Российская академия наук
Сибирское отделение
Институт систем информатики
им. А. П. Ершова

И. А. Лисицын

СИСТЕМЫ ВИЗУАЛИЗАЦИИ И РЕДАКТИРОВАНИЯ ГРАФОВЫХ ОБЪЕКТОВ

ОБЗОР

Препринт
76

Новосибирск 2000

Настоящий обзор посвящен описанию и сравнению существующих на сегодняшний день инструментальных средств визуализации и конструирования графовых объектов. Рассматривается весь спектр характеристик систем — от графических возможностей и качества пользовательского интерфейса до наличия дополнительных средств выполнения и визуализации алгоритмов на графах. Оценивается общее состояние в области создания программных средств, связанных с визуализацией графов, и указываются тенденции, определяющие развитие данного направления.

Работа поддержана Российским фондом фундаментальных исследований, грант № 00-07-90296.

Siberian Division of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

Ivan A. Lisitsyn

GRAPH VISUALIZATION AND EDITING SYSTEMS

Preprint
76

Novosibirsk 2000

This review is devoted to description and comparison of currently available tools for visualization and construction of graph objects. The entire spectrum of system characteristics is considered, from graphic possibilities and quality of the user interface to additional means intended to perform and visualize graph algorithms. In conclusion, the state-of-the-art of software design related to graph visualization is assessed and the trends of its further development are outlined.

This work is partially supported by Russian Foundation for Basic Research under grant N 00-07-90296.

1. ВВЕДЕНИЕ

Графы широко применяются в различных областях точных и естественных наук, так как позволяют строить наглядные и удобные для обработки модели сложных структур. При создании таких моделей граф наделяется некоторой семантикой, которая обычно выражается набором атрибутов, приписываемых его вершинам и дугам.

В программировании рассматриваются управляющие графы программ, графы состояний, модели компьютерных сетей, сети Петри, модели СБИС и другие графовые структуры. Существенным образом графы используются в теории компиляции и преобразования программ. В любом трансляторе программа на входном языке переводится в некоторое промежуточное представление, более удобное для дальнейшей обработки, например для целей оптимизации, а также для переносимости. Часто это представление бывает графовым [1].

Преимущества графовых моделей во многих случаях становятся ощутимыми только при наличии хороших инструментальных средств их визуализации и обработки. Поэтому в настоящее время в мире происходит значительный рост интереса к методам и системам визуализации графов, о чем свидетельствует рост публикаций, содержащих описание новых алгоритмов, способов визуализации графов и новых инструментальных средств. Полную библиографию, содержащую более 300 ссылок на различные публикации в этой области до 1994-го г., можно найти в [8]. Более современный обзор теоретических разработок содержится в [9].

Настоящий обзор посвящен инструментальным средствам визуализации и конструирования графовых объектов.

2. ИСТОРИЯ РАЗВИТИЯ СРЕДСТВ ВИЗУАЛИЗАЦИИ ГРАФОВ

Разработка средств визуализации графов началась во второй половине 80-х гг. и существенно активизировалась с начала 90-х. Это было связано, с одной стороны, с возросшей потребностью оперировать большими объемами информации и сложными структурами данных, которые достаточно естественным образом представляются графами, с другой стороны, с существенным прогрессом в развитии аппаратных средств, позволившим сделать графический интерфейс и компьютерную графику удобными и эффективными средствами общения человека с компьютером.

Многие системы, особенно ранние разработки, возникали как части более крупных проектов, в которых требовался компонент, производящий визуализацию. Некоторые из таких систем позже удавалось сделать более универсальными и оформить как самостоятельные программные продукты. В других случаях особенности изначальной ориентации системы оказывались столь существенными, что она могла применяться только для узкоспециальных целей. Таких систем в настоящее время создано достаточно много. Еще большее количество различных программ содержит сравнительно простые компоненты визуализации графов, предназначенные только для вывода некоторой структурированной информации. Примером может служить вывод дерева классов в компиляторе Borland C++.

В середине 90-х гг. были сделаны попытки создания универсальных систем визуализации графов. Хотя при создании этих систем заранее намечалось, в каких проектах их предполагалось использовать, их универсальность теперь закладывалась изначально.

Широкое использование универсальных графовых визуализаторов и особенно графовых редакторов затруднялось тем, что в каждом проекте требовалось работать с графами, имеющими специфическую семантику. В то же время не существовало систем, которые позволяли бы настраиваться на семантику графа без изменения кода самой системы. Требовалось, во-первых, создать соответствующее внутреннее представление, во-вторых, организовать визуализацию графа так, чтобы полноценно представлять все атрибуты его вершин и дуг.

Выходом из положения стали библиотеки классов на C++, позволяющие описывать семантику графа путем создания производных классов с дополнительными атрибутами. Такие библиотеки обычно включают множество других полезных функций, связанных как с визуализацией графов, так и с выполнением на них различных теоретико-графовых алгоритмов и алгоритмов рисования.

