**Управление окном приложения**

***Разворачивание и сворачивание окна***

В заголовке окна расположены кнопки Свернуть и Развернуть, с помощью которых можно свернуть окно в значок на панели задач или развернуть его на весь экран. Выполнить подобные действия из программы позволяют следующие методы класса QWidget:

**showMinimized()** — сворачивает окно на панель задач. Эквивалентно нажатию кнопки свернуть в заголовке окна;

**showMaximized()** — разворачивает окно до максимального размера. Эквивалентно нажатию кнопки Развернуть в заголовке окна;

**showFullScreen()** — включает полноэкранный режим отображения окна. Окно отображается без заголовка и границ;

**showNormal()** — отменяет сворачивание, максимальный размер и полноэкранный режим, возвращая окно к изначальным размерам;

**activateWindow()** — делает окно активным (т. е. имеющим фокус ввода). В Windows, если окно было ранее свернуто в значок на панель задач, оно не будет развернуто в изначальный вид.

**setWindowState(<Флаги>)** — изменяет состояние окна в зависимости от переданных флагов. В качестве параметра указывается комбинация следующих атрибутов из класса QtCore.Qt через побитовые операторы:

* WindowNoState— нормальное состояние окна;
* WindowMinimized — окно свернуто;
* WindowMaximized — окно максимально развернуто;
* WindowFullScreen — полноэкранный режим.

Проверить текущий статус окна позволяют следующие методы:

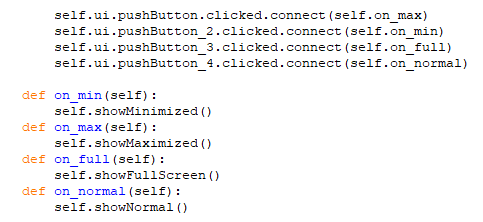
* isMinimized() — возвращает True, если окно свернуто, и False — в противном случае;
* isMaximized() — возвращает True, если окно раскрыто до максимальных размеров, и False — в противном случае;
* isFullScreen() — возвращает True, если включен полноэкранный режим, и False — в противном случае;
* isActiveWindow() — возвращает True, если окно имеет фокус ввода, и False — в противном случае;
* windowState() — возвращает комбинацию флагов, обозначающих текущий статус окна.

Пример проверки использования полноэкранного режима:

if window.windowState() & QtCore.Qt.WindowFullScreen:

print ("Полноэкранный режим")

Пример разворачивания и сворачивания окна приведен в листинге 1.



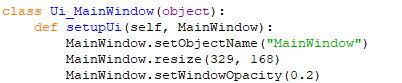
Листинг 1.Разворачивание и сворачивание окна

***Управление прозрачностью окна***

Сделать окно полупрозрачным позволяет метод setWindowOpacity() класса QWidget. Формат метода:

setWindowOpacity(<Вещественное число от 0.0 до 1.0>)

Число 0.0 соответствует полностью прозрачному окну, а число 1.0 — отсутствию прозрачности. Для получения степени прозрачности окна из программы предназначен метод windowOpacity() (см. листинг 2).



Листинг 2. Полупрозрачное окно

***Модальные окна***

***Модальным*** называется окно, которое не позволяет взаимодействовать с другими окнами в том же приложении, — пока модальное окно не будет закрыто, сделать активным другое окно нельзя.

Указать, что окно является модальным, позволяет метод setWindowModality(<Флаг>) из класса QWidget. В качестве параметра могут быть указаны следующие атрибуты из класса QtCore.Qt:

NonModal — 0 — окно не является модальным. Это поведение окна по умолчанию;

