```python
#widgets.py

# coding: utf-8

from utils import QtCore, QtWidgets, QtGui, AbstractFunction
import config
from ui_design.ui_design import Ui_Form
import os
import typing


class WidgetMain(QtWidgets.QWidget, Ui_Form, AbstractFunction):
    def __init__(self, parent=None):
        super().__init__(parent=parent)
        self.setupUi(self)

        layout = QtWidgets.QVBoxLayout(self)

        layout.addWidget(self.group_viewer)
        layout.insertWidget(1, self.group_choose_image)

        layout.insertWidget(1, self.group_decide_position)
        self.group_decide_position.hide()

        layout.insertWidget(1, self.group_input_text)
        self.group_input_text.hide()

        layout.setStretchFactor(self.group_viewer, 1)

        self.setLayout(layout)

        self.current_image = ""
        self.current_pix = QtGui.QPixmap()
        self.text_pos = 0, 0, 0, 0

        self.graphics_viewer.sig_position.connect(self.handle_user_pick_position)

        self.btn_choose_image.clicked.connect(self.handle_choose_a_image)
        self.btn_apply_image.clicked.connect(self.handle_switch_to_decide_position)
        self.btn_goback_image.clicked.connect(self.handle_switch_to_choose_image)
        self.btn_apply_position.clicked.connect(self.handle_switch_to_input_text)
        self.btn_goback_position.clicked.connect(self.handle_switch_to_decide_position)

        self.edit_x.setReadOnly(True)
        self.edit_y.setReadOnly(True)
        self.edit_w.setReadOnly(True)
        self.edit_h.setReadOnly(True)

        self.setWindowTitle(config.app_name_cn)

        self.btn_apply_text.clicked.connect(self.handle_display_text)
        self.btn_save_image.clicked.connect(self.handle_save_image)
        self.btn_check.setCheckState(QtCore.Qt.Checked)
        self.btn_check.clicked.connect(self.handle_toggle_rect)

    def handle_choose_a_image(self):
        ld = self.get_last_directory()
        fp, _etx = QtWidgets.QFileDialog.getOpenFileName(
            parent=self, caption="Choose An Image", directory=ld, filter="Image Files(*.jpg *.jpeg *.png)"
        )
        if not fp:
            return
        fp = os.path.abspath(fp)
        self.current_image = fp
        self.current_pix = QtGui.QPixmap(fp)
        # print(self.current_pix.size())
        self.save_last_directory(dir_path=os.path.dirname(fp))
        self.graphics_viewer.display_pix(pix=self.current_pix)

    def handle_switch_to_decide_position(self):
        if not self.current_image:
            return self.show_warning_message(
                message="Please Choose An Image", title="Please Choose Image", parent=self, only_yes=True
            )
        self.__hide_all()
        self.group_decide_position.show()
        self.graphics_viewer.setAllowDrawRect(True)
        self.graphics_viewer.clear_text()
        self.graphics_viewer.display_rect()

    def handle_switch_to_choose_image(self):
        self.__hide_all()
        self.group_choose_image.show()
        self.graphics_viewer.setAllowDrawRect(False)
        self.graphics_viewer.hide_rect()

    def __hide_all(self):
        for i in [self.group_choose_image, self.group_decide_position, self.group_input_text]:
            i.hide()

    def handle_switch_to_input_text(self):
        if self.text_pos == (0, 0, 0, 0):
            return self.show_warning_message(
                message="Please choose an area!", parent=self, title="Warning", only_yes=True
            )

        self.__hide_all()
        self.group_input_text.show()
        self.graphics_viewer.setAllowDrawRect(False)
        self.handle_toggle_rect()

    def handle_user_pick_position(self, text_pos):
        # save and update the chosen area
        x, y, w, h = text_pos
        self.edit_x.setText(f"{x:.0f}")
        self.edit_y.setText(f"{y:.0f}")
        self.edit_w.setText(f"{w:.0f}")
        self.edit_h.setText(f"{h:.0f}")

        self.text_pos = text_pos

    def handle_display_text(self):
        txt = self.edit_txt.toPlainText()
        # txt.setWordWrapMode(QtGui.QTextOption.WrapMode)
        if not txt:
            return self.show_warning_message(
                message="Please input some text", parent=self, title="Warning", only_yes=True
            )

        font = self.font_chooser.currentFont()
        size = self.size_chooser.value()
        font.setPixelSize(size)
        metrics = QtGui.QFontMetrics(font)
        x, y, w, h = self.text_pos

        r = metrics.boundingRect(int(x), int(y), int(w), int(h), QtCore.Qt.AlignLeft, txt)

        if r.width() * r.height() > w * h:
            resp = self.show_warning_message(
                message="The text is too long for the area being chosen. " "Try break long lines?",
                title="Warning",
                parent=self,
                only_yes=False,
            )
            if resp is False:
                return self.graphics_viewer.display_text(txt=txt, font=font)

        def break_line(line: str, width: int) -> typing.List[str]:
            length = len(line)
            ls = [0]
            while True:
                last = ls[-1]
                t = line[last:]
                rect = metrics.boundingRect(0, 0, 0, 0, QtCore.Qt.AlignLeft, t)
                s = rect.width()
                if s <= width:
                    break
                for i in range(1, length):
                    t = line[last : last + i]
                    rect = metrics.boundingRect(0, 0, 0, 0, QtCore.Qt.AlignLeft, t)
                    s = rect.width()

                    if s == width:
                        ls.append(last + i)
                        break
                    elif s > width:
                        ls.append(last + i - 1)
                        break

            ls.append(length)
            results = []
            for i in range(len(ls) - 1):
                results.append(line[ls[i] : ls[i + 1]])
            return results

        lines = txt.split("\n")  # break line
        pieces = []

        for line in lines:
            pieces.extend(break_line(line=line, width=w))

        r = metrics.boundingRect(int(x), int(y), int(w), int(h), QtCore.Qt.AlignLeft, "\n".join(pieces))

        if r.width() * r.height() > w * h:
            resp = self.show_warning_message(
                message="The area your chosen can not hold that much txt your typed
```

```python
even after line break. "
                "Hit Yes, the txt on the image will be cleared",
                title="Warning",
                parent=self,
                only_yes=False,
            )
            if resp:
                return self.graphics_viewer.clear_text()
        self.graphics_viewer.display_text("\n".join(pieces), font=font)

    def handle_save_image(self):
        """save modified image"""
        ld = self.get_last_directory()
        fp, _ext = QtWidgets.QFileDialog.getSaveFileName(
            parent=self, caption="Save image", directory=ld, filter="Png
Image(*.png)"
        )
        if not fp:
            return
        fp = os.path.abspath(fp)
        self.save_last_directory(dir_path=os.path.dirname(fp))
        pix = self.graphics_viewer.save_image()
        pix.save(fp, "PNG")

    def handle_toggle_rect(self):
        if self.btn_check.checkState() == QtCore.Qt.Checked:
            self.graphics_viewer.display_rect()
        else:
            self.graphics_viewer.hide_rect()


# if __name__ == '__main__':
#     import sys
#     app = QtWidgets.QApplication(sys.argv)
#     m = WidgetMain()
#     m.show()
#     sys.exit(app.exec_())
```