Зачастую библиотеки для работы с графами возникают как побочный продукт при создании систем визуализации графов. И наоборот, при разработке каждой библиотеки создается хотя бы небольшая система, необходимая для тестирования библиотеки и демонстрации ее возможностей. Кроме того, новые системы могут создаваться на основе уже существующих библиотек.

В последнее время с развитием науки о визуализации графов и ее повсеместным распространением начали появляться многочисленные системы, развивающие отдельные аспекты данного направления и считающиеся перспективными в настоящее время. В качестве примеров можно привести

системы, работающие с трехмерными изображениями графов [28, 29, 37]. Хотя до сих пор не создано полноценного редактора для такого представления, трехмерная визуализация графов уже используется на практике. Другим примером является визуализация очень крупных графов, то есть графов с настолько большим числом вершин, что применение обычных методов визуализации для них невозможно [30, 31]. Данная проблематика не в последнюю очередь связана с популяризацией Интернета и необходимостью вести исследования различного характера в области больших компьютерных сетей.

Стоит отметить, что создание среднего по возможностям графового редактора сравнительно легкая и очень полезная задача для начинающего программиста или человека, решившего освоить методы создания графических программ или, к примеру, изучить новый язык программирования. Этим, по всей видимости, объясняется большое число систем, написанных на ставшем популярным во второй половине 90-х гг. языке Java. В качестве примеров можно привести VGJ [26], Grappa [27] и Graph Drawing Server [35, 36]. Иногда такие системы состоят из клиентской и серверной частей. Клиентская часть служит графовым редактором, а на сервере запускаются алгоритмы рисования или преобразования графов.

Для подробного рассмотрения в рамках данного обзора были отобраны самые известные на сегодняшний день универсальные средства визуализации графов. Наиболее полный список систем и библиотек, относящихся в той или иной мере к данной области, можно найти в [33] и [34].

При описании систем будут рассматриваться следующие аспекты:

1. Основное назначение системы.
2. Графические возможности.
3. Интерфейс пользователя.
4. Встроенные алгоритмы рисования графов.
5. Прочие возможности системы.
6. Дополнительная информация.

3. СИСТЕМА DAVINCI

Система daVinci [15, 16] разрабатывается с конца 1992 г. в отделении Computer Science Бременского университета (Германия). Авторы системы сообщают, что у них имеются сведения о более чем 2000 установок daVinci по всему миру. Существуют версии системы для нескольких UNIX-платформ, включая рабочие станции и персональные компьютеры.

Данная система изначально предназначалась для визуализации и редактирования ориентированных ациклических графов, которые обычно изображаются с условием, что все дуги направлены в одну сторону, например вниз. Стандартный алгоритм рисования для этого случая раскладывает вершины по уровням так, что все дуги идут от меньших уровней к большим. Если дуга проходит через несколько уровней, то в нее добавляются сгибы, рассматриваемые как дополнительные вершины с нулевыми размерами, лежащие в пересекаемых дугой уровнях. После этого при помощи различных эвристик происходит упорядочение вершин и сгибов дуг внутри каждого уровня с тем, чтобы минимизировать число пересечений дуг графа. Первое описание алгоритма такого типа можно найти в [38]. Все последующие усовершенствования, в том числе алгоритм данной системы, базируются на этой работе.

Позже в систему были внесены элементарные изменения, позволившие ей работать и с графами произвольного вида. Эти изменения состояли в введении для каждой дуги специального параметра, задающего, на каком ее конце рисовать стрелку. Доступны четыре варианта: Parent -> Child, Parent <- Child, Parent --- Child и Parent <-> Child. Таким образом, во внутреннем представлении, которое используется алгоритмом рисования, граф остается ориентированным и ациклическим, хотя внешне может выглядеть иначе.

Система имеет средние графические возможности. Для дуг предусмотрено пять стилей линий (дуги рисуются ломаными линиями) и выбор цвета из 24-х вариантов. Для вершин задается форма: Box, Rhombus, Circle, Ellipse, Text или Icon. Вершины формы Text и Icon не имеют каемки, характерной для всех других форм вершин. Для вершин формы Icon дополнительно задается имя файла, из которого считывается изображение вершины. Каемка вершины, рисуемая всегда черным цветом, может быть одинарной или двойной. Для внутреней части вершины цвет выбирается также из 24-х вариантов. Каждая вершина может содержать внутри некоторый текст, возможно многострочный. Для изображения этого текста можно выбирать один из четырех доступных шрифтов. Палитра шрифтов дополнена переключателями bold и italic. Размер шрифта устанавливается сразу для всего графа. Дуги графа не имеют ассоциированного с ними текста.