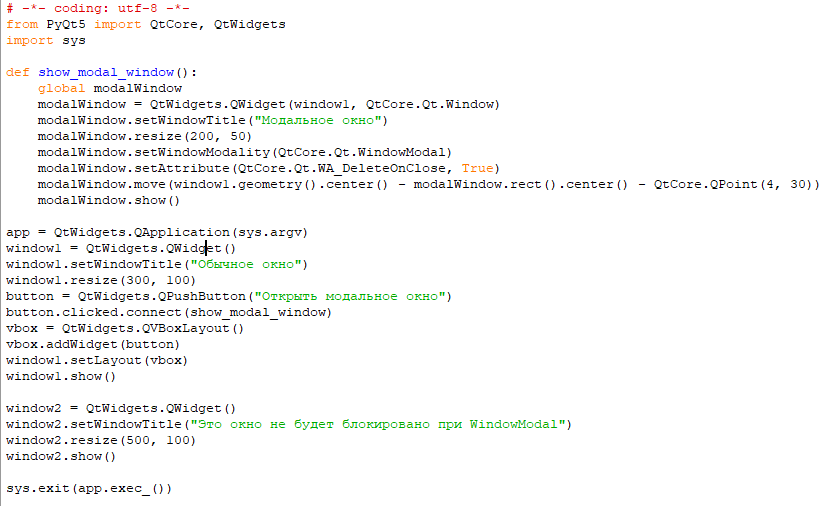
WindowModal — 1 — окно блокирует только родительские окна в пределах иерархии;

ApplicationModal — 2 — окно блокирует все окна в приложении.

Окна, открытые из модального окна, не блокируются. Следует также учитывать, что метод setWindowModality() должен быть вызван до отображения окна.

Получить текущее значение модальности позволяет метод windowModality(). Проверить, является ли окно модальным, можно с помощью метода isModal() — он возвращает True, если окно является модальным, и False — в противном случае.

Создадим два независимых окна. В первом окне разместим кнопку, по нажатию которой откроется модальное окно, — оно будет блокировать только первое окно, но не второе. При открытии модального окна отобразим его примерно по центру родительского окна (листинг 3).

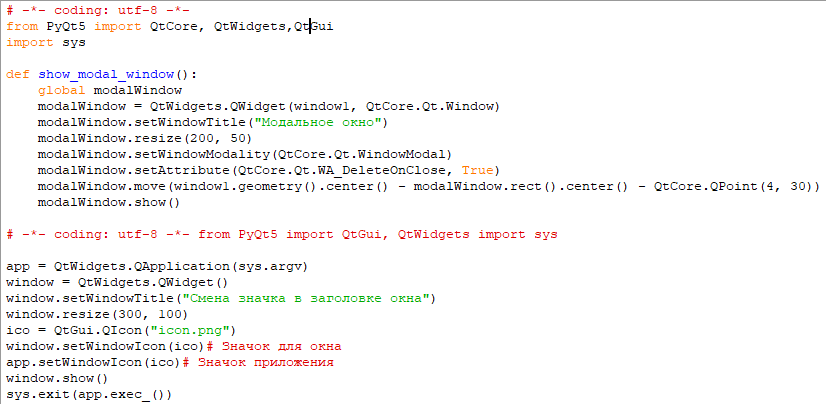


Листинг 3. Модальные окна

***Смена значка в заголовке окна***

По умолчанию в левом верхнем углу окна отображается стандартный значок. Отобразить другой значок позволяет метод setWindowIcon() класса QWidget. В качестве параметра метод принимает экземпляр класса QIcon из модуля QtGui. Чтобы загрузить значок из файла, следует передать путь к файлу конструктору этого класса.

Получить список поддерживаемых форматов файлов можно с помощью статического метода supportedImageFormats() класса QImageReader, объявленного в модуле QtGui. Метод возвращает список с экземплярами класса QByteArray. Список поддерживаемых форматов: BMP, CUR, DDS, GIF, ICNS, ICO, JP2, JPEG, JPG, MNG, PBM, PGM, PNG, PPM, SVG, SVGZ, TGA, TIF, TIFF, WEBP, XBM, XPM.



