УДК 004.514

ПРОЕКТИРОВАНИЕ ГЛАВНОГО ОКНА ИНТЕРФЕЙСА ПРОГРАММНОЙ СРЕДЫ OPENFOAM НА БАЗЕ ТЕХНОЛОГИЙ PYQT4 И PYTHON 3.4

[1]Читалов Д.И., [1,2]Калашников С.Т.

[1]Отдел фундаментальных проблем аэрокосмических технологий, ФГБНУ «Южно-Уральский научный центр», Миасс, e-mail: cdi9@yandex.ru;

[2]ОАО «ГРЦ Макеева», Миасс, e-mail: src@makeyev.ru

Настоящая статья описывает устройство главного окна графического интерфейса пользователя для программной среды (ПС) OpenFOAM и пошаговый процесс его разработки. Целью работы является исследование особенностей создания главного окна для работы с консольной программой OpenFOAM и разработка версии окна в виде отдельного приложения с базовыми опциями. Выполнение работы основано на изучении выбранного инструментария: библиотеки PyQt4, которая используется в качестве средства создания компонентов главного окна и элементов управления экранных форм, и высокоуровневого языка программирования Python 3.4, обеспечивающего взаимодействие интерфейса с ПС OpenFOAM. Результатом проведенной работы является отдельное программное приложение, состоящее из исполняемого и вспомогательного файлов, обеспечивающих отображение окна и функционирование его базовых опций.

Ключевые слова: графический интерфейс пользователя, главное окно программы, OpenFOAM, язык программирования Python, библиотека PyQt4, открытое программное обеспечение

DESIGNING THE MAIN WINDOW OF THE OPENFOAM SOFTWARE ENVIRONMENT INTERFACE BASED ON PYQT4 AND PYTHON 3.4 TECHNOLOGIES

[1]Chitalov D.I., [1,2]Kalashnikov S.T.

[1]Department of Fundamental Problems of Aerospace Technologies of Federal State Budget Scientific Institution «South Ural Scientific Center», Miass, e-mail: cdi9@yandex.ru;

[2]State Rocket Center of Makeyev, Miass, e-mail: src@makeyev.ru

This article describes the construction of the main GUI window for OpenFOAM software and its step-by-step development. The aim of the work is to study the features of creating a main window for working with the OpenFOAM console program and developing a version of the window as a separate application with basic options. The work is based on the study of the chosen toolkit: the PyQt4 library, which is used as a means of creating components of the main window and controls for screen forms, and a high-level Python 3.4 programming language that interfaces the interface with the OpenFOAM MS. The result of this work is a separate software application, consisting of executable and auxiliary files, providing the display of the window and the operation of its basic options.

Keywords: graphical user interface, the main program window, OpenFOAM, Python 3.4, the PyQt4 library, open source software

Данная статья является продолжением опубликованной работы [1], посвященной особенностям разработки графического интерфейса для ПС OpenFOAM – открытой совместимой платформы для постановки экспериментов по численному моделированию задач механики сплошных сред, в частности задач, относящихся к области гидродинамики.

В настоящей статье подробно рассматриваются вопросы создания главного окна интерфейса: подготовка его структуры, особенности создания каждого компонента, принципы определения и вызова процедур-обработчиков при выполнении различных действий пользователем и при наступлении событий, связанных с главным окном.

Графический интерфейс любой программы, как правило, имеет главное окно, посредством которого происходит взаимодействие с пользователем. В данном случае главное окно позволяет вести работу с формой импорта файла расчетной сетки для проекта задачи OpenFOAM и с формами редактирования служебных файлов.

Целью работы является исследование возможностей технологий PyQt4 и Python 3.4 в качестве инструментария для разработки главного окна пользовательского интерфейса. Исследования основываются на изучении документации [2, 3] и учебной литературы [4, 5] по данным технологиям.

