

```

1 #widgets.py
2
3 # coding: utf-8
4
5 from utils importQtCore, QtWidgets, QtGui, AbstractFunction
6 import config
7 from ui_design.ui_design import Ui_Form
8 import os
9 import typing
10
11
12 class WidgetMain(QtWidgets.QWidget, Ui_Form, AbstractFunction):
13     def __init__(self, parent=None):
14         super().__init__(parent=parent)
15         self.setupUi(self)
16
17         layout = QtWidgets.QVBoxLayout(self)
18
19         layout.addWidget(self.group_viewer)
20         layout.insertWidget(1, self.group_choose_image)
21
22         layout.insertWidget(1, self.group_decide_position)
23         self.group_decide_position.hide()
24
25         layout.insertWidget(1, self.group_input_text)
26         self.group_input_text.hide()
27
28         layout.setStretchFactor(self.group_viewer, 1)
29
30         self.setLayout(layout)
31
32         self.current_image = ""
33         self.current_pixmap = QtGui.QPixmap()
34         self.text_pos = 0, 0, 0, 0
35
36         self.graphics_viewer.sig_position.connect(self.handle_user_pick_position)
37
38         self.btn_choose_image.clicked.connect(self.handle_choose_a_image)
39         self.btn_apply_image.clicked.connect(self.handle_switch_to_decide_position)
40         self.btn_goback_image.clicked.connect(self.handle_switch_to_choose_image)
41         self.btn_apply_position.clicked.connect(self.handle_switch_to_input_text)
42
43         self.btn_goback_position.clicked.connect(self.handle_switch_to_decide_position)
44
45         self.edit_x.setReadOnly(True)
46         self.edit_y.setReadOnly(True)
47         self.edit_w.setReadOnly(True)
48         self.edit_h.setReadOnly(True)
49
50         self.setWindowTitle(config.app_name_cn)
51
52         self.btn_apply_text.clicked.connect(self.handle_display_text)
53         self.btn_save_image.clicked.connect(self.handle_save_image)
54         self.btn_check.setCheckState(QtCore.Qt.Checked)
55         self.btn_check.clicked.connect(self.handle_toggle_rect)
56
57     def handle_choose_a_image(self):
58         ld = self.get_last_directory()
59         fp, _etx = QtWidgets.QFileDialog.getOpenFileName(
60             parent=self, caption="Choose An Image", directory=ld, filter="Image
61             Files (*.jpg *.jpeg *.png)")

```

```

60         )
61         if not fp:
62             return
63         fp = os.path.abspath(fp)
64         self.current_image = fp
65         self.current_pix = QtGui.QPixmap(fp)
66         # print(self.current_pix.size())
67         self.save_last_directory(dir_path=os.path.dirname(fp))
68         self.graphics_viewer.display_pix(pix=self.current_pix)
69
70     def handle_switch_to_decide_position(self):
71         if not self.current_image:
72             return self.show_warning_message(
73                 message="Please Choose An Image", title="Please Choose Image",
74                 parent=self, only_yes=True
75             )
76         self.__hide_all()
77         self.group_decide_position.show()
78         self.graphics_viewer.setAllowDrawRect(True)
79         self.graphics_viewer.clear_text()
80         self.graphics_viewer.display_rect()
81
82     def handle_switch_to_choose_image(self):
83         self.__hide_all()
84         self.group_choose_image.show()
85         self.graphics_viewer.setAllowDrawRect(False)
86         self.graphics_viewer.hide_rect()
87
88     def __hide_all(self):
89         for i in [self.group_choose_image, self.group_decide_position,
90                 self.group_input_text]:
91             i.hide()
92
93     def handle_switch_to_input_text(self):
94         if self.text_pos == (0, 0, 0, 0):
95             return self.show_warning_message(
96                 message="Please choose an area!", parent=self, title="Warning",
97                 only_yes=True
98             )
99         self.__hide_all()
100         self.group_input_text.show()
101         self.graphics_viewer.setAllowDrawRect(False)
102         self.handle_toggle_rect()
103
104     def handle_user_pick_position(self, text_pos):
105         # save and update the chosen area
106         x, y, w, h = text_pos
107         self.edit_x.setText(f"{x:.0f}")
108         self.edit_y.setText(f"{y:.0f}")
109         self.edit_w.setText(f"{w:.0f}")
110         self.edit_h.setText(f"{h:.0f}")
111
112         self.text_pos = text_pos
113
114     def handle_display_text(self):
115         txt = self.edit_txt.toPlainText()
116         # txt.setWordWrapMode(QtGui.QTextOption.WrapMode)
117         if not txt:
118             return self.show_warning_message(
119                 message="Please input some text", parent=self, title="Warning",
120                 only_yes=True