Листинг 4. Смена значка в заголовке окна

***Изменение цвета фона окна***

Чтобы изменить цвет фона окна (или компонента), следует установить палитру с настроенной ролью Window (или Background). Цветовая палитра содержит цвета для каждой роли и состояния компонента. Указать состояние компонента позволяют следующие атрибуты из класса QPalette (модуль QtGui):

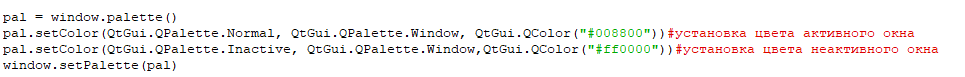
* Active и Normal — 0 — компонент активен (окно находится в фокусе ввода);
* Disabled — 1 — компонент недоступен;
* Inactive — 2 — компонент неактивен (окно находится вне фокуса ввода).

Получить текущую палитру компонента позволяет его метод palette(). Чтобы изменить цвет для какой-либо роли и состояния, следует воспользоваться методом setColor() класса QPalette. Формат метода:

setColor([<Состояние>, ]<Роль>, <Цвет>)

В параметре <Роль> указывается, для какого элемента изменяется цвет. Например, атрибут Window (или Background) изменяет цвет фона, а WindowText (или Foreground) — цвет текста. Полный список атрибутов имеется в документации по классу QPalette (см. http://doc.qt.io/ qt-5/qpalette.html).

В параметре <Цвет> указывается цвет элемента. В качестве значения можно указать атрибут из класса QtCore.Qt (например, black, white и т. д.) или экземпляр класса QColor (например, QColor("red"), QColor("#ff0000"), QColor(255, 0, 0) и др.).

Листинг 5. Изменение цвета фона окна

***Вывод изображения в качестве фона***

В качестве фона окна (или компонента) можно использовать изображение. Для этого необходимо получить текущую палитру компонента с помощью метода palette(), а затем вызвать метод setBrush() класса QPalette. Формат метода:

setBrush([<Состояние>, ]<Роль>, <QBrush>)

В третьем параметре указывается кисть — экземпляр класса QBrush из модуля QtGui.

В параметре <Стиль кисти> указываются атрибуты из класса QtCore.Qt, задающие стиль кисти, — например: NoBrush, SolidPattern, Dense1Pattern, Dense2Pattern, Dense3Pattern, Dense4Pattern, Dense5Pattern, Dense6Pattern, Dense7Pattern, CrossPattern и др. С помощью этого параметра можно сделать цвет сплошным (SolidPattern) или имеющим текстуру (например, атрибут CrossPattern задает текстуру в виде сетки).

В параметре <Цвет> указывается цвет кисти. В качестве значения можно указать атрибут из класса QtCore.Qt (например, black, white и т. д.) или экземпляр класса QColor (например, QColor("red"), QColor("#ff0000"), QColor(255, 0, 0) и др.). При этом установка сплошного цвета фона окна может выглядеть так:

pal = window.palette()

pal.setBrush(QtGui.QPalette.Normal,QtGui.QPalette.Window, QtGui.QBrush(QtGui.QColor("#008800"), QtCore.Qt.SolidPattern))

window.setPalette(pal)

Параметры <QPixmap> и <QImage> позволяют передать объекты изображений. Конструкторы этих классов принимают путь к файлу, который может быть как абсолютным, так и относительным.



Листинг 6. Использование изображения в качестве фона

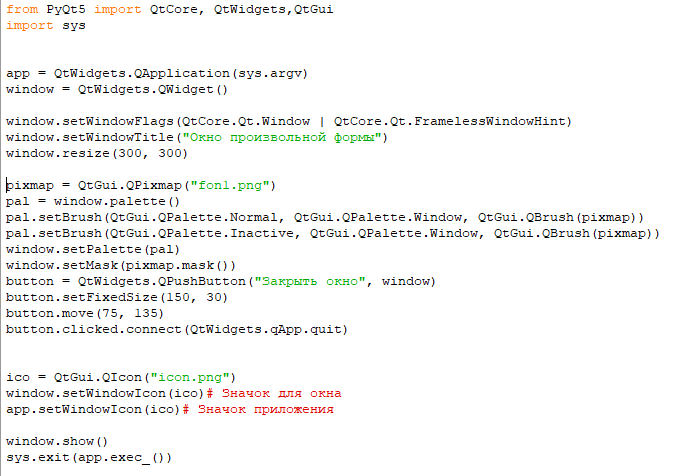
***Создание окна произвольной формы***

Чтобы создать окно произвольной формы, нужно выполнить следующие шаги:

