**Глава 33**

**Создание компонентов**

**во время выполнения программы:**

**проект HTOWERS**

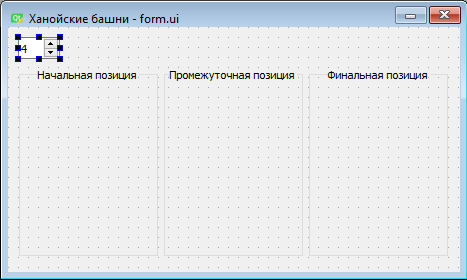
Завершающий пример книги, проект HTOWERS, представляет собой компью­терную реализацию известной логической задачи "Ханойские башни". Эта за­дача состоит в следующем: имеются три колышка, на один из которых нани­зано (в порядке уменьшения размера) несколько дисков, образующих "башню"; требуется переместить всю башню на один из пустых колышков, пользуясь другим пустым колышком как вспомогательным. Переносить можно по одному диску, причем больший диск нельзя помещать на меньший. Заметим, что для решения задачи с *N* дисками минимальное число переносов равно 2*N* – 1 (см. [2]).

В приложении HTOWERS вместо колышков используются компоненты-кон­тейнеры QGroupBox, а вместо дисков перемещаются прямоугольные блоки (компоненты QLabel), создаваемые непосредственно в ходе выполнения про­граммы. Перемещение блоков реализуется с помощью механизма drag & drop. Предусматриваются средства проверки правильности решения, а также демо-режим, показывающий, в каком порядке надо перемещать блоки, чтобы ре­шить задачу. В заключительном разделе главы описываются действия, позво­ляющие задать идентификационные данные для разработанного приложения (название приложения, его краткое описание, сведения об авторе, номер вер­сии).

**33.1. Создание начальной позиции**

После создания формы Form типа Widget разместите в ней три компонента типа QGroupBox (они получат имена groupBox, groupBox\_2 и groupBox\_3) и компонент типа QSpinBox (он получит имя spinBox). Настройте свойства формы Form и добавленных компонентов (листинг 33.1) и расположите ком­поненты в соответствии с рис. 33.1.

Определите конструктор и функцию load для формы Form (листинг 33.2). До­бавьте стандартный метод main для приложений с графическим интерфесом и выполните стандартные команды создания файла с заготовкой для формы и запуска приложения.



**Рис. 33.1.** Вид формы Form для проекта HTOWERS

на начальном этапе разработки

**Листинг 33.1. Настройка свойств**

**оржрождорж**

Form: objectName = **Form**, windowTitle = **Ханойские башни**

groupBox1: title = **Начальная позиция**

groupBox2: title = **Промежуточная позиция**

groupBox3: title = **Конечная позиция**

spinBox: minimum = **2**, maximum = **10**, value = **4**

**Листинг 33.2. Подключение модулей и методы класса Form**

**оржрождорж**

**from ui\_form import \***

**from PyQt5.QtWidgets import \***

**import random, sys**

**class Form(Ui\_Form, QWidget):**

**def \_\_init\_\_(self):**

**super(self.\_\_class\_\_, self).\_\_init\_\_()**

**self.setupUi(self)**

**self.setFixedSize(self.size())**

**self.load()**

**def load(self):**

**n = self.spinBox.value()**

**w = self.groupBox.width()**

**h = self.groupBox.height()**

**for i in range(0, n):**

**lb = QLabel(self.groupBox)**

**lb.setAutoFillBackground(True)**

**lb.setFrameShape(QFrame.Box)**

**lb.resize((w - 10) \* (n - i) / n, (h - 15) / n)**

**lb.move((w - lb.width()) / 2, h - 2 - (i + 1) \* lb.height())**

**lb.setStyleSheet("background-color: RGB({}, {}, {})"**

**.format(random.randint(0, 255), random.randint(0, 255),**

**random.randint(0, 255)))**

**lb.show()**

**Результат.** При запуске программы на компоненте groupBox рисуется башня из четырех разноцветных блоков — компонентов типа QLabel (цвет блоков определяется случайным образом).

**Комментарии**

1. Компоненты-метки QLabel создаются непосредственно при выполнении метода load. Метка создается с помощью конструктора QLabel c парамет­ром родительского компонента, в котором будет изображена созданная метка, стиль ее границы, размер, положение и фоновый цвет.
2. Отметим, что мы не использовали менеджеры компоновки для размещения компонентов-контейнеров QGroupBox, так как при их использовании пере­менные w и h получают несоответствующие реальной ситуации значения. Так как форма имеет постоянные размеры, можно сначала добавить контей­неры в вертикальный лейаут, а затем удалить его. После удаления мене­джера компоновки виджеты сохранят необходимые координаты и размеры.
3. По умолчанию созданные метки являются невидимыми, поэтому для их отображения на экране необходим вызов метода show().

