ТЕХНОЛОГИЯ РАЗРАБОТКИ ВИЗУАЛИЗАТОРОВ АЛГОРИТМОВ

Г.А. Корнеев
Научный руководитель – д.т.н., профессор А.А. Шалыто

Предлагается новая технология разработки визуализаторов, основанная на применении конечных автоматах и поддержанная разработанным автором пакетом *Vizi*.

Введение

При изучении алгоритмов обработки информации, представляемой различными структурами данных [1, 2], важную роль играют визуализаторы алгоритмов, позволяющие в наглядной форме динамически отображать детали их работы. Это открывает возможность использования новой технологии обучения дискретной математике и программирования [3, 4].

Визуализатор – это программа, в процессе работы которой на экране компьютера динамически демонстрируется применение алгоритма к выбранному набору данных. Визуализаторы позволяют изучать работу алгоритмов в пошаговом режиме, аналогичном режиму трассировки программ. Они при необходимости допускают трассировку укрупненными шагами, игнорируя рутинную часть вычислительного процесса, что существенно, например, для переборных алгоритмов.

В настоящей работе рассматривается процесс построения визуализаторов и из частей.

Процесс построения визуализатора алгоритма достаточно сложен, поэтому разобьем его на отдельные составляющие.

В работе рассматриваются основные части визуализатора и особенности их разработки с использованием пакета *Vizi*. Затем описывается порядок разработки визуализатора с использованием пакета *Vizi*. В заключении статьи проводится сравнительный анализ методов построения визуализаторов на основе пакета *Vizi* и без него.

Основные части визуализатора

Визуализатор является сложным программным продуктом. Поэтому выделим в нем основные части, которые можно рассматривать самостоятельно. Тогда процесс построения визуализатора сведется к их построению и интеграции.

Выделение основных частей визуализатора

Целесообразно выделить следующие основные части визуализатора:

- логика визуализатора;
- визуальное представление;
- набор комментариев;
- элементы управления;
- интерфейс визуализатора;
- проектная документация.

Опишем указанные части подробнее.

Логика визуализатора – часть визуализатора, осуществляющая передвижение по алгоритму и предоставляющая данные другим частям визуализатора для отображения их пользователю. Логика визуализатора также определяет *интересные* состояния – те состояния, в которых информация отображается пользователю.

Например, при визуализации алгоритма поиска максимального потока в сети вряд ли стоит выделять в поиске кратчайшего пути между вершинами отдельные состояния. Действия таких «подалгоритмов» обычно отображаются как одна операция. С другой

стороны, в визуализаторе алгоритма Дейкстры поиска кратчайшего пути будет выделено множество интересных состояний.

Логика визуализатора должна обеспечивать возможность движения по алгоритму как вперед, так и назад на неограниченное количество шагов. При этом состояние визуализатора на любом шаге должно оставаться неизменным, вне зависимости от последовательности действий, совершенных пользователем.

Визуальное представление – часть визуализатора, определяющая, что и как будет отображаться пользователю в интересных состояниях. Обычно визуальное представление задается набором изображений (схем изображений), которые отображаются пользователю в процессе визуализации.

Визуальное представление предназначено для облегчения понимания визуализируемого алгоритма. Например, в визуализаторе алгоритма на графах граф может отображаться различными способами: графически (кружками и стрелками), матрицей смежности или списками ребер.

Набор комментариев – часть визуализатора, определяющая, какие сообщения будут выводиться пользователю в интересных состояниях. Комментарий может включать в себя данные, предоставленные логикой визуализатора, и описывать визуальное представление.

Комментарии поясняют текущее действие алгоритма и помогают пользователю глубже и быстрее понять смысл производимых действий, а, следовательно, и сам алгоритм. Хороший набор комментариев позволяет понять алгоритм и без визуального представления.

Элементы управления – часть визуализатора, посредством которой пользователь управляет визуализатором. При этом определяются набор действий, которые пользователь может осуществлять с визуализатором, и связь этих действий с логикой визуализатора. Примерами таких действий могут служить передвижение по алгоритму вперед и назад (при этом могут быть использованы маленькие и большие шаги), изменение количества элементов (вершин в графе, размерность массива), генерация случайных исходных данных.

Интерфейс визуализатора – часть, указывающая, каким образом остальные части визуализатора отображаются на экране (графический интерфейс), а также определяющая способы взаимодействия пользователя с элементами управления (функциональный интерфейс). В нашем случае интерфейс визуализатора должен определять отображение визуального представления и набора комментариев.

Проектная документация отображает все стадии разработки визуализатора и описывает получившийся продукт. Она позволяет легко разобраться во внутренней структуре визуализатора и помогает находить и устранять ошибки.

Разработка основных частей визуализатора

Рассмотрим разработку основных частей визуализатора по отдельности и покажем, как пакет *Vizi* может быть использован для упрощения их разработки.

Заметим, что, несмотря на то, что строится визуализатор алгоритма, часто говорят о визуализируемой программе, а не о визуализируемом алгоритме. Эти понятия существенно различаются.