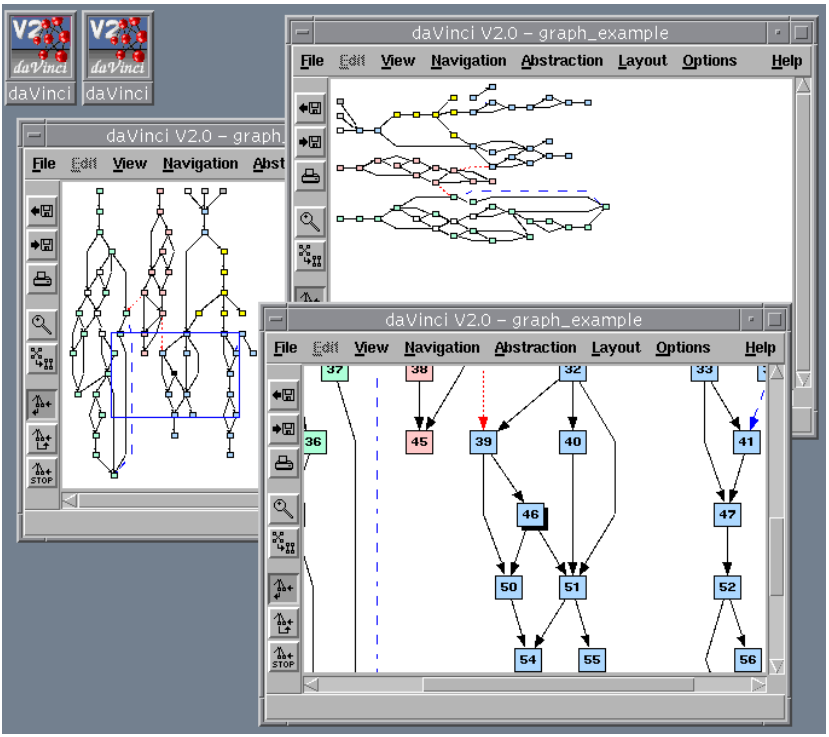

Рис. 1. Система daVinci

Интерфейс системы достаточно удобен, хотя обладает рядом мелких недостатков. Имеется панель инструментов, в которую, однако, вынесено лишь минимальное количество функций. Как утверждается в документации, система daVinci сама по себе не является графовым редактором. Графовый редактор может подключаться к ней как отдельное приложение. Действительно, после запуска системы меню Edit затемнено и операции редактирования графа недоступны. Для подключения редактора нужно выбрать в меню File пункт Connect application, после чего в открывшемся блоке диалога выбрать файл с графовым редактором. После этого граф можно редактировать с помощью меню Edit.

Под редактированием графа понимается изменение его топологической структуры, а также параметров вершин и дуг. Корректировку расположения вершин можно производить и при отключенном редакторе. Вершины до-

бавляются в граф с помощью команд меню Insert root и Insert child. Первая добавляет вершину, не имеющую родителя, вторая — вершину, родителем которой становится выделенная вершина графа. Можно также добавить дополнительную дугу, выделив две вершины и выбрав пункт Insert edge. После добавления или удаления вершины или дуги алгоритм рисования запускается заново.

Изображение графа, полученное алгоритмом рисования, можно корректировать, перемещая вершины внутри уровня и с уровня на уровень. При этом дополнительные сгибы дуг возникают автоматически. Их тоже можно двигать внутри уровня. После такой корректировки граф можно записать в файл вместе с текущим расположением. Другой вариант записи — запись без информации о расположении. При чтении такого файла система автоматически запускает алгоритм рисования.

Масштаб изображения графа можно менять от 1 до 100%. При этом текст вершин изображается только при максимальном масштабе. Полезной возможностью является открытие отдельного окна, содержащего обзорный вид графа с прямоугольником, ограничивающим видимую в обычном окне часть графа. Однако этот прямоугольник нельзя, как обычно в таких случаях, перемещать по графу. Вместо этого в обзорном окне можно выделить вершину, после чего данный прямоугольник вместе с изображением в обычном окне переместится так, что выбранная вершина попадет в его центр. При этом окно будет многократно перемигивать, полностью перерисовываясь при каждом сдвиге прямоугольника, который перемещается к цели в несколько шагов. Этот процесс в описании системы назван "animation".

Алгоритм расположения имеет параметр качества от 1 до 5. Чем выше этот параметр, тем дольше работает алгоритм и тем лучше получается итоговое расположение. Кроме этого можно регулировать размеры промежутков между вершинами и устанавливать направление развертывания уровней (стандартный вариант — сверху вниз). Запускать алгоритм можно либо на всем графе, либо только на видимой в данный момент его части.

В системе предусмотрено еще две полезные возможности, называемые авторами abstraction и navigation. Abstraction есть способ сокрытия части информации о графе, которая может быть в определенные моменты излишней. Есть два варианта такого сокрытия для произвольной вершины графа:

1) скрыть все дуги, инцидентные данной вершине;
2) скрыть подграф, образованный данной вершиной и всеми ее сыновьями.

