УДК 004.94:004.514:550.822.3

ТЕХНОЛОГИЯ СОЗДАНИЯ МУЛЬТИЗАДАЧНОЙ ГРАФИЧЕСКОЙ ОБОЛОЧКИ СИСТЕМЫ ВИЗУАЛИЗАЦИИ ЦИФРОВОЙ МОДЕЛИ КЕРНА

П. Ю. Тимохин, М. В. Михайлюк
Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук, webpismo@yahoo.de, mix@niisi.ras.ru

В статье описана технология разработки мультизадачной графической оболочки системы визуализации цифровой модели керна, основанная на создании подсистем виджетов с использованием открытой кроссплатформенной библиотеки Qt. Предложен метод разделения графического интерфейса системы визуализации между исполнительным ядром и модулями задач, а также модель подключения модулей задач к ядру. На основе предложенных технологий, методов и алгоритмов создан программный комплекс, реализующий графическую оболочку системы визуализации цифровой модели керна. Созданный комплекс успешно апробирован при исследовании цифровых моделей кернов баженовской свиты и песчаника и показал адекватность разработанного решения поставленной задаче. Разработанное в ФГУ ФНЦ НИИСИ РАН решение направлено на развитие технологии «цифрового месторождения» и может быть использовано в исследованиях специалистов нефтегазовой отрасли, виртуальных лабораториях и образовательных приложениях.

Ключевые слова: система визуализации, керн, цифровая модель, графическая оболочка, виджет.

IMPLEMENTATION TECHNOLOGY OF MULTITASKING GRAPHICAL SHELL OF VISUALIZATION SYSTEM FOR DIGITAL MODEL OF THE CORE MATERIAL

P. Yu. Timokhin, M. V. Mikhaylyuk
System Research Institute, Russian Academy of Sciences
webpismo@yahoo.de, mix@niisi.ras.ru

The paper proposes a technology for implementation of a multitasking graphical shell of a visualization system for digital model of the core material, based on development of widget subsystems using the open source cross-platform Qt library. The study provides a method to split the graphical interface of the visualization system between executive kernel and task modules, as well as a model to connect task modules to the kernel. Based on the developed technology, methods and algorithms a program complex implementing the graphical shell of the visualization system for digital model of the core material is created. The created complex is successfully tested through studies of digital models of the core materials of the Bazhenov Formation and Sandstone and showed adequacy of the proposed solution to the problem. The solution, developed in the System Research Institute, Russian Academy of Sciences, is aimed at the development of the “digital oilfield” technology and can be used in researches by specialists from oil and gas industry, virtual laboratories and educational applications.

Keywords: visualization system, core material, digital model, graphical shell, widget.

Стратегически важным направлением отечественного нефтегазового инжиниринга является создание новой технологии добычи трудноизвлекаемых запасов углеводородов (целики нефти, низкопроницаемые пласты, баженовская свита), основанной на использовании постоянно обновляемой суперкомпьютерной геолого-гидродинамической модели месторождения («цифровое месторождение») [1]. Одной из важных составляющих «цифрового месторожде-

ния» является система визуализации цифровой модели керна [2] – столбца породы, извлеченного на месте нефтяного месторождения. Она включает в себя методы построения и визуализации трехмерной модели порового пространства керна, двухмерных срезов и кросс-сечений модели керна, визуализации неустойчивого вытеснения нефти из пористых сред и др. [3–5]. Данные методы имеют отличные друг от друга входные параметры и форматы представления графической информации, в связи с чем возникает задача создания эргономичной графической оболочки системы визуализации цифровой модели керна.