- *Визуализируемый алгоритм* – алгоритм, который поясняет визуализатор.
- *Визуализируемая программа* – конкретная реализация алгоритма, на основе которой строится визуализатор.

Логика визуализатора разрабатывается для каждого визуализатора в отдельности. При этом наличие логики визуализатора для другого (даже схожего) алгоритма не облегчает разработку логики для нового алгоритма.

В соответствии с назначением логика визуализатора может быть представлена как совокупность двух частей:

- *программа визуализации*,
- *модель данных*.

Программа визуализации служит для передвижения по алгоритму. В соответствии с выдвинутыми требованиями она должна быть обратимой – допускать движение как вперед, так и назад. Программа визуализации также определяет интересные состояния и должны сигнализировать другим частям об их наступлении.

Модель данных – структура данных, в которой хранятся значения всех переменных программы. Она служит для сохранения состояния алгоритма, а также передачи информации от программы визуализации к другим частями визуализатора.

Таким образом, текущее состояние модели данных и точка выполнения программы визуализации (отсчитываемая по динамическим и текстуальным индексам [5]) полностью задают состояние визуализатора.

Пакет *Vizi* облегчает создание обоих частей логики визуализатора. Для этого визуализируемая программа записывается в виде XML-описания, которое полностью повторяет структуру исходной программы.

Программа визуализации строится на основе системы взаимосвязанных конечных автоматов, которая генерируется по XML-описанию визуализируемой программы. При этом для каждой процедуры строятся прямой и обратный автоматы, позволяющие осуществлять шаги вперед и назад.

Модель данных также автоматически генерируется пакетом *Vizi* по XML-описанию визуализируемой программы и является структурой (классом), в которую вынесены переменные, используемые визуализируемой программой.

Визуальное представление шагов алгоритма должно быть выбрано так, чтобы наиболее наглядным образом отобразить состояние визуализируемого алгоритма.

В большинстве случаев не имеет смысла разрабатывать визуальное представление с нуля. Например, во многих случаях требуется отображать одинаковые структуры данных, такие как таблицы, массивы и списки. В этих случаях следует воспользоваться уже разработанными элементами для их отображения, входящими в пакет *Vizi*.

Действия, осуществляемые для создания изображений в интересных состояниях, могут быть указаны в XML-описании визуализируемой программы. Это облегчает программирование.

Визуальное представление разрабатывается после того, как будут определены интересные состояния алгоритма. При этом разработанное визуальное представление может не быть самодостаточным, а требовать дополнительных комментариев. Поэтому и введена возможность отображения комментариев.

Набор комментариев. В отличие от визуального представления и элементов управления, комментарии необходимо разрабатывать «с нуля», так как для каждого алгоритма они уникальны. По комментариям в совокупности с визуальным представлением должно быть понятно, как работает алгоритм и зачем выполняются те или иные действия.

Оперирование с конкретными числами (значениями) во многом помогает разобраться в алгоритме. Поэтому пакет *Vizi* предоставляет средства для включения в комментарии конкретных значений переменных во время исполнения.

Элементы управления. В большинстве визуализаторов используются сходные элементы управления. В частности, все визуализаторы должны содержать кнопки переходов по алгоритму вперед и назад.

В пакете *Vizi* представлены часто используемые элементы управления, интегрированные с автоматным представлением программы визуализатора.

Интерфейс визуализатора. При выполнении нескольких визуализаторов следует придерживаться единого интерфейса. Таким образом, пользователям, освоившим один визуализатор, не составит труда разобраться в остальных.

В пакете *Vizi* реализован единый интерфейс визуализаторов, состоящий из трех областей, расположенных одна над другой:

- область визуального представления;
- область комментариев;
- область элементов управления.

Такой интерфейс делает работу с визуализатором простой и позволяет получать скриншоты визуального представления с комментариями, например для последующей публикации.

Проектная документация является неотъемлемой частью визуализатора и должна разрабатываться одновременно с ним. Более подробную информацию о проектной документации к визуализаторам, созданным на основе пакета *Vizi*, можно найти в работе [7].

Таким образом, пакет *Vizi* позволяет упростить программирование всех основных частей визуализатора, кроме создания проектной документации, что существенно убыстряет создание визуализаторов.

Порядок разработки визуализатора

В настоящем разделе рассматривается порядок разработки визуализаторов с использованием пакета *Vizi*.

Пакет *Vizi* позволяет упростить и убыстрить процесс разработки визуализатора. Опишем соответствующий порядок разработки визуализатора.

- Анализ литературы.
- Создание визуализируемой программы.
    - Реализация алгоритма.
    - Отладка реализации.
- Разработка концепции визуализатора
    - Выделение интересных состояний.
    - Разработка концепции визуального представления.
    - Разработка набора комментариев.
    - Разработка элементов управления.
- Построение XML-описания визуализируемой программы.
    - Выделение модели данных.
    - Упрощение конструкции программы.
    - Запись XML-описания.
    - Отладка XML-описания.
- Добавление комментариев к XML-описанию.
- Реализация визуального представления.
- Реализация элементов управления.
- Интеграция визуализатора.
    - Генерация кода по XML-описанию.
    - Совместная отладка логики и визуального представления.
- Оформление проектной документации.

