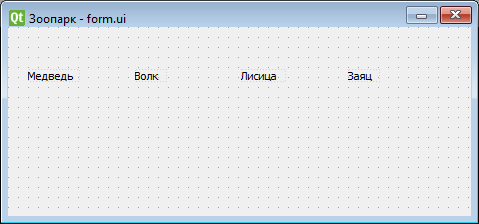
Глава 11

Перетаскивание (drag & drop):  
проект ZOO

Проект ZOO знакомит с различными аспектами режима перетаскивания drag & drop (запуск режима drag & drop и различные варианты его завершения, действия при перетаскивании на недопустимый приемник, дополнительное выделение источника и приемника, настройка вида курсора в режиме пе­ретаскивания).

11.1. Перетаскивание меток по форме

После создания формы типа QWidget разместите в ней четыре метки (label – label\_4) и настройте свойства формы и добавленных меток (листинг 11.1). Положение меток настройте в соответствии с рис. 11.1. Добавьте в текстовом редакторе в форму кастомный класс DragDropLabel (наследник QLabel). Создайте класс DragDropLabel в файле dragdroplabel.py (листинг 11.2). Кроме того, опреде­лите обработчики событий dragEnterEvent и dropEvent класса Form (листинг 11.3). В конструктор формы добавьте оператор фиксации размера.



**Рис. 11.1.** Вид формы Form для проекта ZOO на начальном этапе разработки

**Листинг 11.1. Настройка свойств формы Form и ее компонентов**

Form: text = **Зоопарк**, acceptDrops = **true**

label: text = **Медведь**

label\_2: text = **Волк**

label\_3: text = **Лиса**

label\_4: text = **Заяц**

**Листинг 11.2. Класс DragDropLabel**

**from PyQt5.QtCore import \***

**from PyQt5.QtGui import \***

**from PyQt5.QtWidgets import \***

**class DragDropLabel(QLabel):**

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

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

**drag = QDrag(self)**

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

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

**Листинг 11.3. События dragEnterEvent и dropEvent**

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

**event.accept()**

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

**src = event.source()**

**src.move(event.pos())**

**Результат.** Метки с названиями зверей можно перетаскивать с помощью левой кнопки мыши. При перетаскивании метки-*источника* (source) она остается на месте, однако вид курсора мыши изменя­ется, что является признаком режима перетаскивания. В качестве *приемника* (target) пока определена лишь сама форма: при отпускании над ней кнопки мыши происходит перемещение метки зверя на указанную позицию (см. комментарии 1 и 2).

***ПРИМЕЧАНИЕ***

В данном проекте в текстах обработчиков, связанных с перетаскиванием, будем использовать имена src и trg для объектов-источников и объектов-приемников соответственно.

**Недочет.** Разрешается перетащить метку не только на свободную часть формы, но и на другую метку; при таком перетаскивании происходит наложение двух меток.

**Исправление.** Определите обработчик события dragMoveEvent класса Form (листинг 11.4).

**Листинг 11.4. Событие dragMoveEvent**

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

**trg = self.childAt(event.pos())**

**if (trg == None):**

**event.accept()**

**else:**

**event.ignore()**

**Результат.** В начальный момент перетаскивания курсор мыши остается разрешающим. Перетаскива­ние одной метки на другую по-прежнему запрещено (см. комментарий 3).

Комментарии

1. Для активизации режима перетаскивания класс QDrag имеет метод exec\_, параметр которого типа Qt.DropActions указывает разрешенный результат (эффект) успешного перетаскивания. Предусмотрены пять основных эффекта: CopyAction, MoveAction, LinkAction, TargetMoveAction и IgnoreAction (их названия связаны с вариантами действий при перетаскивании). Если в ходе перетаскивания может возникнуть один из нескольких эффектов (например, CopyAction или MoveAction), то нужно укзать все эти эффекты, объединяя их операцией | (побитовое ИЛИ), например, Qt.CopyAction | Qt.MoveAction. Во втором варианте вызова функции exec\_ существует параметр – действие по умолчанию, которое будет выполняться при указании нескольких эффектов автоматически.

На самом деле, указывать действие MoveAction было необязательно, так как в первом варианте вызова exec\_ это действие указано по умолчанию. Однако для наглядности мы решили дополнительно указать, что выполняется именно перемещение.

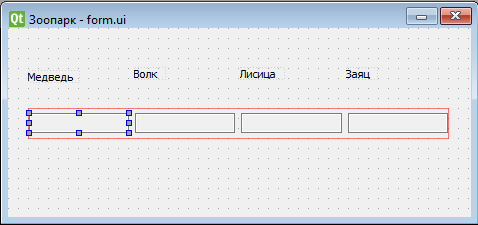
Перед запуском процесса перетаскивания объекту класса QDrag нужно указать объект mimeData. Данное поле хранит информацию о типе MIME (специальный стандарт Multipurpose Internet Mail Extensions) передаваемых данных. Использование данного стандарта обеспечивает безопасность передачи информации между приложениями и копирования внутри одного приложения. Типами MIME может быть текстовая информация (text/plain), html-страница (text/html) или, например, изображение (image/\*). В нашем случае достаточно лишь знать положение курсора (а эта информация может быть получена из событий), поэтому используется конструктор по умолчанию класса QMimeData.

1. Для перемещения метки на новое место необходимо знать позицию курсора мыши при завершении режима перетаскивания. Данная позиция содержится в свойстве pos для параметра event.
2. Установив для формы значение свойства acceptDrops равным True, мы сделали форму допустимым приемником. Поскольку для меток свойство acceptDrops осталось равным False (значение по умолчанию), метки не считаются допустимыми приемниками, т. е. они как бы "невидимы" в режиме перетаскивания. Поэтому, когда курсор мыши в режиме перетаскивания проходил над невидимыми для него компонентами-метками, он считал, что находится над формой, и адресовал события dragEnterEvent и dragDropEvent именно форме. Если в этот момент источник отпускался, то он накладывался на расположенную под ним метку (недочет). Для исправления отмеченного недочета был определен обработчик события dragMoveEvent. В нем выполняется следующая проверка: если в текущей точке дочерних элементов формы не найдено (используется метод childAt, единственный параметр которого – координаты точки), то событие перемещения выполняется, иначе игнорируется (курсор будет иметь вид запрещающего знака).

11.2. Перетаскивание меток в поля ввода

Разместите в форме Form горизонтальный менеджер компоновки, расположите в нем четыре поля ввода типа QLineEdit (lineEdit – lineEdit\_4) и настройте позицию лейаута в соответствии с рис. 11.2. Для каждого поля ввода положите свойства readOnly равным **true** (данная настройка запрещает пользователю редактировать текст в поле ввода, однако не влияет на возможность программного изменения текста) и styleSheet равным **background-color: #f0f0f0**.

Добавьте в текстовом редакторе в форму кастомный класс DragDropLineEdit и определите его в файле dragdroplineedit.py (листинг 11.5).



**Рис. 11.2.** Вид формы Form для проекта ZOO на промежуточном этапе разработки

**Листинг 11.5. Класс DragDropLineEdit**

**from PyQt5.QtCore import \***

**from PyQt5.QtGui import \***

**from PyQt5.QtWidgets import \***

**class DragDropLineEdit(QLineEdit):**

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

**if self.text() == '':**

**event.accept()**

**else:**

**event.ignore()**

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

**src = event.source()**

**self.setText(src.text())**

**src.setVisible(False)**