Эффективным подходом является создание графической оболочки, состоящей из ядра, реализующего базовые функции системы визуализации, и динамически подключаемых к ядру программных модулей, расширяющих его функциональность (плагинов) [6]. В настоящее время существует ряд распространенных парадигм построения графических интерфейсов программных систем (Model-View-Controller, Hierarchical MVC, Presentation-Abstraction-Control) [7], однако разработка технологий практической реализации таких парадигм, в частности, для систем типа «ядро – плагин» является актуальным предметом для исследований. В системе ParaView [8] визуализации научных данных ядро графической оболочки содержит элементы управления для множества универсальных методов анализа и визуализации, из которых исследователь выбирает необходимые для решения конкретной прикладной задачи. Эта система является открытой и имеет возможность добавления плагинов, реализующих дополнительные методы обработки данных и элементы управления. Система Amira-Avizo [9] визуализации и анализа научных и промышленных данных является мультизадачной, и расширение ее базовых функций осуществляется с помощью модулей, решающих определенные задачи, например, модуль XMolecular визуализации молекулярных структур. Данная система является коммерческой, и ее реализация закрыта.

В работе предложена технология создания мультизадачной графической оболочки, в которой подключаемые к ядру модули решают определенные задачи по визуализации цифровой модели керна в рамках единой системы визуализации. Предлагаемая технология является кроссплатформенной и основана на использовании современного открытого комплекса Qt средств разработки графических интерфейсов [10].

Создание графической оболочки системы визуализации

В предлагаемой технологии система визуализации цифровой модели керна имеет модульную структуру и включает в себя исполняемое *ядро* и ряд динамически связываемых с ним библиотечных *модулей задач* (рис. 1). В ядре содержится главное окно и некоторый базовый функционал системы визуализации, а в каждом модуле задачи – программная реализация отдельного метода визуализации цифровой модели керна, включающего блок графических расчетов и набор необходимых *элементов графического интерфейса* (ЭГИ). Совокупность наборов ЭГИ и главного окна образуют графическую оболочку системы визуализации (на рис. 1 выделена серым цветом).

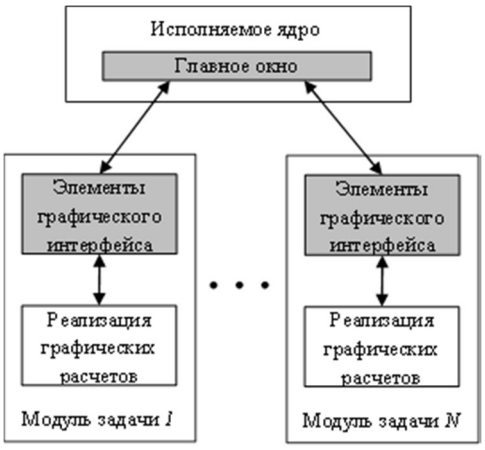


Рис. 1. Графическая оболочка в системе визуализации

Суть предлагаемой технологии состоит в том, что графическая оболочка динамически трансформируется под выбор исследователя: набор ЭГИ метода визуализации встраивается в главное окно (становится видимым), если этот метод выбран исследователем в меню главного окна, а в остальных случаях набор ЭГИ является исключенным из главного окна и скрыт от исследователя.

Создание такой графической оболочки выполняется на основе виджетов библиотеки Qt. Виджет – это самодостаточный визуальный элемент интерфейса пользователя, способный посылать *сигналы*, содержащие информацию о событии (например, о выборе пункта меню, изменении значения текстового поля и т. п.), и формировать ответную реакцию на сигналы от других виджетов с помощью специальных функций – *слотов* [10]. Предлагаемая технология создания графической оболочки основана на построении подсистем виджетов (виджетов, взаимосвязанных сигналами и слотами) и включает в себя:

- создание подсистемы виджетов главного окна;
- создание подсистемы виджетов модуля задачи;
- подключение модулей задач к исполняемому ядру.

Рассмотрим эти составляющие:

1. Подсистема виджетов главного окна графической оболочки создается на основе виджета QMainWindow. Он включает в себя ряд виджетов с фиксированным расположением (рис. 2): контейнер панели меню (QMenuBar); центральный виджет (QWidget); строку состояния (QStatusBar), а также области пристыковки свободно перемещаемых виджетов типа «панель инструментов» и «плавающее окно». При переносе свободно перемещаемого виджета в область пристыковки происходит его автоматическое встраивание между ближайшей стороной главного окна и центральным виджетом.