На первом этапе производится анализ литературы. При этом рассматриваются существующие модификации алгоритма и одна из них выбирается для визуализации.

На следующем этапе реализуется выбранная модификация алгоритма. Одновременно полученная реализация отлаживается. При этом выделяются шаги алгоритма, требующие особого внимания. Полученная информация используется на следующем этапе. При реализации алгоритма на втором этапе следует избегать применения сложных структурных конструкций, что впоследствии облегчит упрощение программы для записи XML-описания.

На третьем этапе разрабатывается общая концепция визуализатора. Для этого сначала выделяются те шаги алгоритма, которые представляют наибольший интерес, и соответствующие интересные состояния. Затем разрабатывается общая концепция визуального представления и ее конкретизация для каждого выделенного шага. Одновременно с разработкой визуального представления для каждого шага пишутся комментарии, поясняющие действия, выполняемые алгоритмом. После этого разрабатываются элементы управления визуализатором, в частности, определяется, какие параметры и в каких пределах сможет регулировать пользователь.

На четвертом этапе осуществляется построение XML-описания визуализируемой программы.

Для этого сначала выделяется модель данных. При использовании системы визуализации *Vizi* этот шаг существенно упрощается, так как возможность использования локальных переменных и параметров процедур предусмотрена в XML-описании. Таким образом, этот шаг сводится к простому преобразованию заголовков процедур в XML-формат.

После выделения модели данных производится упрощение структуры. При этом программа должна использовать ограниченный набор операторов. После упрощения получается программа, более удобная для преобразования в систему автоматов, так как она не содержит громоздких синтаксических конструкций.

Упрощенная программа может быть непосредственно записана в XML-формате, так как в нем предусмотрены элементы для кодирования всех видов операторов. Программа, записанная в XML-формате, может быть автоматически преобразована пакетом *Vizi* в систему взаимосвязанных автоматов. Полученная программа отлаживается с использованием средств, предоставляемых пакетом *Vizi*.

На пятом этапе к XML-описанию визуализатора добавляются комментарии, разработанные на третьем этапе. Для этого у каждого элемента, соответствующего оператору исходной программы, заполняются поля комментариев и их параметров.

На шестом этапе реализуется концепция визуального представления шагов визуализатора. Этот этап может быть упрощен за счет использования визуальных элементов, входящих в пакет *Vizi*. Для связи визуального представления и XML-описания применяются элементы `draw`.

Реализация элементов управления на седьмом этапе существенно упрощается за счет использования элементов управления, входящих в пакет *Vizi*.

На восьмом этапе осуществляется интеграция визуализатора с использованием средств, предоставляемых пакетом *Vizi*. При этом по XML-описанию автоматически генерируется программа визуализации, модель данных, система отображения комментариев, а также осуществляется интеграция с реализацией визуального представления и стандартными элементами управления. После отладки получается готовый визуализатор.

На заключительном этапе оформляется проектная документация. Заметим, что проектная документация должна вестись все время, пока выполняется проект. На последнем этапе производятся только оформление проектной документации и обобщение полученных результатов.

Заключение

Использование пакета *Vizi* позволяет уменьшить как техническую, так и интеллектуальную сложность разработки визуализатора.

Техническая сложность уменьшается за счет разработки основных частей визуализатора с использованием компонент, входящих в пакет *Vizi*.

Интеллектуальная сложность разработки уменьшается за счет того, что разработанный пакет автоматически строит обращенную версию программы. Ранее построение обращенной версии программы было связано с большими трудностями, и на этом этапе допускалось много ошибок.

Литература

1. Кнут Д. Искусство программирования. Том 1. Основные алгоритмы. М.: Вильямс, 2000.
2. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы. Построение и анализ. М.: МЦНМО, 1999.
3. Казаков М.А., Столяр С.Е. Визуализаторы алгоритмов как элемент технологии преподавания дискретной математики и программирования // Международная научно-методическая конференция «Телематика-2000». СПб, 2000. С.189-191.
4. Корнеев Г.А., Парфенов В.Г., Столяр С.Е., Васильев В.Н. Визуализаторы алгоритмов как основной инструмент технологии преподавания дискретной математики и программирования / Труды международной научно-методической конференции «Телематика-2001». СПб.: СПбГИТМО (ТУ), 2001. С.119-10.
5. Дал У., Дейкстра Э., Хоор К. Структурное программирование. М.: Мир, 1975.
6. Лингер Р., Миллс Х., Уитт Б. Теория и практика структурного программирования. М.: Мир, 1982.
7. Корнеев Г.А., Шалыто А.А. Требования в визуализаторам алгоритмов, выполняемых на базе технологии Vizi. http://ctddev.ifmo.ru/vizi/Requirements-4_00.pdf.