После выполнения этих операций вершина помечается специальным образом. Если далее запустить алгоритм рисования, то он уже выдаст другое расположение. К сожалению, после применения обратного преобразования открывшиеся вершины могут появиться в каком угодно месте плоскости. Чтобы привести расположение в порядок, нужно снова запускать алгоритм рисования.

Navigation предусматривает ряд возможностей для выделения вершин путем "хождения" по графу. Так, например, можно выделить вершину, а потом перейти к ее родителю, правому или левому брату и т. д. Можно также организовать поиск вершины по ее тексту.

В заключение отметим, что несомненным достоинством системы является наличие подробной документации, оформленной в виде HTML-файлов. При этом возможно получение контекстных справок, для чего из системы автоматически запускается Netscape Navigator (если он присутствует на компьютере). Для поиска в help-файлах по ключевым словам используется специальный Java-applet.

4. СИСТЕМА VCG

Система VCG [13, 14] — Visualization of Compiler Graphs, — разработанная в Saarland University в Германии, не является редактором графов и предназначена только для их визуализации. Она принимает графы в виде файлов специального формата, который помимо самого графа может содержать дополнительную информацию о методе визуализации отдельных его компонент, например координаты некоторых вершин.

Формат файла достаточно сложен, поэтому такие файлы вряд ли возможно создавать и редактировать вручную, хотя они и текстовые. Таким образом, система целиком предназначена для визуализации автоматически сгенерированных графов.

Система VCG создавалась для UNIX X11-платформ, но позже была перенесена на MS Windows 3.1 с небольшими сокращениями. Хотя при этом перенесении использовались новые библиотеки, многие особенности системы перекочевали из X11 и для пользователя Windows выглядят несколько странно.

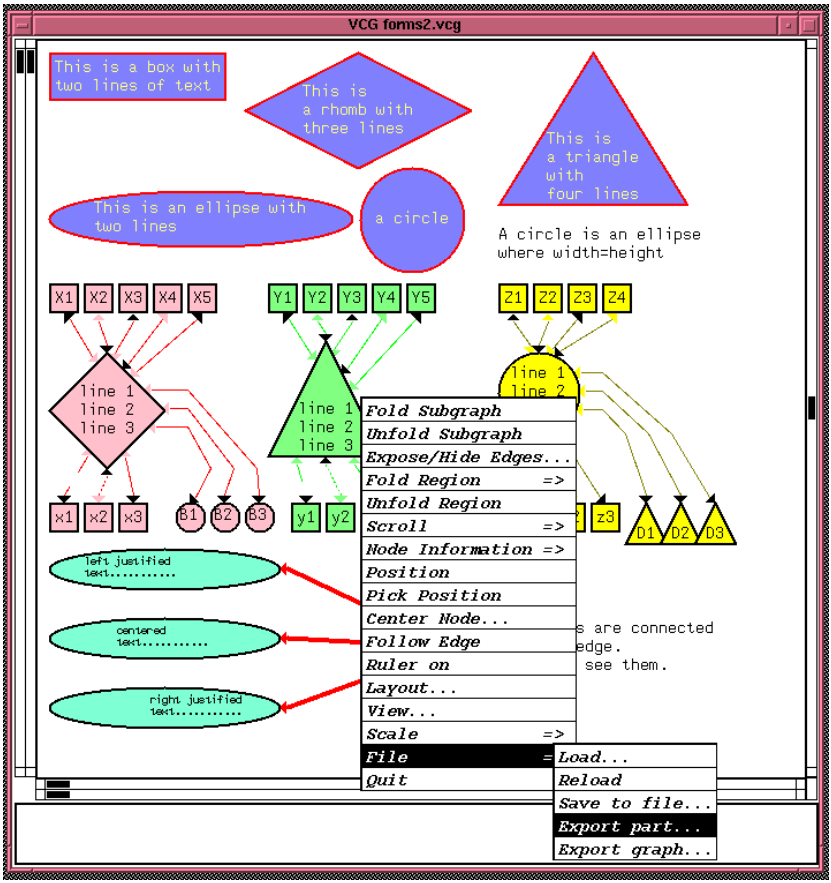

Рис. 2. Различные варианты вершин и дуг в системе VCG

Система имеет сравнительно хорошие графические возможности. Вершины изображаются прямоугольниками, эллипсами, треугольниками и ромбами, дуги — ломаными линиями либо сплайнами, аппроксимирующими ломаные.

Используется несколько вариантов присоединения дуги к вершине. Цвет вершин и дуг выбирается из 254 вариантов. Вершины и дуги могут иметь многострочные текстовые метки. При этом шрифт меток и его размер задаются отдельно для каждого элемента графа. К сожалению, система

не использует шрифты Windows, поэтому набор доступных шрифтов ограничен теми, которые поставляются вместе с системой в неизвестном формате. Пополнять этот набор невозможно.