На рис. 1 отражена структура главного окна интерфейса в рамках текущей разработки. Набор вложенных директорий и конфигурационных файлов проекта задачи предполагается представить в виде дерева (левая прикрепляемая панель) с возможностью открытия формы редактирования файла. При этом формы для загрузки файла расчетной сетки и редактирования каждого конфигурационного файла должны отображаться в центральном компоненте (рабочая область). Верхняя прикрепляемая панель отвечает за вывод названия проекта задачи, а нижняя – выполняет функцию области уведомлений для информирования пользователя о событиях, связанных с работой ПС OpenFOAM. Для управления опциями главного окна используется панель инструментов.

Рис. 1. Структура главного окна интерфейса ПС OpenFOAM: 1 – название проекта задачи, 2 – дерево проекта задачи, 3 – рабочая область, 4 – область уведомлений

Рис. 2. Диаграмма структуры главного окна в стиле ООП: 1 – блок начальных инструкций главного класса, 2 – блок компонентов интерфейса

Описание программного кода вводной части документа

Инструкции вводной части документа отвечают за подключение стандартных библиотек и внешних пользовательских модулей, в частности вспомогательного класса «file_form» из файла «file_form», расположенного в директории «add_classes» и класса «mesh_form» из файла «mesh_form» директории «forms», соответствующего форме импорта расчетной сетки.

Листинг 1. Вводная часть документа.

```
from add_classes.file_form import file_form
from forms.mesh_form import mesh_form
```

Описание блока начальных инструкций главного класса

Программный код листинга 2 в соответствии с диаграммой структуры главного окна в стиле объектно-ориентированного программирования (рис. 2) определяет раздел главного класса, конструктора класса и вызов конструктора базового класса. При этом конструктору класса во второй строке могут быть переданы пользовательские переменные «prj_name», «new_dir», «solver», содержащие название проекта задачи, путь до проекта и название решателя OpenFOAM соответственно.

Листинг 2. Блок начальных инструкций главного класса.

```
class GUIWindow(QtGui.QMainWindow):
  def __init__(self, prj_name, new_dir, solver,
  parent=None):
    QtGui.QMainWindow.__init__(self, parent)
```

Адрес проекта задачи формируется путем «сцепления» содержимого переменных «new_dir», «prj_name» и «solver» (листинг 3), переданных в главное окно из стартового окна [1]. Адрес директории проекта задачи также может быть указан в переменной «full_dir» в виде абсолютного пути, например «/home/Desktop/test», а название проекта задачи (переменная «prj_name») – в виде строки, например «test». Аналогично «вручную» может быть определен решатель «rhoCentralFoam».

Листинг 3. Переменные адреса директории, названия проекта задачи и решателя.

```
self.full_dir = new_dir+"/"+prj_name
self.solver = "rhoCentralFoam"
```

Описание блока компонентов интерфейса

Листинги 4–8 содержат инструкции создания компонентов интерфейса: панели инструментов программы, верхней (название проекта задачи), левой (область дерева проекта), нижней (область уведомлений) и центральной (рабочая область) прикрепляемых панелей.

Листинг 4. Блок инструкций панели инструментов и ее элементов управления.

```
self.GUI_tools = QtGui.QToolBar()
self.mesh_import = QtGui.QAction(self)
self.mesh_img = self.style().standardIcon(QtGui.
QStyle.SP_DirOpenIcon)
self.mesh_import.setIcon(self.mesh_img)
self.mesh_import.setToolTip('Открыть форму импорта сетки')
self.mesh_import.triggered.connect(self.on_
mesh_import)
self.GUI_tools.addAction(self.mesh_import)
self.addToolBar(QtCore.Qt.TopToolBarArea,
self.GUI_tools)
```

Листинг 5. Блок инструкций верхней прикрепляемой панели с названием проекта задачи.

```
self.pnl = QtGui.QDockWidget()
self.pnl.setFeatures(self.pnl.NoDockWidgetFeatures)
self.project_name_lbl = QtGui.QLabel("Название
проекта: " + self.prj_name)
self.project_name_lbl.setStyleSheet("border-
style: none;")
self.pnl.setWidget(self.project_name_lbl)
self.addDockWidget(QtCore.
Qt.TopDockWidgetArea, self.pnl)
```