1. Создать изображение нужной формы с прозрачным фоном и сохранить его, например, в формате PNG.
2. Создать экземпляр класса QPixmap, передав конструктору класса абсолютный или относительный путь к изображению.
3. Установить изображение в качестве фона окна с помощью палитры.
4. Отделить альфа-канал с помощью метода mask() класса QPixmap.
5. Передать получившуюся маску в метод setMask() окна.

Убрать рамку окна, например, передав комбинацию следующих флагов:

QtCore.Qt.Window | QtCore.Qt.FramelessWindowHint



Листинг 7. Создание окна произвольной формы

Получившееся окно можно увидеть на рис. 21.1.

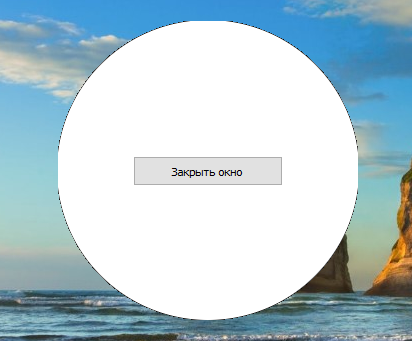


Рис 1. Окно круглой формы

***Всплывающие подсказки***

Всплывающая подсказка - подсказка в отдельном окне без рамки при наведении указателя мыши на компонент. Подсказка автоматически скроется после увода курсора мыши или спустя определенное время. Чтобы создать всплывающие подсказки для окна или любого другого компонента и управлять ими, нужно воспользоваться следующими методами из класса QWidget:

***setToolTip(<Текст>)*** — задает текст всплывающей подсказки. В качестве параметра можно указать простой текст или HTML-код. Чтобы отключить вывод подсказки, достаточно передать в этот метод пустую строку;

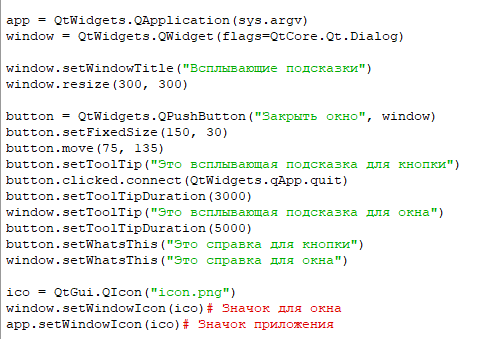
***toolTip()*** — возвращает текст всплывающей подсказки;

***setToolTipDuration(<Время>)***— задает время, в течение которого всплывающая подсказка будет присутствовать на экране. Значение должно быть указано в миллисекундах. Если задать значение -1, PyQt будет сама вычислять необходимое время, основываясь на длине текста подсказки (это поведение по умолчанию);

***toolTipDuration()*** — возвращает время, в течение которого всплывающая подсказка будет присутствовать на экране;

***setWhatsThis(<Текст>)*** — задает текст справки. Обычно этот метод используется для вывода информации большего объема, чем во всплывающей подсказке. У диалоговых окон в заголовке окна есть кнопка Справка, по нажатию которой курсор принимает вид стрелки со знаком вопроса, — чтобы в таком случае отобразить текст справки, следует нажать эту кнопку и щелкнуть на компоненте. Можно также сделать компонент активным и нажать комбинацию клавиш <Shift>+<F1>. В качестве параметра можно указать простой текст или HTML-код. Чтобы отключить вывод подсказки, достаточно передать в этот метод пустую строку;

***whatsThis()*** — возвращает текст справки. Создадим окно с кнопкой и зададим для них текст всплывающих подсказок и текст справки (листинг 21.11).



Листинг 8. Всплывающие подсказки