В описании системы утверждается, что для размещения графа в ней используется 13 основных методов и 4 способа сокращения числа пересечений дуг. Само по себе заявление достаточно странное, так как сокращение числа пересечений дуг есть одна из задач любого метода рисования графов. Трудно сказать, что имели в виду авторы, говоря про 13 методов, но, если судить по внешнему виду, представляется более естественным утверждать, что в данном случае мы имеем дело с одним алгоритмом, содержащим очень большое число параметров. Сам же алгоритм аналогичен алгоритму системы daVinci и тоже базируется на [38]. Подробно он описан в документации к системе. Авторы утверждают, что приоритетом при разработке этого алгоритма для них была скорость работы. Действительно, алгоритм работает достаточно быстро даже на больших графах, производя при этом сравнительно качественное размещение. Кроме того, в системе предусмотрено переключение алгоритма в ускоренный режим по истечении отведенного на размещение времени.

Достаточно широк набор средств просмотра графа, включающий возможность произвольного масштабирования изображения, наличие обзорного окна и использование техники Fisheye. Эта техника позволяет увеличивать отдельные фрагменты графа, оставляя сам граф в поле зрения, но с меньшим масштабом. Внешне это выглядит, как просмотр изображения через очень сильное увеличительное стекло.

Система имеет возможность экспортировать изображение графа в PostScript, Windows BMP и несколько других графических форматов, используемых в основном UNIX-системами.

VCG имеет ряд дополнительных возможностей, суть которых аналогична тому, что в daVinci называется abstraction.

VCG поставляется вместе с программой, демонстрирующей возможности системы.

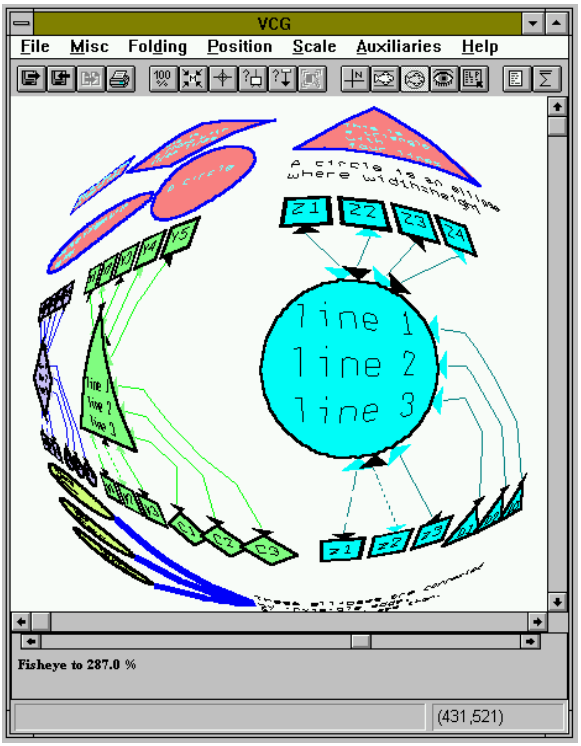

Рис. 3. Техника Fisheye, позволяющая просматривать в увеличенном виде отдельные фрагменты графа

5. СИСТЕМЫ GRAPHED И GRAPHLET

Система GraphEd [10] разрабатывалась в Passau University с 88-го по 94-й г., после чего усилия создателей были направлены на новый проект, получивший название Graphlet [11, 12]. Так как последний включает и расширяет возможности своего предшественника, его и будем рассматривать.

Graphlet представляет собой комплект инструментальных средств для создания графовых редакторов и программирования алгоритмов рисования графов. Graphlet базируется на библиотеке LEDA и языке Tcl/Tk. Tcl — универсальный интерпретируемый язык программирования, а Tk — расши-

рение Tcl, дополняющее его графическими возможностями. Существуют интерпретаторы Tcl/Tk для различных UNIX-платформ, а также для MS Windows 95/NT.

Graphlet включает следующие компоненты:

- ядро, содержащее основные структуры данных и интерфейсную часть алгоритмов;
- интерфейсный уровень, содержащий реализацию некоторого расширения Tcl для работы с графами, названного LedaScript;
- основной компонент: графовый редактор, написанный на LedaScript;
- модули, содержащие графовые алгоритмы, в том числе алгоритмы рисования, реализованные на C++ и LEDA или LedaScript.