**33.2. Создание начальной позиции**

Добавьте в конструктор класса Form соединение сигнала valueChanged() ви­джета spinBox со слотом load() формы Form (листинг 33.3).

**Листинг 33.3. Соединение сигнала valueChanged() и слота load()**

**оржрождорж**

def \_\_init\_\_(self):

super(self.\_\_class\_\_, self).\_\_init\_\_()

self.setupUi(self)

self.setFixedSize(self.size())

self.load()

**self.spinBox.valueChanged.connect(self.load)**

**Результат.** При изменении значения компонента spinBox создается новая башня с указанным числом блоков.

**Ошибка.** Новая башня рисуется под старой.

**Исправление.** В начало метода load() вставьте следующие операторы:

**for i in range(len(self.groupBox.children())):**

**self.groupBox.children()[i].deleteLater()**

**Результат.** Теперь все блоки старой башни исчезают с экрана, а также осво­бождают свои ресурсы.

**Комментарии**

1. Для удаления блоков старой башни требуется удалить дочерние элементы компонента groupBox, чей список можно получить с помощью метода children(). Метод deleteLater() заставляет уничтожить объект после того, как все ожидаемые объектом события будут ему предоставлены.

**33.3. Перетаскивание блоков на новое место**

Откройте в любом текстовом редакторе файл form.ui. Добавьте после тега widget код для DragDropGroupBox - наследника класса QGroupBox, в котором будет реализован механизм drag & drop (листинг 33.4). Поменяйте у компо­нентов groupBox, groupBox\_2 и groupBox\_3 тип QGroupBox на DragDropGroupBox.

Cоздайте отдельные файлы для классов DragDropGroupBox (листинг 33.5) и DragDropLabel (листинг 33.6).

**Листинг 33.4. XML-код для класса DragDropGroupBox**

**оржрождорж**

**<customwidgets>**

**<customwidget>**

**<class>DragDropGroupBox</class>**

**<extends>QGroupBox</extends>**

**<header>dragdropgroupBox</header>**

**</customwidget>**

**</customwidgets>**

**Листинг 33.5. Класс DragDropGroupBox**

**оржрождорж**

**class DragDropGroupBox(QGroupBox):**

**def \_\_init\_\_(self, parent = None):**

**super(self.\_\_class\_\_, self).\_\_init\_\_(parent)**

**def dragEnterEvent(self, event):**

**event.acceptProposedAction()**

**def dropEvent(self, event):**

**lb = event.source()**

**lb.setParent(self)**

**lb.move((self.width() - lb.width()) / 2,**

**self.height() - 2 – len(self.children()) \* lb.height())**

**lb.show()**

**Листинг 33.6. Класс DragDropLabel**

**оржрождорж**

**class DragDropLabel(QLabel):**

**def \_\_init\_\_(self, parent = None):**

**super(self.\_\_class\_\_, self).\_\_init\_\_(parent)**

**def mousePressEvent(self, event):**

**if event.button() == Qt.LeftButton:**

**drag = QDrag(self)**

**drag.setMimeData(QMimeData())**

**drag.exec\_(Qt.MoveAction)**

**Результат.** Любой блок (т. е. метку типа QLabel) можно переместить на дру­гую башню (т. е. в другой компонент QGroupBox), причем перемещенный блок всегда будет располагаться на вершине башни.

**Ошибка.** Переместить можно не только верхний, но и любой другой блок башни.

**Исправление.** Измените метод mousePressEvent класcа DragDropLabel (ли­стинг 33.7).

**Листинг 33.7. Корректировка метода mousePressEvent**

**оржрождорж**

def mousePressEvent(self, event):

if event.button() == Qt.LeftButton:

**index = self.parent().children().index(self)**

drag = QDrag(self)

drag.setMimeData(QMimeData())

**if index == len(self.parent().children()) - 1:**

drag.exec\_(Qt.MoveAction)

**else:**

**drag.exec\_(Qt.IgnoreAction)**

**Результат.** Теперь переместить можно только верхний блок башни; при по-пытке переместить нижние блоки курсор перетаскивания становится запре-щающим.

***ПРИМЕЧАНИЕ***

Мы воспользовались тем обстоятельством, что при добавлении дочернего компо­нента в коллекцию children родительского компонента добавленный компонент располагается в ее конце; таким образом, верхний элемент башни всегда имеет в кол­лекции children индекс, равный числу элементов коллекции children минус 1.

Осталось учесть дополнительное условие задачи: блок не может перемещаться

на башню с верхним блоком меньшего размера. Для этого изменим метод dropEvent класса DragDropGroupBox (листинг 33.8).

