmentFlag.AlignCenter)  
 self.layout.addLayout(slider\_layout)  
 self.layout.addLayout(button\_layout)  
 self.layout.addWidget(self.reset\_button,alignment=Qt.AlignmentFlag.AlignCenter)  
  
 self.setLayout(self.layout)  
  
 self.pixmap = None *# Store the original pixmap* self.original\_pixmap = None *# Store a backup of the original image* self.rotation\_angle = 0 *# Track rotation angle* def load\_image(self):  
 *""" Opens a file dialog to select an image and displays it in QLabel """* image\_path, \_ = QFileDialog.getOpenFileName(self, "Open Image File", "", "Images (\*.png \*.jpg \*.jpeg \*.bmp)")  
 if image\_path:  
 self.pixmap = QPixmap(image\_path)  
 self.original\_pixmap = QPixmap(image\_path) *# Store original* self.rotation\_angle = 0 *# Reset rotation* self.label.setPixmap(self.pixmap)  
 self.label.setScaledContents(False)  
 self.label.setFixedSize(500, 400)  
  
 def resize\_image(self):  
 *""" Resizes the image based on the slider value """* if self.pixmap:  
 scale\_factor = self.slider.value() / 100  
 new\_width = int(self.pixmap.width() \* scale\_factor)  
 new\_height = int(self.pixmap.height() \* scale\_factor)  
 resized\_pixmap = self.pixmap.scaled(new\_width, new\_height)  
 self.label.setPixmap(resized\_pixmap)  
  
 def rotate\_image(self):  
 *""" Rotates the image by 90 degrees """* if self.pixmap:  
 self.rotation\_angle += 90 *# Increase rotation angle* transform = QTransform().rotate(self.rotation\_angle)  
 rotated\_pixmap = self.pixmap.transformed(transform, Qt.TransformationMode.SmoothTransformation)  
 self.label.setPixmap(rotated\_pixmap)  
 self.pixmap = rotated\_pixmap *# Update current pixmap* def convert\_to\_grayscale(self):  
 *""" Converts the image to grayscale """* if self.pixmap:  
 image = self.pixmap.toImage()  
 grayscale\_image = image.convertToFormat(QImage.Format\_Grayscale8)  
 self.pixmap = QPixmap.fromImage(grayscale\_image)  
 self.label.setPixmap(self.pixmap)  
  
 def save\_image(self):  
 *""" Saves the modified image to a file """* if self.pixmap:  
 save\_path, \_ = QFileDialog.getSaveFileName(self, "Save Image", "", "Images (\*.png \*.jpg \*.jpeg \*.bmp)")  
 if save\_path:  
 self.pixmap.save(save\_path)  
  
 def reset\_image(self):  
 *""" Resets the image to its original size and state """* if self.original\_pixmap:  
 self.pixmap = self.original\_pixmap  
 self.label.setPixmap(self.pixmap)  
 self.slider.setValue(100)  
 self.rotation\_angle = 0 *# Reset rotation*if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = ImageEditor()  
 window.show()  
 sys.exit(app.exec())