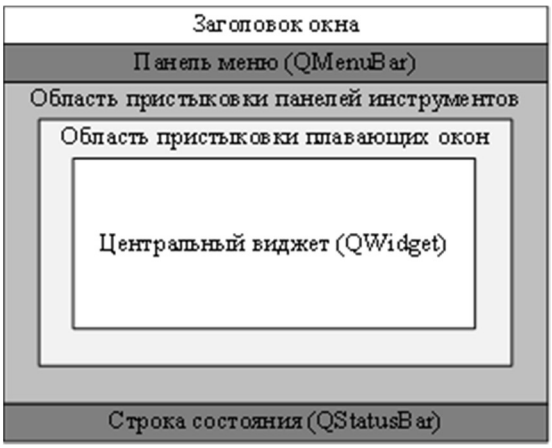


Рис. 2. Структура главного окна графической оболочки

Подсистема виджетов главного окна создается в два этапа. На первом этапе создается ее макет с помощью визуального редактора Qt Designer, входящего в пакет установки библиотеки Qt. В макете в меню главного окна добавляется раздел «Задача», предназначенный для вызова методов визуализации цифровой модели керна, и раздел со справкой по системе визуализации. Созданный макет сохраняется в XML-формате и добавляется в файл исполняемого ядра системы визуализации. На втором этапе в исполняемом ядре с помощью функции *setupUi* библиотеки Qt [10] из сохраненного XML-макета создается непосредственно сама подсистема виджетов главного окна.

2. Подсистема виджетов модуля задачи. Создание подсистемы виджетов модуля задачи включает в себя:

1) создание внешних виджетов – крупных виджетов-контейнеров, которые будут подключаться непосредственно к главному окну графической оболочки. В данной работе используются следующие основные внешние виджеты: контейнер панели меню (QMenuBar); контейнеры панелей инструментов (QToolBar); окно визуализации (QOpenGLWidget); контейнеры плавающих окон (QDockWidget);

2) создание внутренних виджетов – виджетов, расположенных внутри внешних виджетов, с помощью которых исследователь непосредственно взаимодействует с методом визуализации, например, полей ввода целых и действительных чисел (QSpinBox и QDoubleSpinBox), пунктов меню (QAction), графиков (QChartView) и др.;

3) связывание внутренних виджетов, на которые воздействует пользователь (пункты меню, кнопки панелей инструментов и т. д.), с блоком графических расчетов модуля задачи (см. рис. 1).

Подсистема виджетов модуля задачи так же, как и подсистема виджетов главного окна (см. п. 1), создается в два этапа.

На первом этапе создается макет подсистемы виджетов с помощью Qt Designer. Основой макета является базовый виджет QMainWindow, на котором размещаются простые внешние виджеты (контейнеры панели меню, панелей инструментов, плавающих окон). Созданный макет добавляется в файл модуля задачи.

На втором этапе в модуле задачи на основе макета создаются внешние и внутренние виджеты, а также выполняется связывание внутренних виджетов с блоком графических расчетов модуля задачи. Внешние виджеты, входящие непосредственно в макет, создаются с помощью функции *setupUi*, а остальные виджеты создаются и размещаются с помощью базовых классов Qt (более подробно они описаны в [10]). Связывание внутренних виджетов с блоком графических расчетов осуществляется с помощью упомянутого выше механизма сигналов и слотов и включает в себя:

- выбор стандартного сигнала внутреннего виджета, описывающего осуществляемое пользователем воздействие. Примером является сигнал *triggered* виджета QAction, который возникает, когда пользователь нажимает пункт меню, кнопку панели инструментов или комбинацию горячих клавиш;
- создание слота, реализующего ответную реакцию модуля задачи на выбранный сигнал внутреннего виджета, например, чтение значений параметров из панели инструментов и передача их в функцию визуализации. Слот создается в основном виджете QMainWindow модуля задачи с помощью квалификатора «*slots:*»;
- связывание выбранного сигнала и созданной функции-слота с помощью Qt-функции *connect* (W_{SRC}, *signalFunc*, W_{DST}, *slotFunc*), где W_{SRC} – виджет-источник сигнала *signalFunc*, а W_{DST} – виджет-приемник с функцией-слотом *slotFunc*.

