PyQt-Frameless-Window

Release v0.2.1

zhiyiYo

Aug 29, 2023

CONTENTS

CHAPTER

ONE

QUICK START

1.1 Install

For PyQt5:

```
pip install PyQt5-Frameless-Window
```

For PyQt6:

```
pip install PyQt6-Frameless-Window
```

For PySide2:

```
pip install PySide2-Frameless-Window
```

For PySide6:

```
pip install PySideSix-Frameless-Window
```

1.2 Requirements

CHAPTER

TWO

USAGE

2.1 Minimal example

To use the frameless window, we only need to inherit `FramelessWindow` or `FramelessMainWindow`. Here is a minimal example:

```
import sys

from PyQt5.QtWidgets import QApplication
from qframelesswindow import FramelessWindow


class Window(FramelessWindow):

    def __init__(self, parent=None):
        super().__init__(parent=parent)
        self.setWindowTitle("PyQt-Frameless-Window")
        self.titleBar.raise_()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    demo = Window()
    demo.show()
    sys.exit(app.exec_())
```

For more complex requirements, see demo.py and main_window.py.

2.2 Customize title bar

PyQt-Frameless-Window uses `TitleBar` as the default title bar. `TitleBar` provides the ability to moving window and contains three basic buttons, including minimize button, maximize/restore button and close button. These buttons are inherited from `TitleBarButton`, and we can use `setXXXColor()` or qss to change the style of buttons. Here is an example:

```
from qframelesswindow import FramelessWindow, TitleBar


class CustomTitleBar(TitleBar):
```

(continues on next page)

(continued from previous page)

```
    """ Custom title bar """

    def __init__(self, parent):
        super().__init__(parent)

        # customize the style of title bar button
        self.minBtn.setHoverColor(Qt.white)
        self.minBtn.setHoverBackgroundColor(QColor(0, 100, 182))
        self.minBtn.setPressedColor(Qt.white)
        self.minBtn.setPressedBackgroundColor(QColor(54, 57, 65))

        # use qss to customize title bar button
        self.maxBtn.setStyleSheet("""
            TitleBarButton {
                qproperty-normalColor: black;
                qproperty-normalBackgroundColor: transparent;
                qproperty-hoverColor: white;
                qproperty-hoverBackgroundColor: rgb(0, 100, 182);
                qproperty-pressedColor: white;
                qproperty-pressedBackgroundColor: rgb(54, 57, 65);
            }
        """)


class Window(FramelessWindow):

    def __init__(self, parent=None):
        super().__init__(parent=parent)
        # change the default title
        self.setTitleBar(CustomTitleBar(self))
```

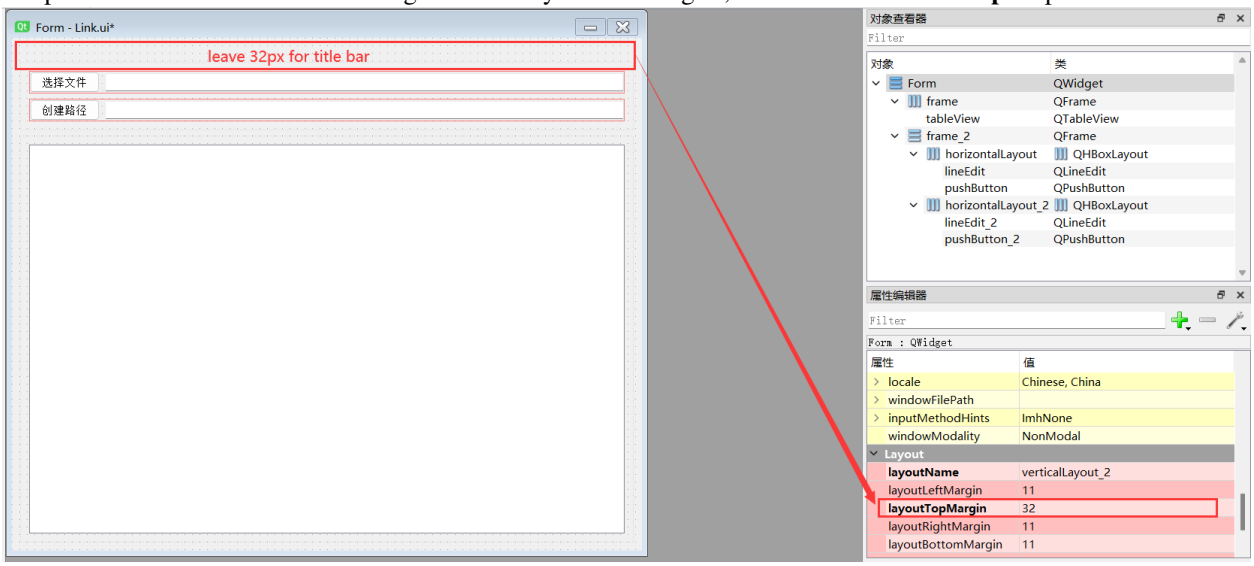
If we want a title bar with icon and title, just replace `TitleBar` with `StandardTitleBar`.

```
from qframelesswindow import FramelessWindow, StandardTitleBar


class Window(FramelessWindow):

    def __init__(self, parent=None):
        super().__init__(parent=parent)
        # replace the default title bar with StandardTitleBar
        self.setTitleBar(StandardTitleBar(self))

        self.setWindowIcon(QIcon("screenshot/logo.png"))
        self.setWindowTitle("PyQt-Frameless-Window")

        # don't forget to put the title bar at the top
        self.titleBar.raise_()
```

