# Сцены визуализации в вычислительных блокнотах

П. А. Васёв[1], В. Л. Авербух[1,2], М. О. Бахтерев[1,2],
Д. В. Манаков[1], Д. Ю. Филоненко[3], М. А. Форгани[2]
vasev@imm.uran.ru
[1]Институт математики и механики им. Н.Н. Красовского УрО РАН, г. Екатеринбург
[2] Уральский федеральный университет им. Б.Н. Ельцина, г. Екатеринбург
[3] Уральский государственный архитектурно-художественный университет, г. Екатеринбург

*В работе описан феномен вычислительных блокнотов и современное состояние визуализации в них. Проведен анализ сложившегося состояния в контексте визуального программирования. Представлена реализация визуального программирования сцен визуализации для блокнотов.*

***Ключевые слова:*** *визуализация, вычислительные блокноты, визуальное программирование.*

## 1. Введение

К настоящему времени стал популярным способ численного моделирования с помощью вычислительных блокнотов. Пользователь через веб-браузер вводит команды и программы, а блокнот их интерпретирует, производит вычисления, и показывает результаты.

Однако современные блокноты имеют не всегда удобные средства визуализации данных. По нашему мнению, это связано с тем, что описание видов отображения (сцен визуализации) проводится на том же языке, что и вычисления. Однако языки вычислений не приспособлены для такой задачи, что влечет громоздкость описаний и лишние затраты сил.

Мы предлагаем формировать описание визуализации с помощью графического интерфейса с небольшим набором основных действий: добавить визуализацию, выбрать способ отображения, указать необходимые переменные блокнота — источники данных для построения графического объекта.

Предполагаем, что этот способ поможет инженеру-ученому легче и быстрее конструировать визуальное представление промежуточных и конечных результатов счёта.

## 2. Вычислительные блокноты

Вычислительный блокнот (computational notebook, [2]) – это человеко-машинный интерфейс, позволяющий создавать программы из последовательности т. н. клеток. В клетках может содержаться текст, программный код, другие данные. Текст отображается как есть, а программный код — выполняется, и результат его работы выводится на экран блокнота рядом с исходной клеткой.

Пример блокнота показан на рис 1. Заголовки, текст и формулы — это клетка с текстом. Клетки «in[2]» и «in[5]» — с программным кодом. Изображения — результат выполнения кода клетки «in[5]».

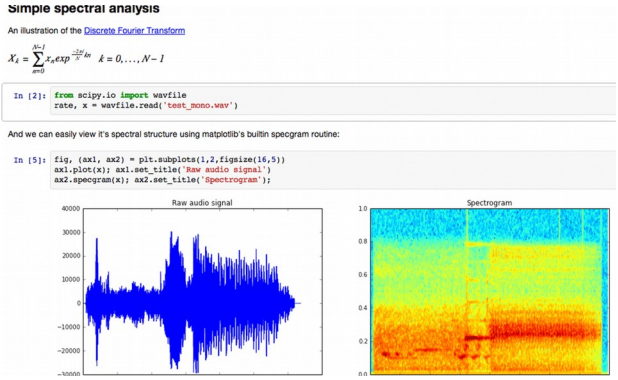


**Рис. 1.** Вычислительный блокнот.

Важной особенностью блокнотов является то, что обычный текст и программный код поставлены в равнозначные позиции. Технически и то и другое является входом для интерпретации; текст интерпретируется как данные в формате markdown, а код — интерпретатором программ.

Происхождение вычислительных блокнотов авторы связывают со следующими идеями программирования:

- Цикл «чтение-вычисление-вывод» (REPL). Эта идея развивается блокнотами: очередная команда пользователя не просто выполняется, а сохраняется в отдельной клетке, которую можно в последствии редактировать и выполнять повторно;
- Грамотное программирование (literate programming). Идея развивается блокнотами до состояния, когда и текст, и программный код считаются равноправными участниками программы.

Широко известные программы, использующие идею вычислительных блокнотов: Mathematica, Maple, Mathcad.

В связи с развитием веб-технологий появились новые блокноты, например:

- Observable Notebooks
- Apache Zeppelin
- Apache Spark Notebook
- JupyterLab (Jupyter Notebook, IPython)
- Nteract
- R Notebooks
- Google Colab

Одни блокноты поддерживают простой режим вычисления клеток — сверху вниз.

Другие блокноты поддерживают «реактивность»: определяют зависимости и автоматически пересчитывают клетки, если данные, от которых клетки зависят, изменились. Это может касаться как данных, рассчитанных в других клетках, так и данных из внешних источников.

Некоторые блокноты поддерживают потоковый режим поступления данных. Например, Spark Notebook умеет обрабатывать потоки из внешних источников (см. видеозапись на странице [3]). Также Spark и Observable умеют порождать в клетках объекты, являющиеся источниками потоков (их термины - streams, generators).