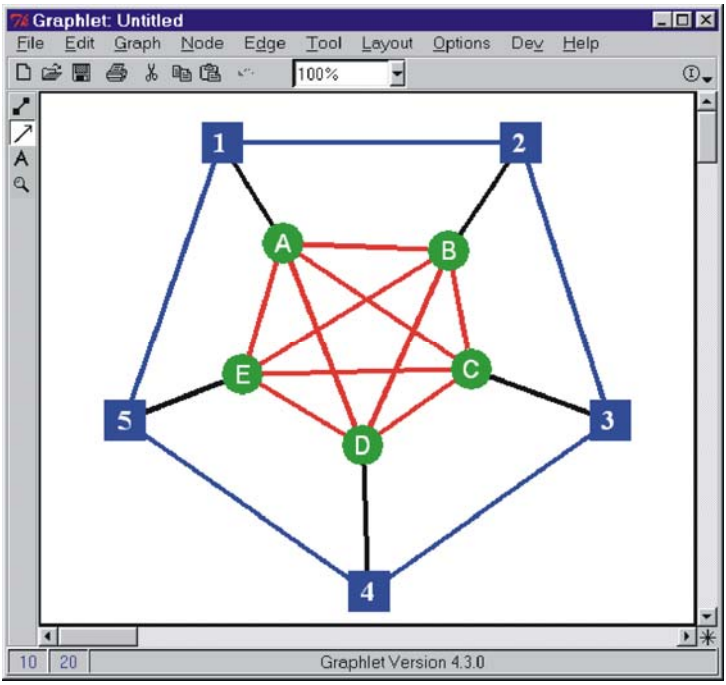

Рис. 4. Система GraphLet

Graphlet использует для хранения графов новый формат файла, названный GML и претендующий на существенную универсальность. Он является текстовым и позволяет определять дополнительные атрибуты для элементов графа. В настоящее время помимо Graphlet существует несколько систем, которые позволяют читать и записывать графы в формате GML.

Графовый редактор Graphlet можно охарактеризовать как систему с достаточно специфическим набором возможностей, которые удается изучить только после упорной борьбы с не очень удобным интерфейсом системы и постоянными сообщениями о внутренних ошибках, возникающих во время работы.

По сути дела, кроме тех средств, которые могут реально понадобиться при создании визуального представления графа, в систему включено множество других возможностей, полезность которых сомнительна. Примеры таких "странных" возможностей: вершины в форме дуги окружности (причем всегда верхней правой четверти) или задание сдвига для точки входа дуги в вершину — если этот сдвиг не нулевой, то дуга может кончаться, не доходя до вершины либо, наоборот, заходя на нее.

Реальных же графических возможностей не так много. Из обычных форм вершин используются только прямоугольник и эллипс. Цвет задается произвольно для внутренности вершины и ее каемки. Ширина каемки регулируется. Дуги рисуются ломаными линиями, для которых также задаются цвет и ширина. Вершины и дуги могут иметь по одной текстовой метке. Хотя формат GML и, как утверждается авторами, внутреннее представление системы могут хранить произвольное количество атрибутов элементов графа, в системе такая возможность не реализована. Шрифт и его размер можно выбирать для каждого элемента графа отдельно.

Как уже было сказано, интерфейс системы оставляет желать лучшего. Система работает в пяти режимах, переключаемых с помощью панели управления. Первый режим служит для создания вершин и дуг. Второй — для выделения групп вершин и дуг. После выделения группы элементов графа можно устанавливать любые параметры (метки, размеры и т.д.) одновременно для всех выделенных объектов. Кроме того, можно перемещать всю группу с помощью мыши и пользоваться функциями Cut/Copy/Paste. Третий режим служит для редактирования меток. Фактически он отличается от второго только запретом выделять более одного объекта. Четвертый режим — для просмотра графа. Все функции редактирования в нем отключены. Наконец, пятый режим реализует стандартную zoom-функцию (выделение на экране прямоугольника и увеличение его содержимого).

Для редактирования всех параметров элементов графа используется один блок диалога с несколькими кнопками, переключающими его содержимое. Стоит отметить, что размеры вершины графа в системе можно изменить, только введя их в численном виде в соответствующие поля этого блока диалога. Система имеет возможность автоматически корректировать размеры вершин, исходя из размера содержащегося в них текста, но в этом случае, если вершина имеет форму эллипса, после такой коррекции текст обязательно будет выступать за ее границу.

Достоинством системы является наличие большого количества интегрированных в нее алгоритмов рисования графов, а также некоторых теоретико-графовых алгоритмов. Всего в системе доступно 14 алгоритмов рисования (5 для планарных графов, 2 для деревьев, 1 для ациклических графов и 6 для графов произвольного вида).

Graphlet работает на нескольких UNIX-платформах, а также под MS Windows 95/NT, что достигается в основном за счет переносимости Tcl/Tk. Хотя, как уже было сказано, существует интерпретатор этого языка для MS Windows, сам язык изначально разрабатывался для UNIX-платформ, поэтому все интерфейсные особенности взяты из XWindow. Следствие — медленная работа и неудобство для пользователя, привыкшего к другим стандартам.

В заключение хочется отметить оригинальный подход авторов данного продукта к написанию документации. Ими написан внушительных размеров манускрипт, посвященный языку GraphScript, описание C++ интерфейса для расширения этого языка и описание формата GML. Однако сам графовый редактор, который, как заявлено в [11], является основным компонентом Graphlet, не только не имеет встроенной help-поддержки, но и вообще никак не документирован.