Листинг 6. Блок инструкций левой прикрепляемой панели с деревом проекта задачи.

```
self.fsw = QtGui.QDockWidget()
self.fsw.setFeatures(self.fsw.NoDockWidget-
Features)
self.fsw_lbl = QtGui.QLabel("Файловая структура проекта")
self.fsw_lbl.setStyleSheet("border-style:
none;" "font-size: 8pt;")
self.fsw_lbl.setAlignment(QtCore.
Qt.AlignCenter)
self.fsw_tab = QtGui.QGridLayout()
self.fsw_tab.addWidget(self.fsw_lbl, 0, 0)
self.fsw_fr = QtGui.QFrame()
self.fsw_fr.setFixedSize(400, 32)
self.fsw_fr.setStyleSheet("background-color:
beige;" "border-width: 1px;" "border-style: solid;"
"border-color: dimgray;" "border-radius: 4px;")
self.fsw_fr.setLayout(self.fsw_tab)
self.fsw.setTitleBarWidget(self.fsw_fr)
self.prj_struc = QtGui.QTreeView()
self.prj_struc.setFixedSize(400, 680)
self.prj_struc.mod = QtGui.QFileSystemModel()
self.prj_struc.mod.setRootPath(self.full_dir)
self.prj_struc.setModel(self.prj_tree.mod)
self.prj_struc.header().hide()
self.prj_struc.setColumnHidden(1, True)
self.prj_struc.setColumnHidden(2, True)
self.prj_struc.setColumnHidden(3, True)
self.prj_struc.setRootIndex(self.prj_struc.
mod.index(self.full_dir))
self.prj_struc.mod.directoryLoaded.
connect(self.fetchAndExpand)
self.prj_struc.setItemsExpandable(False)
```

```
self.prj_struc.clicked.connect(self.on_prj_
struc_clicked)
self.fsw.setWidget(self.prj_struc)
self.addDockWidget(QtCore.
Qt.LeftDockWidgetArea, self.fsw)
```

Для отображения вложенности папок и файлов при открытии главного окна реализована процедура-обработчик «fetchAndExpand» (листинг 10). Она выполняется при наступлении события «directoryLoaded». За открытие форм редактирования конфигурационных файлов проекта задачи отвечает функция «on_prj_struc_clicked» (листинг 11).

Листинг 7. Блок инструкций нижней прикрепляемой панели (области уведомлений).

```
self.serv_mes = QtGui.QDockWidget("Служебные
сообщения")
self.serv_mes.setFixedSize(1000, 170)
self.serv_mes.setFeatures(self.serv_mes.No-
DockWidgetFeatures)
self.listWidget = QtGui.QListWidget()
self.serv_mes.setWidget(self.listWidget)
self.addDockWidget(QtCore.
Qt.BottomDockWidgetArea, self.serv_mes)
```

Следующий программный код (листинг 8) определяет прикрепляемую панель, используемую в качестве рабочей области (центрального компонента). В ней отображается форма импорта файла расчетной сетки, а также форма редактирования конфигурационного файла, выбранного пользователем, и фрейм с заголовком.

Листинг 8. Блок инструкций центральной прикрепляемой панели (рабочей области).

```
self.ffw = QtGui.QDockWidget()
self.ffw.setFeatures(self.ffw.NoDockWidget-
Features)
self.ffw_lbl = QtGui.QLabel()
self.ffw_lbl.setAlignment(QtCore.
Qt.AlignCenter)
self.ffw_tab = QtGui.QGridLayout()
self.ffw_tab.addWidget(self.ffw_lbl, 0, 0)
self.ffw_frame = QtGui.QFrame()
self.ffw_frame.setFixedSize(590, 43)
self.ffw_frame.setStyleSheet("border-width:
1px;" "border-style: solid;" "border-color:
dimgray;" "border-radius: 4px;" "background-
color: beige;")
self.ffw_frame.setLayout(self.ffw_tab)
self.setCentralWidget(self.ffw)
```

Описание программного кода функций главного окна

Первой используемой функцией главного окна является функция открытия в прикрепляемой панели центрального компонента формы импорта расчетной сетки после нажатия пользователем соответствующей кнопки панели инструментов (листинг 4). Программный код этой функции главного окна представлен в листинге 9.