Отметим, что для реализации нестандартного поведения виджета (например, цветовой индикации введенных значений параметров, выходящих за границы допустимого диапазона) создается производный класс от базового класса виджета, в который добавляются собственные сигналы с помощью квалификатора «*signals:*», а генерация этих сигналов осуществляется с помощью квалификатора «*emit*» [10].

3. Подключение модулей задач к исполняемому ядру. В предлагаемой технологии подключение модулей задач к исполняемому ядру осуществляется динамически на стадии загрузки системы визуализации. Предлагаемая модель подключения модулей задач основана на отложенной загрузке динамических библиотек и включает в себя следующие этапы:

1) создание в каждом модуле задачи набора F интерфейсных функций:
- *create()* – создает подсистему виджетов и блок графических расчетов;
- *destroy()* – удаляет подсистему виджетов и блок графических расчетов;
- *getTaskName()* – возвращает короткое название метода визуализации;
- *runGui()* – встраивает подсистему виджетов в главное окно исполняемого ядра;
- *closeGui()* – исключает подсистему виджетов из главного окна исполняемого ядра;

2) создание в главном окне исполняемого ядра следующих функций-слотов:

- *runTask()* – запускает метод визуализации (модуль задачи), выбранный пользователем;
- *closeCurrentTask()* – закрывает текущий метод визуализации (модуль задачи);

3) формирование меню доступных методов визуализации в главном окне графической оболочки.

Рассмотрим эти этапы.

Создание набора *F* интерфейсных функций. Из данного этапа рассмотрим более подробно работу функций *runGui()* и *closeGui()* на примере модуля задачи, содержащего внешние виджеты, описанные в п. 2.

Функция *runGui()* включает в себя выполнение следующих шагов:

1. Вставка разделов меню модуля задачи (вместе с пунктами) в панель меню главного окна (после раздела «Задача») с помощью функции *insertMenu* виджета QMenuBar.

2. Добавление панелей инструментов модуля задачи в главное окно с помощью функции *addToolBar* виджета QMainWindow.

3. Добавление плавающих окон модуля задач в главное окно с помощью функции *addDockWidget* виджета QMainWindow.

4. Добавление окна визуализации модуля задачи в главное окно. Для этого вначале создается компоновочный слой и к нему прикрепляется окно визуализации модуля задачи с помощью функции *addWidget* виджета QBoxLayout. Полученный слой прикрепляется к центральному виджету главного окна с помощью функции *setLayout* виджета QWidget. Использование промежуточного слоя позволяет избежать относительно медленного пересоздания окон визуализации при переключении между различными методами визуализации в графической оболочке.

Задача функции *closeGui()* состоит в приведении вида главного окна к состоянию до подключения модуля задачи. Это реализуют следующие шаги:

1. Отсоединение окна визуализации модуля задачи от компоновочного слоя с помощью функции *removeWidget* виджета QBoxLayout (сам слой удаляется автоматически).

2. Отсоединение плавающих окон модуля задачи от главного окна с помощью функции *removeDockWidget* виджета QMainWindow.

3. Отсоединение панелей инструментов модуля задачи от главного окна с помощью функции *removeToolBar* виджета QMainWindow.

4. Извлечение разделов меню модуля задачи из меню главного окна с помощью функции *removeAction* виджета QMenuBar.

Создание функций-слотов в исполняемом ядре. Функции-слоты *runTask()* и *closeCurrentTask()* выполняют вызовы функций *runGui()* и *closeGui()* модуля задачи, выбранного исследователем в разделе «Задача» меню главного окна. Пункт меню (модуль задачи), который выбрал пользователь, определяется с помощью функции *sender* базового объекта QObject библиотеки Qt.

Формирование меню доступных методов визуализации реализуется на стадии загрузки системы визуализации и выполняется за два шага.

На первом шаге создается список *taskList* имен модулей задач, доступных системе визуализации. Такими являются динамические библиотеки с именами формата «*xModule*» (*x* – акроним метода визуализации), помещенные в специальную директорию «*Tasks*» на стадии установки системы визуализации на компьютер исследователя.