When the window icon or title changes, the icon and title of `StandardTitleBar` will also change accordingly. However, we can also use `StandardTitleBar.setTitle()` or `StandardTitleBar.setIcon()` to change them manually.

2.3 Work with Qt Designer

To prevent the title bar from being blocked by other widgets, we need to leave **32px** space for title bar.



After compiling the ui file into a Ui class, we can use the frameless window through multiple inheritance. Here is an example:

```
class Ui_Form(object):
    def setupUi(self, Form):
        Form.resize(400, 423)
        self.verticalLayout_2 = QVBoxLayout(Form)
        self.verticalLayout_2.setContentsMargins(-1, 32, -1, -1)
        self.frame_2 = QFrame(Form)
        self.verticalLayout = QVBoxLayout(self.frame_2)
        self.horizontalLayout = QHBoxLayout()
        self.pushButton = QPushButton(self.frame_2)
        self.horizontalLayout.addWidget(self.pushButton)
        self.lineEdit = QLineEdit(self.frame_2)
        self.horizontalLayout.addWidget(self.lineEdit)
        self.verticalLayout.addLayout(self.horizontalLayout)
        self.horizontalLayout_2 = QHBoxLayout()
        self.pushButton_2 = QPushButton(self.frame_2)
        self.horizontalLayout_2.addWidget(self.pushButton_2)
        self.lineEdit_2 = QLineEdit(self.frame_2)
        self.horizontalLayout_2.addWidget(self.lineEdit_2)
        self.verticalLayout.addLayout(self.horizontalLayout_2)
        self.verticalLayout_2.addWidget(self.frame_2)
        self.frame = QFrame(Form)
        self.horizontalLayout_3 = QHBoxLayout(self.frame)
        self.tableView = QTableView(self.frame)
        self.horizontalLayout_3.addWidget(self.tableView)
        self.verticalLayout_2.addWidget(self.frame)

        self.retranslateUi(Form)

    def retranslateUi(self, Form):
```

(continues on next page)

(continued from previous page)

```
        _translate = QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.pushButton.setText(_translate("Form", ""))
        self.pushButton_2.setText(_translate("Form", ""))


class Window(FramelessWindow, Ui_Form):

    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupUi(self)
```

CHAPTER

THREE

WINDOW EFFECT

PyQt-Frameless-Window use `WindowEffect` class to control the effect of frameless window. You can add shadow, animation or blur effect to window through `WindowEffect`.

3.1 Acrylic effect

PyQt-Frameless-Window provides the `AcrylicWindow` class, which uses the acrylic blur effect.