Листинг 9. Функция открытия формы импорта расчетной сетки.

```
def on_mesh_import(self):
  self.ffw.setTitleBarWidget(self.ffw_frame)
  self.ffw_lbl.setText("Форма импорта сетки")
  self.ffw_lbl.setStyleSheet("border-style:
  none;" "font-size: 8pt;")
  self.ffw.setWidget(mesh_form(self))
```

Функция «fetchAndExpand» (листинг 10) выполняется при загрузке модели данных проекта задачи в виджет-дерево и автоматически раскрывает содержимое папок: «0», «system», «constant», «polyMesh».

Листинг 10. Функция раскрытия содержимого проекта задачи.

```
def fetchAndExpand(self, full_dir):
  index = self.prj_struc.mod.index(full_dir)
  self.prj_struc.expand(index)
  for i in range(self.prj_struc.mod.rowCount(index)):
    child = index.child(i, 0)
    if self.prj_struc.mod.isDir(child) and self.prj_
    struc.mod.fileName(child) == "0" or
    self.prj_struc.mod.fileName(child) == "system"
    or  self.prj_struc.mod.fileName(child)==
    «constant» or
    self.prj_struc.mod.fileName(child)==
    "polyMesh":
      self.prj_struc.mod.setRootPath(self.
      prj_struc.mod.filePath(child))
```

Третья функция главного окна (листинг 11) описывает поведение приложения при выборе пользователем определенной «ветви» дерева проекта задачи, т.е. при выборе определенного файла проекта. При этом происходит открытие формы редактирования файла с заголовком, в котором отображается надпись с именем файла.

Листинг 11. Функция выбора пользователем «ветви» дерева проекта задачи.

```
def on_prj_struc_clicked(self, index):
  indexItem = self.prj_struc.mod.index(index.
  row(), 0, index.parent())
  file_name=self.prj_struc.mod.fileName(indexItem)
  file_form.inp_file_form_func(self, file_name)
  file_name_title = file_form.out_file_name_func()
  if file_name_title != None:
    self.ffw.setTitleBarWidget(self.ffw_frame)
    self.ffw_lbl.setText("Форма параметров
    файла: « + "<font color='peru'>" + file_
    name_title +"</font>")
    self.ffw_label.setStyleSheet("border-style:
    none;" "font-size: 8pt;")
  else:
    self.clear_label = QtGui.QLabel()
    self.ffw.setTitleBarWidget(self.clear_label)
  file_form = file_form.out_file_form_func()
  self.ffw.setWidget(file_form)
```