На втором шаге для каждого модуля задачи из списка *taskList* выполняется следующий алгоритм подключения модуля задачи:

1. Загрузка модуля задачи в адресное пространство процесса системы визуализации с помощью функции *load()* класса QLibrary.

2. Получение адреса интерфейсных функций модуля задачи с помощью функции *resolve(funcName)* класса QLibrary, где *funcName* – имя функции из набора *F*.

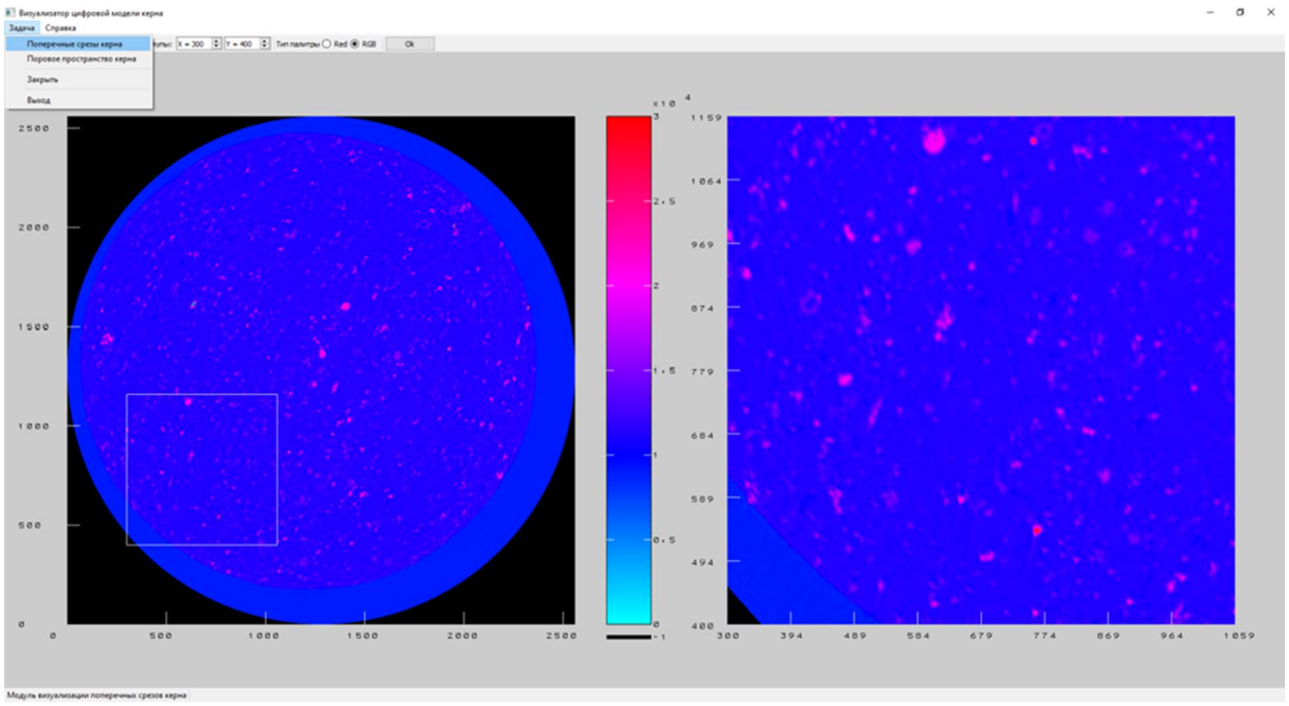
3. Создание подсистемы виджетов и блока графических расчетов модуля задачи с помощью функции *create()* из набора *F*.

4. Получение из модуля задачи короткого имени «*shortName*» метода визуализации с помощью функции *getTaskName()* из набора *F*.

5. Добавление в раздел меню «Задача» главного окна пункта «*shortName*» с помощью функции *addAction()* библиотеки Qt.

6. Связывание сигнала *triggered* добавленного пункта меню со слотом *runTask()* модуля задачи с помощью функции *connect* (см. п. 2). При этом виджетом-источником является пункт меню (QAction), а виджетом-приемником – основной виджет QMainWindow главного окна.