Открытый вопрос: могут ли блокноты и сами являться источником потоковых данных, при подключении их в другие блокноты в качестве модулей?

## 3. Визуализация в блокнотах

Блокноты предлагают различные варианты визуализации данных. Во-первых, они имеют встроенные средства визуализации. Например, они могут представить результат выполнения кода клетки в виде таблицы,


Работа опубликована при финансовой поддержке РФФИ, грант 18-07-20045\18

изображения, графика или диаграммы. Эти средства просты и достаточны для широкого класса задач.

Кроме того, многие блокноты предлагают возможность внедрения дополнительных средств визуализации.

Анализ таких средств показал, что они разделяются на два вида:

1. Специализированные. Обеспечивают визуализацию заданного типа объектов с помощью определённого заранее вида отображения.

2. Универсальные. Предоставляют пользователю возможность самому сформировать вид отображения его данных.

Наблюдения за блокнотами показали, что обычный режим использования универсальных средств следующий.

Пользователь подключает специальную библиотеку визуализации (например matplotlib или plotly). Вводит программный код в клетку, который через программный интерфейс библиотеки визуализации формирует вид отображения.

В результате выполнения кода этой клетки формируется статичный или интерактивный графический вывод.

На рис. 2 показан пример работы в таком режиме.

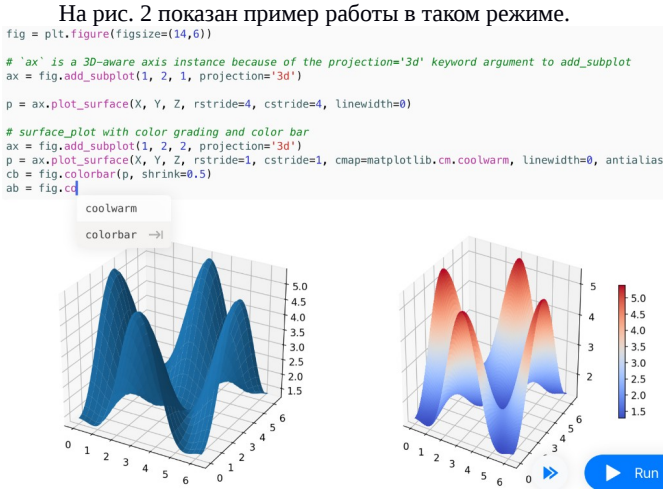


**Рис. 2.** Код визуализации и результат его работы.

## 4. Идея внедрения визуального языка

Идея состоит в том, чтобы предоставить пользователю возможность настраивать получаемую визуализацию в блокнотах с помощью визуальных средств. Например, с помощью мышки менять тип графиков, добавлять новые представления (возможно не только графики), настраивать их цвета и другие атрибуты.

То есть вместо того, чтобы писать программные коды, формирующие вид отображения – задавать его визуально.

Это могло бы облегчить работу пользователя. Ведь для написания кодов на программном языке необходимо переключать контекст мышления, так как программные интерфейсы требуют удержания их в памяти пользователя.

В свою очередь визуальные интерфейсы по своей природе построены на системе подсказок. С их помощью снижаются ментальные затраты пользователя.

Применение подобных идей в блокнотах уже наблюдается, но в ограниченном виде. Например, с помощью визуальных команд можно настроить некоторые атрибуты графиков в блокнотах Zeppelin.

Однако к настоящему времени мы не обнаружили ни одного полноценного примера визуального языка для создания видов отображения в блокнотах.

Мы предполагаем, что скорее всего такие попытки были, но они по каким-то причинам не прижились. Далее мы попытаемся выяснить эти причины.

## 5. О визуальных языках в блокнотах

Классически системы визуализации разделяются на специализированные и универсальные [1].

Специализированные системы предназначены для решения конкретной известной задачи визуализации. При этом они учитывают специфику не только по объектам визуализации, но и по использованию методик компьютерной графики, по научным направлениям.

Универсальные системы визуализации это среды, в которых пользователю предоставлена возможность создать требуемый ему вид отображения самостоятельно.

Можно считать, что это среды программирования, в которых пользователь описывает (программирует), какой вид отображения в данный момент требуется.

Наше обычное понимание универсальных систем визуализации строится из того, что они предлагают визуальный язык программирования. Он может быть простой, например построенный на кнопках, меню и т. д., или более развитый, например, позволяющий создавать графы потоков данных с помощью мышки.

Однако формально система визуализации не обязана предоставлять только визуальный язык. Она может обеспечивать возможность общения и через программный интерфейс. Это хорошо видно на примере макросов в универсальных системах визуализации (например в Excel или Paraview). Более того, эти системы умеют не только потреблять, но и генерировать макросы - исходя из визуальных команд, подаваемых пользователем.