6. СИСТЕМА GRAVIS

Система GraVis [18] разрабатывается в группе параллельного программирования университета Tubingen в Германии.

Данная система является графовым редактором, оснащенным дополнительной возможностью запуска внешних модулей обработки графа, что используется в основном для встраивания в систему алгоритмов рисования.

Графические возможности средние. Используются четыре графических примитива для форм вершин: эллипс, прямоугольник, треугольник и ромб. В вершину можно вставлять изображения, которые растягиваются и сжи-

маются при изменении размеров вершины. Вершина с изображением внутри автоматически становится прямоугольной, но ее каемка не рисуется. Прочие вершины имеют тонкую черную каемку, и ее цвет изменить нельзя.

Можно выбирать цвет внутренности вершины и цвет дуг. Имеется 16 доступных цветов, хотя в описании речь идет о полной палитре. Для дуг выбирается также стиль линии из 9 вариантов. Никакие другие параметры изображения регулировать невозможно. В частности, стрелки дуг рисуются самым примитивным образом, а метки вершин и дуг черным цветом и одним и тем же шрифтом. Дуги рисуются ломаными линиями.

По умолчанию метки вершин размещаются в их центрах, а метки дуг — на самом длинном звене. Однако эти метки при желании можно переместить практически в любое место. Подобным образом дело обстоит с точкой привязки дуг к вершинам. По умолчанию дуги ориентируются по центру вершин, но можно перенести точку привязки куда угодно за пределы вершины. Таким образом, изображение графа в системе может в принципе не иметь ничего общего с его топологической структурой.

В системе имеются один основной режим редактирования и один дополнительный, в котором можно выделять группы вершин и сгибов дуг, после чего двигать их куда-либо или менять размеры. Такое построение интерфейса авторы считают новаторством и особенно отмечают, что в их системе операция, производимая щелчком кнопки мыши, зависит от того, где этот щелчок был произведен.

Сгибы дуг являются неким подобием дополнительных вершин. Скорее всего в реализации так и есть. Однако не очень понятно, как далеко эта аналогия распространяется и до каких пор она совпадает с желаниями авторов. В частности, сгибы дуг могут иметь собственные метки (зачем это нужно, если при запуске любого алгоритма рисования все сгибы уничтожаются и потом, если нужно, добавляются заново?). Для изменения размера вершин с помощью мыши около выделенной вершины появляется специальное поле, к которому нужно подвести курсор и нажать левую кнопку, после чего менять размеры, двигая мышь. Если выделить группу вершин, то таким образом можно менять размеры всех вершин группы. Сгибы дуг также можно выделять, чтобы перемещать их, но менять их размеры нельзя (соответствующее поле отсутствует). Если же выделить группу, состоящую из вершин и сгибов дуг, а потом воспользоваться полем какой-либо вершины для изменения размеров, то размеры точек сгиба (т.е. отмечающих их окружностей) также будут изменяться.

Существует возможность использовать сетку для размещения вершин, однако использовать эту сетку крайне неудобно, потому что она всегда

квадратная, причем с фиксированным размером. При перемещении вершин она относительная, при изменении их размеров — абсолютная (было бы разумно сделать наоборот). При этом размеры вершин меняются сразу на два деления сетки. Если учесть, что эти деления столь крупны, что на экране по горизонтали их умещается порядка десяти, то очевидно, что от такой сетки нет совершенно никакой пользы.

В системе можно отметить три несомненно положительных момента: 1) наличие функций Undo и Redo; 2) возможность выделять мышью некоторую область окна, а затем просматривать ее в увеличенном масштабе; 3) возможность запуска внешних модулей. Внешний модуль получает на вход текущий граф, обрабатывает его и либо возвращает результат системе, либо сам выводит его на экран, если это просто текст. При запуске модуля может возникать блок диалога для ввода дополнительных параметров алгоритма. В систему встроено порядка 15 модулей, около 10 из которых содержат алгоритмы рисования графов. Есть модуль для генерации случайных графов.

Интересной особенностью системы является возможность создания иерархических графов. Однако эта возможность организована крайне неэффективно. Можно выделить группу вершин и заменить ее одной мета-вершиной. Если мы теперь захотим посмотреть на свернутые вершины, то для этого потребуется открыть новое окно. Увидеть данные вершины в том же окне, что и раньше, можно, только заменив обратно мета-вершину на выделенный подграф.

Графы можно сохранять либо в собственном формате системы, либо в формате GML.

Помимо ряда отмеченных недоделок и ошибок к недостаткам системы можно отнести крайне медленное функционирование, например: при добавлении в граф вершины между щелчком мыши и появлением новой вершины на экране проходит больше секунды (эксперимент проведен на 486DX 100Мгц, 32Мбайт). Кроме того, данная система является лидером среди своих аналогов по количеству, мягко говоря, "преувеличений достоинств", содержащихся на www-странице системы и в [17].