```

```

118         )
119
120         font = self.font_chooser.currentFont()
121         size = self.size_chooser.value()
122         font.setPixelSize(size)
123         metrics = QtGui.QFontMetrics(font)
124         x, y, w, h = self.text_pos
125
126         r = metrics.boundingRect(int(x), int(y), int(w), int(h),
QtCore.Qt.AlignLeft, txt)
127
128         if r.width() * r.height() > w * h:
129             resp = self.show_warning_message(
130                 message="The text is too long for the area being chosen. " "Try
break long lines?",
131                 title="Warning",
132                 parent=self,
133                 only_yes=False,
134             )
135             if resp is False:
136                 return self.graphics_viewer.display_text(txt=txt, font=font)
137
138         def break_line(line: str, width: int) -> typing.List[str]:
139             length = len(line)
140             ls = [0]
141             while True:
142                 last = ls[-1]
143                 t = line[last:]
144                 rect = metrics.boundingRect(0, 0, 0, 0, QtCore.Qt.AlignLeft, t)
145                 s = rect.width()
146                 if s <= width:
147                     break
148                 for i in range(1, length):
149                     t = line[last : last + i]
150                     rect = metrics.boundingRect(0, 0, 0, 0, QtCore.Qt.AlignLeft, t)
151                     s = rect.width()
152
153                     if s == width:
154                         ls.append(last + i)
155                         break
156                     elif s > width:
157                         ls.append(last + i - 1)
158                         break
159
160             ls.append(length)
161             results = []
162             for i in range(len(ls) - 1):
163                 results.append(line[ls[i] : ls[i + 1]])
164             return results
165
166         lines = txt.split("\n") # break line
167         pieces = []
168
169         for line in lines:
170             pieces.extend(break_line(line=line, width=w))
171
172         r = metrics.boundingRect(int(x), int(y), int(w), int(h),
QtCore.Qt.AlignLeft, "\n".join(pieces))
173
174         if r.width() * r.height() > w * h:
175             resp = self.show_warning_message(
176                 message="The area your chosen can not hold that much txt your typed

```

```

even after line break. "
177         "Hit Yes, the txt on the image will be cleared",
178         title="Warning",
179         parent=self,
180         only_yes=False,
181     )
182     if resp:
183         return self.graphics_viewer.clear_text()
184     self.graphics_viewer.display_text("\n".join(pieces), font=font)
185
186     def handle_save_image(self):
187         """save modified image"""
188         ld = self.get_last_directory()
189         fp, _ext = QtWidgets.QFileDialog.getSaveFileName(
190             parent=self, caption="Save image", directory=ld, filter="Png
Image(*.png)"
191         )
192         if not fp:
193             return
194         fp = os.path.abspath(fp)
195         self.save_last_directory(dir_path=os.path.dirname(fp))
196         pix = self.graphics_viewer.save_image()
197         pix.save(fp, "PNG")
198
199     def handle_toggle_rect(self):
200         if self.btn_check.checkState() == QtCore.Qt.Checked:
201             self.graphics_viewer.display_rect()
202         else:
203             self.graphics_viewer.hide_rect()
204
205
206 # if __name__ == '__main__':
207 #     import sys
208 #     app = QtWidgets.QApplication(sys.argv)
209 #     m = WidgetMain()
210 #     m.show()
211 #     sys.exit(app.exec_())
212

```