Here is an minimal example:

```
from qframelesswindow import AcrylicWindow


class Window(AcrylicWindow):

    def __init__(self, parent=None):
        super().__init__(parent=parent)
        self.setWindowTitle("Acrylic Window")
```

(continues on next page)

(continued from previous page)

```
        self.titleBar.raise_()

        # customize acrylic effect
        # self.windowEffect.setAcrylicEffect(self.winId(), "106EBE99")

        # you can also enable mica effect on Win11
        # self.windowEffect.setMicaEffect(self.winId(), False)
```

Because moving or resizing the acrylic window on Win10 may get stuck, we can use the following method to toggle acrylic effect:

```
def setAcrylicEffectEnabled(self, enable: bool):
    """ set acrylic effect enabled """
    self.setStyleSheet(f"background:{'transparent' if enable else '#F2F2F2'}")
    if enable:
        self.windowEffect.setAcrylicEffect(self.winId(), "F2F2F299")
        if QOperatingSystemVersion.current() != QOperatingSystemVersion.Windows10:
            self.windowEffect.addShadowEffect(self.winId())
    else:
        self.windowEffect.addShadowEffect(self.winId())
        self.windowEffect.removeBackgroundEffect(self.winId())
```

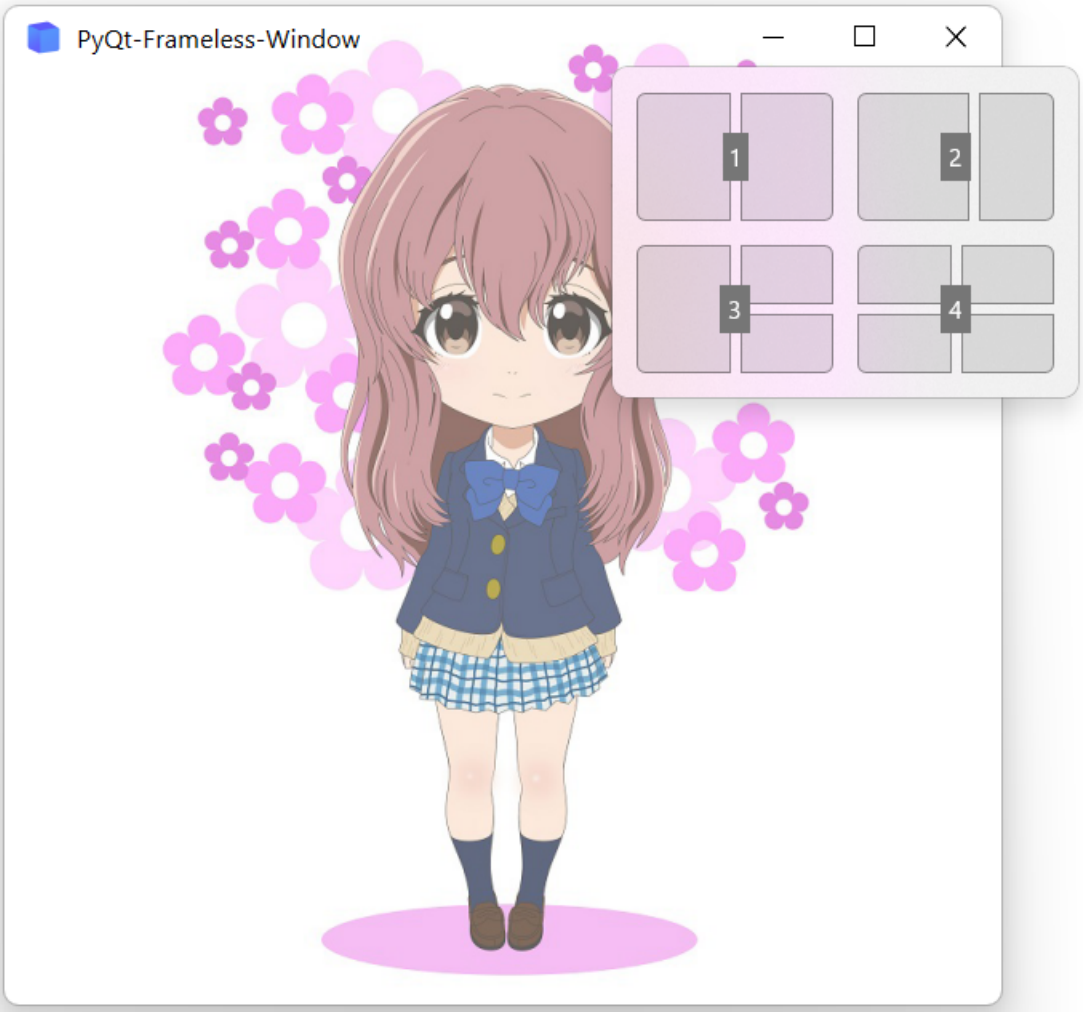
CHAPTER

FOUR

SNAP LAYOUT

4.1 Description

Snap layouts are a new Windows 11 feature to help introduce users to the power of window snapping. Snap layouts are easily accessible by hovering the mouse over a window's maximize button or pressing Win + Z. After invoking the menu that shows the available layouts, users can click on a zone in a layout to snap a window to that particular zone and then use Snap Assist to finish building an entire layout of windows.