Отметим, что при выходе из системы визуализации для каждого модуля задачи из списка *taskList* выполняется освобождение памяти, выделенной для подсистемы виджетов и блока графических расчетов, с помощью функции *destroy()* из набора *F*.



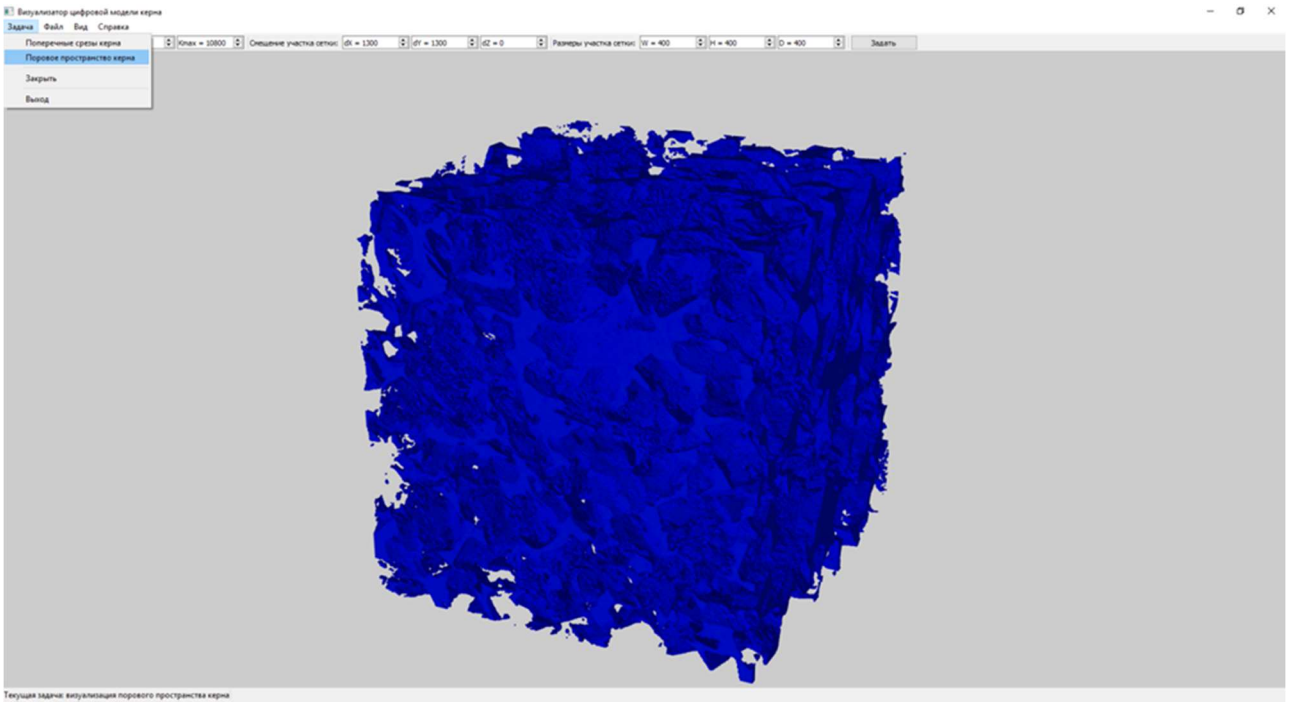


Рис. 3. Вид графической оболочки при выбранных методах визуализации поперечных сечений керна (сверху) и порового пространства керна (внизу)

На основе предложенных технологии, методов и алгоритмов в ФГУ ФНЦ НИИСИ РАН был разработан программный комплекс, реализующий мультизадачную графическую оболочку системы визуализации цифровой модели керна. Были созданы подсистемы виджетов исполнительного ядра и модулей задач визуализации поперечных сечений, кросс-сечений и порового пространства модели керна. Данные модули задач были подключены к исполняемому ядру системы визуализации цифровой модели керна. С помощью разработанного программного комплекса было проведено исследование сечений и порового пространства цифровых моделей образцов кернов баженовской свиты и песчаника, полученных с помощью рентгеновской компьютерной томографии. Исследованные образцы имели размеры 2 560 × 2 560 × 8 600 вокселов (размер воксела 0,0015276 мм) и 2 600 × 2 600 × 7 360 вокселов (размер воксела 0,0029896 мм) соответственно. На рис. 3 показан внешний вид разработанной графической оболочки при выборе исследователем различных методов визуализации.