Проведённый авторами анализ показал, что для получения визуализации в блокнотах в подавляющем большинстве случаев используется именно программный язык, а не визуальный графический интерфейс.

Визуальные методы, как принято считать, обеспечивают пользователю интуитивно-понятную возможность формирования необходимых описаний.

Почему же при формировании видов отображения в блокнотах пользователи предпочитают программный язык взамен визуального? И это при условии, что в программных описаниях, во всех случаях, которые нам удалось найти, мы обнаружили большую громоздкость кода. Рис. 2 иллюстрирует это — потрачено 7 строк кода для получения двух графиков. В среде Интернет мы наблюдали примеры и в 20, и в 100 строк кода для получения достаточно тривиальных визуальных сцен.

Возможные причины, почему пользователи предпочитают программные коды в случае блокнотов:

- программирование визуализации ведется на том же языке, что и вычисления. В этом случае пользователю надо преодолеть только один когнитивный барьер — освоить модель библиотеки визуализации. Мы считаем, что это серьезный барьер, так как он несёт дополнительную деталь, которую требуется удерживать в памяти пользователя помимо самих вычислений в прикладной области. Но практика показывает, что пользователей это не отталкивает. Также их не отталкивает, что язык вычислений может не подходить для описания сцен и нести излишние детали.

- возможность переносить программные коды между разными блокнотами. Пользователь формирует программный код, затрачивая на это массу усилий, но в последующей работе он тратит гораздо меньше сил, копируя и адаптируя предыдущие коды.

- возможность комментирования, встроенная в языки программирования - снижает когнитивные затраты при переносе и последующем восприятии кодов визуализации.

- изобилие примеров кодов в среде Интернет, которые дают пользователю возможность скопировать и адаптировать этот код под свои потребности.


The publication of the conference proceedings was supported by the RFBR, grant No. 18-07-20045\18

В результате у пользователя формируется представление, что специализированная визуализация в блокноте с помощью программных кодов – это хоть и затратная, но доступная и предсказуемая процедура.

## 6. О совместимости визуальных языков с концепцией блокнотов

Возможно также, что препятствие на пути использования визуальных языков программирования для получения визуализации в блокнотах лежит в плоскости самих блокнотов. А именно, в концепции клетки как входного объекта для вычислителя, в результате вычисления которого формируется выходной объект.

В этой концепции логично создавать описание вида отображения именно в клетке, а в выводе клетки наблюдать результат работы этого описания.

Получается, например, что настройка вида отображения на визуальном языке в самой выходной области клетки противоречит этой концепции. Вместе с тем многие визуальные языки построены на парадигме непосредственного взаимодействия с объектами [4].

Среди возможных решений этого противоречия мы предполагаем следующие:

1. Смена концепции вычислительных блокнотов на что-то принципиально новое.
2. Сохранение описания, визуально формируемого пользователем, в коде клетки (по аналогии с формированием макросов в системах визуализации).
3. Сохранение этого описания в мета-информации выхода клетки (поддерживается только в Zeppelin). Это может быть полезным и удобным для пользователя, хоть и является необычным с точки зрения концепции блокнотов.
4. Внедрение визуального языка для клеток, когда код клетки формируется не текстом, а визуально.
5. Визуально-текстовый язык, когда код клетки можно и представлять, и редактировать как в виде текста, так и визуально.
6. Создание типа клетки «визуализация» (в дополнение к используемым сейчас типам «код» и «текст»).

Возможно, существуют другие варианты или точки зрения; авторы будут рады их услышать.

## 7. Тестовая реализация

Для воплощения идеи визуального программирования сцен визуализации в блокнотах авторами разработан модуль для Jupyter Notebook.

Пример использования модуля показан на рис. 3:

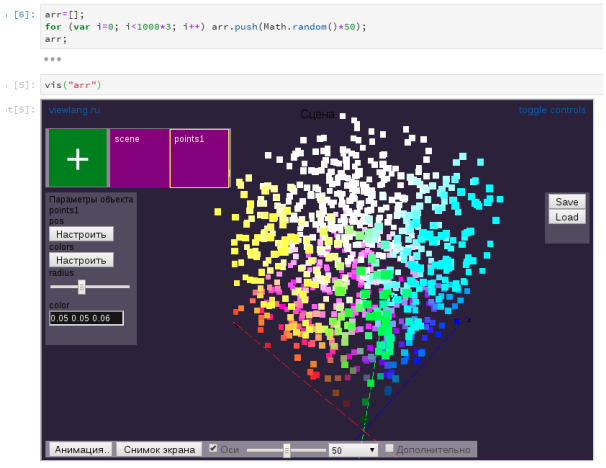


**Рис. 3.** Пример визуализации в блокноте, созданной с помощью визуального языка.

В этом примере пользователь рассчитал массив данных и желает визуализировать его в виде облака точек (мы посчитали это достаточно типовой задачей визуализации).