4.2 Implementation

PyQt-Frameless-Window does not enable the snap layout feature by default, because user may change the maximize button in the title bar. Here is an example shows how to enable snap layout when using the default title bar. You should replace `WindowsFramelessWindow.nativeEvent()` in `qframelesswindow/windows/__init__.py` with the following code:

```
from ..titlebar.title_bar_buttons import TitleBarButtonState

def nativeEvent(self, eventType, message):
    """ Handle the Windows message """
    msg = MSG.from_address(message.__int__())
```

(continues on next page)

(continued from previous page)

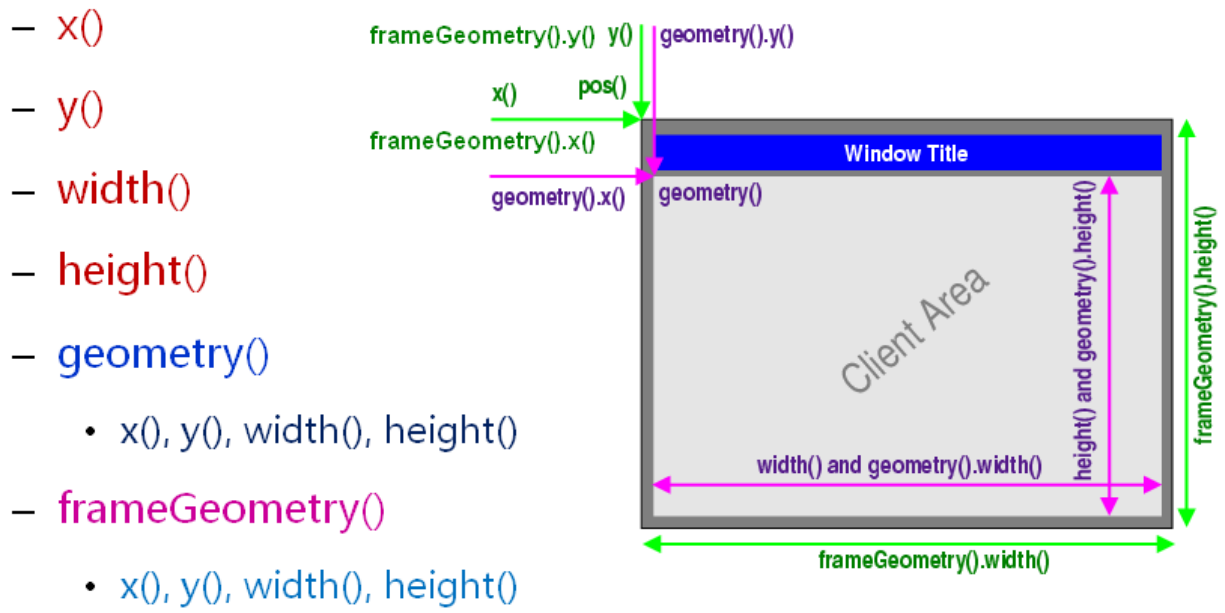
```
    if not msg.hWnd:
        return super().nativeEvent(eventType, message)

    if msg.message == win32con.WM_NCHITTEST and self._isResizeEnabled:
        pos = QCursor.pos()
        xPos = pos.x() - self.x()
        yPos = pos.y() - self.y()
        w, h = self.width(), self.height()
        lx = xPos < self.BORDER_WIDTH
        rx = xPos > w - self.BORDER_WIDTH
        ty = yPos < self.BORDER_WIDTH
        by = yPos > h - self.BORDER_WIDTH
        if lx and ty:
            return True, win32con.HTTOPLEFT
        elif rx and by:
            return True, win32con.HTBOTTOMRIGHT
        elif rx and ty:
            return True, win32con.HTTOPRIGHT
        elif lx and by:
            return True, win32con.HTBOTTOMLEFT
        elif ty:
            return True, win32con.HTTOP
        elif by:
            return True, win32con.HTBOTTOM
        elif lx:
            return True, win32con.HTLEFT
        elif rx:
            return True, win32con.HTRIGHT

    #------------------------------------ ADDED CODE ---------------------------------
↪-----#
        elif self.titleBar.childAt(pos-self.geometry().topLeft()) is self.titleBar.
↪maxBtn:
            self.titleBar.maxBtn.setState(TitleBarButtonState.HOVER)
            return True, win32con.HTMAXBUTTON
    elif msg.message in [0x2A2, win32con.WM_MOUSELEAVE]:
        self.titleBar.maxBtn.setState(TitleBarButtonState.NORMAL)
    elif msg.message in [win32con.WM_NCLBUTTONDOWN, win32con.WM_NCLBUTTONDBLCLK]:
        if self.titleBar.childAt(QCursor.pos()-self.geometry().topLeft()) is self.
↪titleBar.maxBtn:
            QApplication.sendEvent(self.titleBar.maxBtn, QMouseEvent(
                QEvent.MouseButtonPress, QPoint(), Qt.LeftButton, Qt.LeftButton, Qt.
↪NoModifier))
            return True, 0
    elif msg.message in [win32con.WM_NCLBUTTONUP, win32con.WM_NCRBUTTONUP]:
        if self.titleBar.childAt(QCursor.pos()-self.geometry().topLeft()) is self.
↪titleBar.maxBtn:
            QApplication.sendEvent(self.titleBar.maxBtn, QMouseEvent(
                QEvent.MouseButtonRelease, QPoint(), Qt.LeftButton, Qt.LeftButton, Qt.
↪NoModifier))
    #----------------------------------------------------------------------------------
↪------#
```