**Листинг 33.7. Корректировка метода dropEvent**

**оржрождорж**

def dropEvent(self, event):

**k = sys.maxsize**

**if len(self.children()) > 0:**

**k = self.children()[len(self.children()) - 1].width()**

lb = event.source()

**if k > lb.width():**

lb.setParent(self)

lb.move((self.width() - lb.width()) / 2,

self.height() - 2 - len(self.children()) \* lb.height())

lb.show()

**Результат.** При перемещении блока выполняется дополнительное условие.

***ПРИМЕЧАНИЕ***

В последнем варианте метода dropEvent используется переменная k, в которую записывается ширина верхнего блока компонента-приемника или максимальное возможное значение типа int (в Python версии 3 равное sys.maxsize), если компонент-приемник не содержит блоков.

**Ошибка.** При изменении количества меток с помощью компонента spinBox удаляются только те метки, которые находятся в компоненте groupBox.

**Исправление.** Добавьте после цикла по дочерним элементам groupBox аналогичные циклы для groupBox\_2 и groupBox\_3.

**33.4. Восстановление начальной позиции**

**и подсчет числа перемещений блоков**

Разместите в форме Form кнопку pushButton, свойству text кнопки присвойте значение Сброс и свяжите в коде сигнал clicked() кнопки с существующей функцией load(). Кроме того, разместите в форме Form метку label (её свойство text изменять не требуется). Расположите добавленные компоненты в соответ­ствии с рис. 33.2.

В конструктор класса Form перед вызовом метода load() добавьте описания двух полей:

**self.count = 0**

**self.minCount = 0**

и вспомогательный метод info() (листинг 33.8).

В метод load() добавьте три оператора:

**self.count = 0**

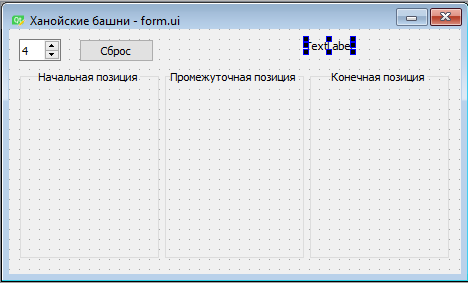
**self.minCount = int(math.pow(2, n) - 1)**

**self.info()**

В метод dropEvent() добавьте два оператора (в условный оператор):

**self.parent().count += 1**

**self.parent().info()**



**Рис. 33.2.** Вид формы Form для проекта HTOWERS

на промежуточном этапе разработки

**Листинг 33.8. Метод info формы Form**

**оржрождорж**

**def info(self):**

**self.label.setText("Число ходов: {} ({})".format(**

**self.count, self.minCount))**

**self.label.adjustSize()**

**Результат.** Для восстановления начальной позиции с тем же количеством блоков следует нажать кнопку **Сброс**. Если при восстановлении начальной позиции требуется изменить число блоков, то по-прежнему достаточно указать новое значение в компоненте spinBox (нажимать кнопку **Сброс** в этом случае не требуется).

Информация о числе ходов (т. е. перемещений блоков) выводится в тексте метки label. Там же в скобках указывается количество ходов, минимально необходимое для решения задачи с данным числом блоков *n* (это количество равно 2*n* – 1).

***ПРИМЕЧАНИЕ***

Обратите внимание на то, что перемещения блока в пределах одного и того же компонента GroupBox при подсчете числа ходов не учитываются.

**Комментарий**

Для нахождения величины 2*n*  была использована функция pow класса math. Поскольку она возвращает результат типа float, полученное значение должно преобразовываться к целому типу.

**33.5. Проверка решения задачи**

Разместите в форме Form под имеющейся меткой label ещё одну метку (label\_2), её свойству text присвойте значение **Задача решена!**, а в настройках свойства palette выберите для роли цвета WindowText зеленый цвет (нажав соответствующую кнопку).

В метод load() добавьте оператор:

**self.label\_2.setVisible(False)**

Добавьте в конец метода dropEvent следующий фрагмент:

**if len(self.parent().groupBox\_3.children()) == self.parent().spinBox.value():**

**self.parent().label\_2.setVisible(True)**

**Результат.** Задача считается решенной и об этом выводится сообщение "Задача решена!", если размер башни в конечной позиции (т. е. в компоненте groupBox\_3) равен общему числу блоков.

**Недочет.** После решения задачи по-прежнему разрешено перемещение блоков.

**Исправление.** Добавьте в начало метода mousePressEvent (см. листинг 33.7) следующий фрагмент:

**if self.parent().parent().label\_2.isVisible():**

**return**

**Результат.** После решения задачи блоки нельзя перемещать.