Для получения визуализации пользователь в блокноте написал вызов специальной функции. В результате её работы появилась специальная область для визуализации.

В этой области пользователь с помощью мышки добавил визуальный объект - «точки», и настроил связь между их положением в пространстве и цветом с рассчитанным в блокноте массивом. Таким образом, пользователь визуально создал требуемый ему вид отображения.

В текущей реализации пользователь может указывать не один, а несколько объектов данных для передачи системе визуализации. Также он может создавать разные визуальные объекты — точки, сферы, отрезки, стрелки, треугольники и т. д.

Например, на рис. 4 пользователь рассчитал дополнительный массив, добавил клетку с вызовом функции визуализации и мышкой создал вид отображения, состоящий из точек и отрезков. Положение точек и отрезков он связал с первым массивом, а цвет точек — со вторым. Дополнительно пользователь мышкой задал цвет отрезков.

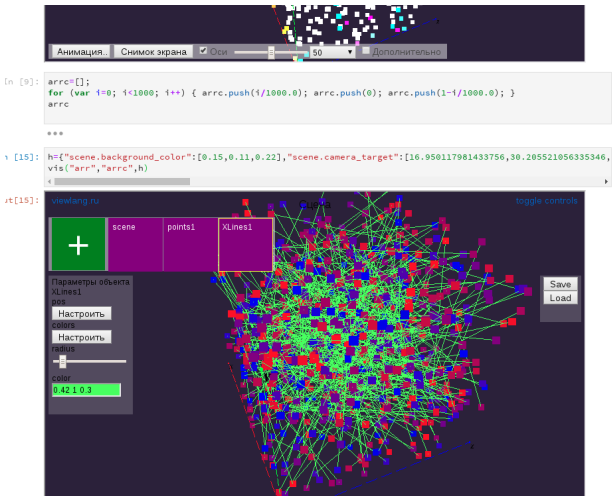


**Рис. 4.** Пример дальнейших действий пользователя.

Для сохранения состояния сцены применяется следующий подход. Если пользователя удовлетворяет полученная сцена, он нажимает кнопку «Сохранить» и получает текстовую запись с описанием настроенного вида отображения в формате json. Далее пользователь сохраняет его в программном коде клетки блокнота, передавая в качестве аргумента в функцию вызова визуализации. При последующем запуске вид отображения воспроизводится по этому описанию.

## 8. Заключение

Дальнейшее развитие представленной работы авторы видят в следующем:

1. Проверка тестовой реализации с целью понимания меры удобства использования в реальных задачах.
2. Выявление набора видов отображений, комбинации которых эффективно решают задачи визуализации в прикладных областях.
3. Создание удобного визуального языка формирования специализированных видов отображения.

Результаты программных разработок публикуются на сайте Сектора компьютерной визуализации ИММ УрО РАН www.cv.imm.uran.ru.


Работа опубликована при финансовой поддержке РФФИ, грант 18-07-20045\18

## Благодарности

Авторы выражают благодарность О. Г. Анненковой за экспертизу в области вычислительных блокнотов и критику настоящей работы.

## Литература

[1] Авербух В.Л., Байдалин А.Ю., Бахтерев М.О., Васёв П.А., Казанцев А.Ю., Манаков Д.В., Опыт разработки специализированных систем научной визуализации // Научная визуализация. Квартал 4. Том 2. Номер 4. 2010. Стр. 27-39.

[2] https://en.wikipedia.org/wiki/Notebook_interface

[3] https://www.lightbend.com/blog/scala-and-spark-notebook-the-next-generation-data-science-toolkit

[4] https://en.wikipedia.org/wiki/Direct_manipulation_interface

## Об авторах

Авербух Владимир Лазаревич, к.т.н., начальник Сектора компьютерной визуализации ИММ УрО РАН, averbukh@imm.uran.ru.

Васёв Павел Александрович, научный сотрудник Сектора компьютерной визуализации ИММ УрО РАН, vasev@imm.uran.ru.

Бахтерев Михаил Олегович, научный сотрудник Сектора компьютерной визуализации ИММ УрО РАН, m.bakhterev@imm.uran.ru.

Манаков Дмитрий Валерьявнович, программист Сектора компьютерной визуализации ИММ УрО РАН, manakov@imm.uran.ru.

Филоненко Дмитрий Юрьевич, канд. культурологии, доцент кафедры Графического дизайна УрГАХУ, philonenko@gmail.com.

Форгани Маджид Али, аспирант Института математики и компьютерных наук УрФУ, majid.forqani@gmail.com.



The publication of the conference proceedings was supported by the RFBR, grant No. 18-07-20045\18