(continues on next page)

(continued from previous page)

```
    elif msg.message == win32con.WM_NCCALCSIZE:
        if msg.wParam:
            rect = cast(msg.lParam, LPNCCALCSIZE_PARAMS).contents.rgrc[0]
        else:
            rect = cast(msg.lParam, LPRECT).contents

        isMax = win_utils.isMaximized(msg.hWnd)
        isFull = win_utils.isFullScreen(msg.hWnd)

        # adjust the size of client rect
        if isMax and not isFull:
            thickness = win_utils.getResizeBorderThickness(msg.hWnd)
            rect.top += thickness
            rect.left += thickness
            rect.right -= thickness
            rect.bottom -= thickness

        # handle the situation that an auto-hide taskbar is enabled
        if (isMax or isFull) and Taskbar.isAutoHide():
            position = Taskbar.getPosition(msg.hWnd)
            if position == Taskbar.LEFT:
                rect.top += Taskbar.AUTO_HIDE_THICKNESS
            elif position == Taskbar.BOTTOM:
                rect.bottom -= Taskbar.AUTO_HIDE_THICKNESS
            elif position == Taskbar.LEFT:
                rect.left += Taskbar.AUTO_HIDE_THICKNESS
            elif position == Taskbar.RIGHT:
                rect.right -= Taskbar.AUTO_HIDE_THICKNESS

        result = 0 if not msg.wParam else win32con.WVR_REDRAW
        return True, result

    return super().nativeEvent(eventType, message)
```

We use `self.titleBar.childAt(pos-self.geometry().topLeft())` rather than `self.titleBar.childAt(xPos, yPos)`, because the size of frameless window will be larger than the screen when the window is maximized.



You can also inherit `FramelessWindow` to rewrite the `nativeEvent` in your project:

```
import sys

if sys.platform != "win32":
    from qframelesswindow import FramelessWindow
else:
    from ctypes.wintypes import MSG

    import win32con
    from PyQt5.QtCore import QPoint, QEvent, Qt
    from PyQt5.QtGui import QCursor, QMouseEvent
    from PyQt5.QtWidgets import QApplication

    from qframelesswindow import FramelessWindow as Window
    from qframelesswindow.titlebar.title_bar_buttons import TitleBarButtonState


    class FramelessWindow(Window):
        """ Frameless window """

        def nativeEvent(self, eventType, message):
            """ Handle the Windows message """
            msg = MSG.from_address(message.__int__())
            if not msg.hWnd:
                return super().nativeEvent(eventType, message)

            if msg.message == win32con.WM_NCHITTEST and self._isResizeEnabled:
                if self._isHoverMaxBtn():
                    self.titleBar.maxBtn.setState(TitleBarButtonState.HOVER)
                    return True, win32con.HTMAXBUTTON
```

(continues on next page)