Заключение. В работе предложена технология создания мультизадачной графической оболочки системы визуализации цифровой модели керна, основанная на создании подсистем виджетов с использованием открытой кроссплатформенной библиотеки Qt. Технология позволяет эффективно объединять в единой системе различные задачи визуализации цифровой модели керна, имеющие отличные друг от друга входные параметры и выходные форматы представления графической информации. Кроме того, предложен метод разделения графического интерфейса системы визуализации между исполнительным ядром и модулями задач, а также модель подключения модулей задач к ядру, которые позволяют добавлять новые методы исследования модели керна в систему визуализации без изменения программного кода ее ядра, а также эффективно изменять набор существующих методов. На основе предложенных технологии, методов и алгоритмов была создана альфа-версия программного комплекса, реализующего графическую оболочку в системе визуализации цифровой модели керна. Созданный комплекс был успешно апробирован при проведении исследований цифровых моделей кернов баженовской свиты и песчаника и показал адекватность разработанного решения поставленной задаче.

Предложенное решение позволяет создавать эффективные программные пакеты, ориентированные как на определенный круг задач по исследованию цифровой модели керна, так и на проведение всестороннего исследования цифровой модели керна с компьютерным моделированием нефтегазовых процессов. Полученные результаты направлены на развитие технологии «цифрового месторождения» и могут быть использованы в исследованиях специалистов нефтегазовой отрасли, виртуальных лабораториях и образовательных приложениях.

Работа выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) «34.9. Системы виртуального окружения: технологии, методы и алгоритмы математического моделирования и визуализации» (0065-2018-0020).

Литература

1. Бетелин В. Б. Цифровое месторождение – путь к трудноизвлекаемым запасам углеводородов // Инновации. 2014. № 1. С. 37–38.

2. Бетелин В. Б. , Никитин В. Ф., Смирнов Н. Н., Михальченко Е. В., Скрылева Е. И., Стамов Л. И., Тюренкова В. В. Компьютерный керносимулятор: подходы и методы // Вестник кибернетики. 2015. № 4. С. 33–44.

3. Михайлюк М. В. , Тимохин П. Ю., Мальцев А. В., Никитин В. Ф., Скрылева Е. И., Тюренкова В. В. Моделирование и визуализация процесса вытеснения нефти из пористой

среды // Вестник кибернетики. 2016. № 3. С. 35–41.

4. Михайлюк М. В. , Мальцев А. В., Торгашев М. А. Моделирование и визуализация порового пространства керна // Тр. НИИСИ РАН. 2017. Т. 7. № 4. С. 78–82.

5. Михайлюк М. В. , Кононов Д. А., Логинов Д. М. Визуализация порового пространства в цифровой модели керна // ИТ-Стандарт : электрон. науч. журн. 2017. № 3. С. 7–10. URL: http://journal.tc22.ru/ (дата обращения: 02.07.2018).

6. Pandolfi R. J., Allan D. B., Arenholz E. et. al. Xi-cam: a versatile interface for data visualization and analysis // Journal of Synchrotron Radiation. 2018. V. 25. № 4. P. 1–10.

7. Karagkasidis A. Developing GUI Applications: Architectural Patterns Revisited / EuroPLoP-2008 : 13th Annual European Conference on Pattern Languages of Programming. Irsee, Germany. July 9–13, 2008. URL: http://ceur-ws.org/Vol-610/paper11.pdf (дата обращения: 02.07.2018).

8. ParaView. URL: https://www.paraview.org/Wiki/ParaView (дата обращения: 02.07.2018).

9. Amira-Avizo Software. URL: https://www.fei.com/software/avizo/ (дата обращения: 02.07.2018).

10. Qt Documentation. URL: http://doc.qt.io/ (дата обращения: 02.07.2018).