Модель непосредственного управления

(Drag and drop)

Перетаскивание - по назначению похоже на механизм буфера обмена вырезать и вставить.

Перетаскивание текста

Создайте новый файл в Python (IDLE). Впишите код программы, который перетаскивает текст виджета lineEdit на виджета Button.

```
import sys
from PyQt5.QtWidgets import (QPushButton, QWidget,
    QLineEdit, QApplication)

class Button (QPushButton):
    def __init__(self, title, parent):
        super().__init__(title, parent)
        self.setAcceptDrops(True) #разрешение перетаскивания для виджета

    def dragEnterEvent(self, e): #сообщаем тип данных (текстовый)
        if e.mimeData().hasFormat('text/plain'):
            e.accept()
        else:
            e.ignore()

    def dropEvent(self, e):# определяет что мы делаем после события перетаскивания (меняем текст виджета)
        self.setText(e.mimeData().text())

class Example (QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self): #Виджет QLineEdit имеет встроенную поддержку операций перетаскивания метод setDragEnabled().
        edit = QLineEdit('', self)
        edit.setDragEnabled(True)
        edit.move(30, 65)

        button = Button("Button", self)
        button.move(190, 65)

        self.setWindowTitle('Метод Drag and Drop')
        self.setGeometry(300,300,300,150)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    myapp = Example()
    myapp.show()
    sys.exit(app.exec_())
```

Рисунок 1. Перетаскивание текста из виджета lineEdit на виджет Button.

Запишите основные методы в тетрадь

self.setAcceptDrops(True) - разрешение перетаскивания для виджета

Сначала мы переопределяем метод dragEnterEvent(). Мы сообщаем о типе данных, который мы допускаем.

Виджет QLineEdit имеет встроенную поддержку операций перетаскивания. Все, что необходимо сделать – это вызвать метод setDragEnabled(), чтобы активировать её.

Перетаскивание кнопки

```
import sys
from PyQt5.QtWidgets import QPushButton, QWidget, QApplication
from PyQt5.QtCore import Qt, QMimeData
from PyQt5.QtGui import QDrag
class Button(QPushButton):
    def __init__(self, title, parent):
        super().__init__(title, parent)
    def mouseMoveEvent(self, e): # функция перетаскивания кнопки
        if e.buttons() != Qt.LeftButton: #перетаскивание левой кнопкой мыши
            return
        mimeData = QMimeData()
        drag = QDrag(self) #создается объект,обеспечивает поддержку перетаскивания данных, основанную на MIME-типе.
        drag.setMimeData(mimeData)
        drag.setHotSpot(e.pos() - self.rect().topLeft())
        dropAction = drag.exec_(Qt.MoveAction)#начинается операция перетаскивания
    def mousePressEvent(self, e):# функция нажатия на кнопку мыши
        QPushButton.mousePressEvent(self, e)
        if e.button() == Qt.RightButton:
            print('Нажать')
class Example(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()
    def initUI(self):
        self.setAcceptDrops(True)
        self.button = Button('Кнопка', self)
        self.button.move(100, 65)
        self.setWindowTitle('Нажать или перенести')
        self.setGeometry(300, 300, 280, 150)
    def dragEnterEvent(self, e):
        e.accept()
    def dropEvent(self, e): #определяем, что произойдёт после отпускания кнопки мыши.
        position = e.pos()#определяем текущую позицию
        self.button.move(position)#переносим кнопку в эту позицию
        e.setDropAction(Qt.MoveAction)
        e.accept()
if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = Example()
    ex.show()
    app.exec_()
```

С помощью метода Drag and Drop:

1. Удалить виджеты формы (кнопки, lineEdit) в «корзину»
2. Удаление строк из PlainTextEdit
3. Увеличить размер виджета на число взятое из другого виджета
4. Уменьшить размер виджета на число взятое из другого виджета
5. Создать тест на соответствие
6. Открыть txt файл в plainTextEdit