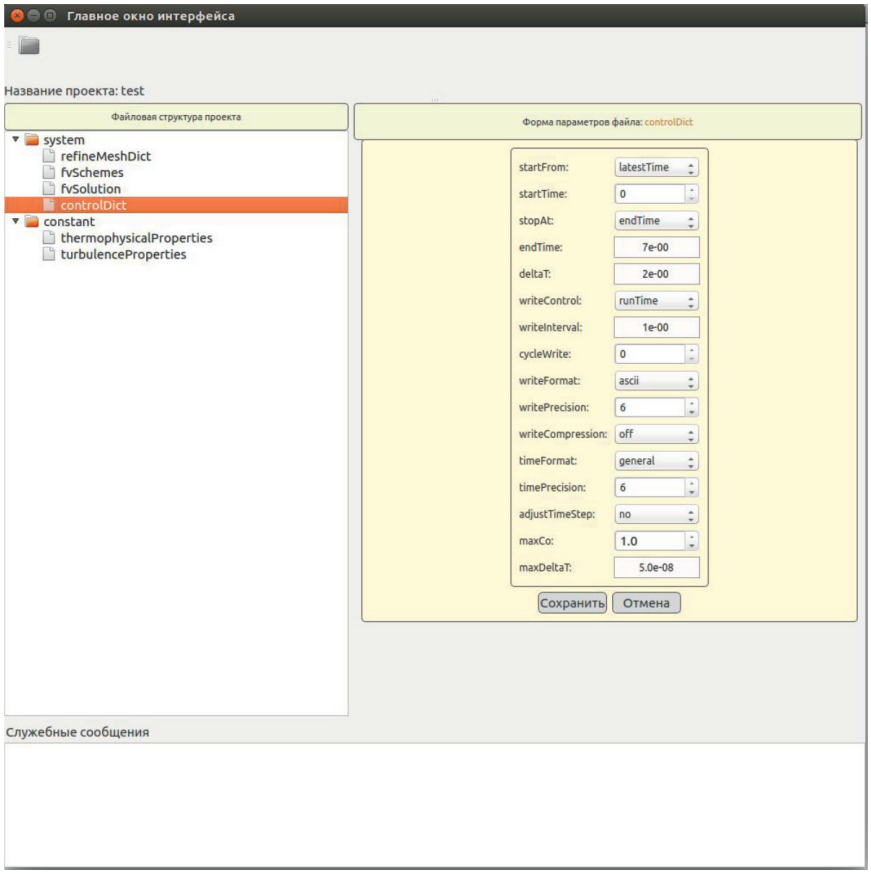

Рис. 3. Результат запуска файла «file_system.py»

В рассматриваемом примере главное окно программы взаимодействует со вспомогательным классом (листинг 12), который при выборе файла из виджета-дерева передает в главное окно название выбранного файла и ссылку на форму его редактирования. Благодаря использованию вспомогательного класса при расширении списка форм, с которыми работает главное окно, изменения вносятся во вспомогательный класс, а не в структуру программного кода главного окна.

Листинг 12. Вспомогательный класс приложения.

```
from forms.controlDict_form import controlDict_
form
class file_form:
  def inp_file_form_func(self, file_name):
    global file_name_gl
    global file_form
    file_name_gl = file_name
    if file_name_gl == "controlDict":
      file_form = controlDict_form(self)
    else:
      file_name_gl = None
      file_form = None
  def out_file_name_func(): return file_name_gl
  def out_file_form_func(): return file_form
```

Тестирование работы приложения

В ходе проведенной работы созданы два файла: «file_system.py», содержащий программный код главного окна, и «file_form.py», содержащий инструкции вспомогательного класса. Для запуска приведенного примера необходимо создать отдельную папку, в которую поместить запускаемый файл «file_system.py» и две директории: «add_classes» и «forms». Запускаемый файл должен содержать программный код листингов 1–11, а блоки кода – расположены в соответствии с диаграммой структуры главного окна в стиле ООП (рис. 2). Кроме того, в запускаемом файле необходимо наличие инструкций, отвечающих за создание объекта приложения и вызов главного класса. Файл «file_form.py» должен содержать код листинга 12. Результат запуска файла «file_system.py» с выбранной ветвью (файлом) «controlDict» представлен на рис. 3.

Заключение

В настоящей статье рассмотрена практическая реализация главного окна ин-

терфейса в виде SDI-приложения для ПС OpenFOAM. Представлен пошаговый процесс создания версии главного окна графической оболочки с базовым набором опций. Материалы, изложенные в настоящей работе, могут быть применены пользователями ПС OpenFOAM для создания собственных интерфейсов, а также разработчиками интерфейсов для других консольных (и не только) программ. Данную статью можно использовать в качестве учебного пособия для изучения основ программирования на базе высокоуровневых языков.

Список литературы

1. Читалов Д.И., Меркулов Е.С., Калашников С.Т. Разработка графического интерфейса пользователя для программного комплекса OpenFOAM // Программная инженерия. – 2016. – вып. 12. – С. 568–574.

2. PyQt4 Reference Guide [Электронный ресурс]. – URL: http://pyqt.sourceforge.net/Docs/PyQt4 (дата обращения: 05.08.2017).

3. Введение в среду PyQt4 [Электронный ресурс]. – URL: http://wiki.python.su/Документации/ВведениеВСредуPyQt4 (дата обращения: 05.08.2017).

4. Прохоренок Н.А., Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012. – 704 с.

5. Хахаев И.А. Практикум по алгоритмизации и программированию на Python. – М.: Альт Линукс, 2010. – 126 с.