**Basic image manipulation (scaling, resizing) using PySide6**

**Basic image manipulation (scaling, resizing)** using **PySide6** in Python. We will create a GUI where users can **load an image**, **resize it**, and **scale it** dynamically.

**Project Structure**

**A close-up of a computer screen

AI-generated content may be incorrect.**

**Complete Code (main.py)**

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.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)  
  
 *# 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,alignment=Qt.AlignmentFlag.AlignCenter)  
 self.layout.addLayout(slider\_layout)  
  
 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(False) *# Scale to fit label size* self.label.setFixedSize(500, 400) *# Set fixed label size* else:  
 self.label.setText("No image selected")  
  
 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)  
 self.label.setPixmap(resized\_pixmap)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = ImageEditor()  
 window.show()  
 sys.exit(app.exec())

**Code Explanation**

**1. Initializing the Window (ImageEditor class)**

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

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

* Sets the window title and size.

**2. QLabel (For Displaying 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; }")

* Displays text when no image is loaded.
* Uses **CSS styling** for better appearance.

**3. QPushButton (For Loading an Image)**

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

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

* Clicking the button **calls load\_image()**.

**4. QFileDialog (Select Image File)**

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

* Opens a **file dialog** for users to select an image.
* Filters file types to **PNG, JPG, JPEG, BMP**.

**5. QLabel Displays the Selected Image**

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

* Loads the selected image using QPixmap.
* Displays it inside QLabel.
* Sets a fixed label size **(500x400 pixels)**.

**6. 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)

* **Minimum size: 10%**, **Maximum size: 200%**.
* **Moves in steps of 10%**.
* **Triggers resize\_image()** when changed.

**7. Resizing the Image**

def resize\_image(self):

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)

* **Calculates new dimensions** based on slider value.
* Uses QPixmap.scaled() to resize the image while **keeping aspect ratio**.

**How to Run the Application**

1. **Install PySide6**:

pip install PySide6

1. **Run the script**:

python main.py

**Expected Behavior**

* Initially, the window displays **"No image loaded"**.
* Clicking **"Load Image"** opens a file dialog.
* Selecting an image **displays it in the window**.
* The **slider resizes the image dynamically** from **10% to 200%** of its original size.

**Possible Enhancements**

1. **Rotate Image**: Add buttons to rotate images.
2. **Save Edited Image**: Save the resized image to a file.
3. **Apply Filters**: Add grayscale, blur, or contrast filters.

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)  
  
 *# Horizontal layout for slider* slider\_layout = QHBoxLayout()  
 slider\_layout.addWidget(QLabel("10%"))  
 slider\_layout.addWidget(self.slider)  
 slider\_layout.addWidget(QLabel("200%"))  
  
 *# Horizontal layout for buttons* button\_layout = QHBoxLayout()  
 button\_layout.addWidget(self.rotate\_button)  
 button\_layout.addWidget(self.grayscale\_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.setLayout(self.layout)  
  
 self.pixmap = None *# Store the original pixmap* 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.rotation\_angle = 0 *# Reset rotation* self.label.setPixmap(self.pixmap)  
 self.label.setScaledContents(False) *# Scale to fit label size* self.label.setFixedSize(500, 400) *# Set fixed label size* else:  
 self.label.setText("No image selected")  
  
 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)  
 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) *# Rotate* rotated\_pixmap = self.pixmap.transformed(transform, Qt.TransformationMode.SmoothTransformation)  
 self.label.setPixmap(rotated\_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)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = ImageEditor()  
 window.show()  
 sys.exit(app.exec())