(continued from previous page)

```
            elif msg.message in [0x2A2, win32con.WM_MOUSELEAVE]:
                self.titleBar.maxBtn.setState(TitleBarButtonState.NORMAL)
            elif msg.message in [win32con.WM_NCLBUTTONDOWN, win32con.WM_NCLBUTTONDBLCLK]␣
↪and self._isHoverMaxBtn():
                e = QMouseEvent(QEvent.MouseButtonPress, QPoint(), Qt.LeftButton, Qt.
↪LeftButton, Qt.NoModifier)
                QApplication.sendEvent(self.titleBar.maxBtn, e)
                return True, 0
            elif msg.message in [win32con.WM_NCLBUTTONUP, win32con.WM_NCRBUTTONUP] and␣
↪self._isHoverMaxBtn():
                e = QMouseEvent(QEvent.MouseButtonRelease, QPoint(), Qt.LeftButton, Qt.
↪LeftButton, Qt.NoModifier)
                QApplication.sendEvent(self.titleBar.maxBtn, e)

            return super().nativeEvent(eventType, message)

        def _isHoverMaxBtn(self):
            pos = QCursor.pos() - self.geometry().topLeft() - self.titleBar.pos()
            return self.titleBar.childAt(pos) is self.titleBar.maxBtn
```

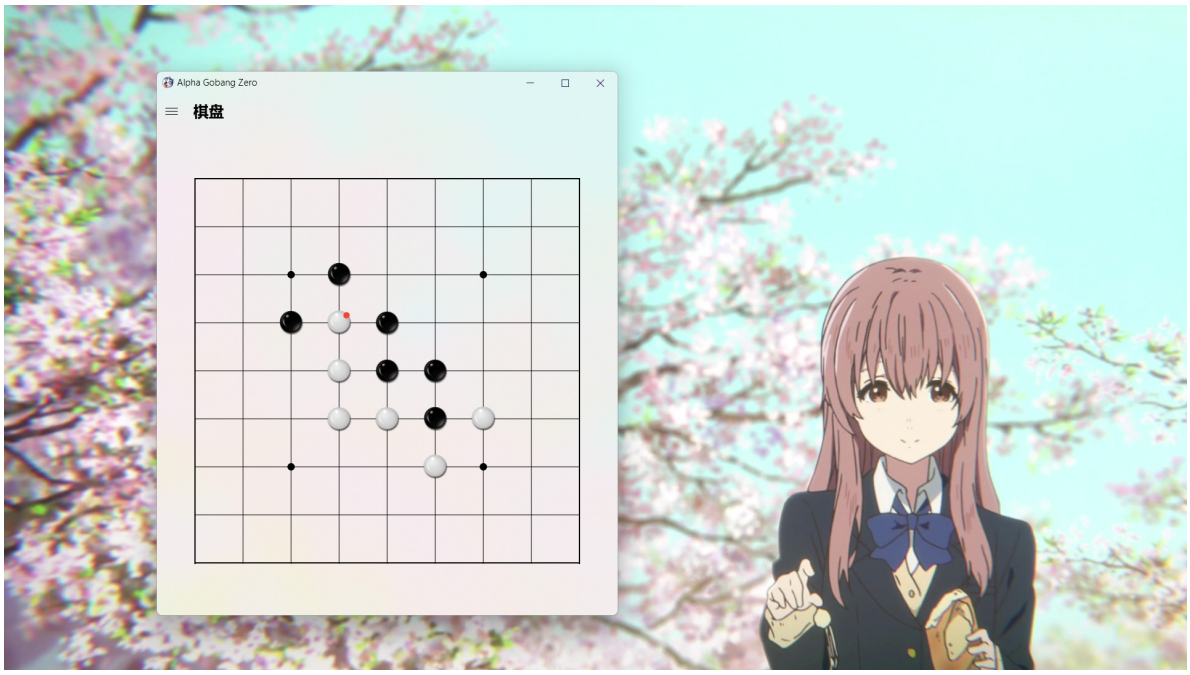
CHAPTER

FIVE

SEE ALSO

Here are some projects that use PyQt-Frameless-Window:

- **zhiyiYo/Groove**: A cross-platform music player based on PyQt5
- **zhiyiYo/Alpha-Gobang-Zero**: A gobang robot based on reinforcement learning



- **zhiyiYo/PyQt-Fluent-Widgets**: A fluent design widgets library based on PyQt5