Около года назад на базе GraVis был начат новый проект, получивший название Y Project [39]. Новая система представляет переработанный и несколько улучшенный вариант графового редактора с аналогичными возможностями. Система написана на языке Java и включает в себя несколько дополнительных функций, таких как анимация алгоритмов. Однако эта возможность предусмотрена только для встроенных в систему алгоритмов и пока не очень развита. Следует отметить, что новая система работает ста-

бильнее и не имеет такого количества недостатков, как GraVis. На рис. 5 представлен пример создания графа в системе Y Project.

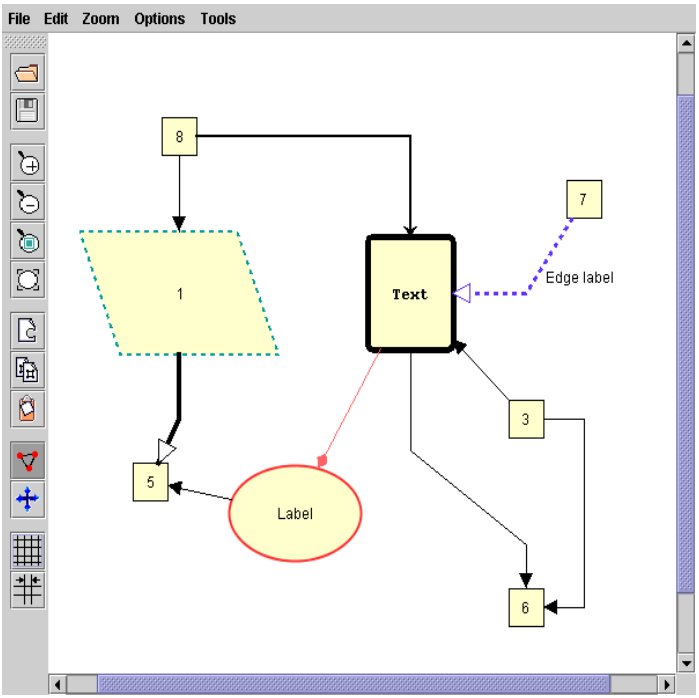

Рис. 5. Окно системы Y Project

7. СИСТЕМА VGJ

VGJ разрабатывается в Auburn University, USA и представляет собой типичный пример написания графового редактора на языке Java.

Систему можно оценить как достаточно примитивный графовый редактор с плохо продуманным пользовательским интерфейсом и слабыми усилиями для получения качественного графического изображения графа.

В данный момент проект находится в незавершенном состоянии (не работают многие необходимые интерфейсные функции, например перемещение сгибов дуг с помощью мыши).

Для хранения графов используется формат GML, однако реальная совместимость с системой Graphlet, для которой этот формат базовый, остается под вопросом. После перенесения записанного графа в Graphlet, топологическая структура графа не изменяется, но изображение существенно искажается. С другой стороны, нельзя сказать определенно, какая из двух систем несет большую ответственность за это искажение.

Графические возможности VGJ включают только минимальный необходимый набор: одноцветные вершины в форме прямоугольников или эллипсов и дуги четырех цветов в виде ломаных линий. Для каждой вершины и дуги предусмотрено по одной текстовой метке. Метка вершины располагается либо под вершиной, либо в ее центре. Шрифт для всех меток один и тот же.

Можно отметить две оригинальные возможности системы:

1) возможность построения трехмерных изображений (граф даже можно вращать в пространстве, выбирая позицию обзора с помощью мыши. Впрочем, изображение графа всегда остается двумерным, т.е. вершины не меняют форму и размеры при поворотах);
2) возможность “свернуть” группу вершин и заменить одной вершиной особого вида.

Группы могут быть вложенными, их можно открывать и закрывать, но больше ничего с ними делать нельзя. Открытая группа никак не отмечается. В GML эти группы записываются как отдельные, ни с чем не соединенные вершины, что выглядит весьма странно.

Недостатки системы в основном происходят из непродуманности тех или иных моментов. Например, force-метод, реализованный в системе, совершенно не учитывает сгибы дуг и размеры вершин; при выборе пункта меню "Edit | Remove all edge bends" уничтожаются все сгибы, включая сгибы в петлях. Подобных примеров можно привести достаточно много.

Возможности системы ограничены и с точки зрения топологии объекта. В частности, в системе нельзя создавать мультиграфы. При добавлении второй дуги между двумя вершинами первая уничтожается.

Наиболее слабым моментом является интерфейс. Неоправданно большое количество режимов и "неочевидных" операций, в которых используются все три кнопки мыши и клавиши Shift и Control. К счастью, все операции со средней кнопкой мыши продублированы другими вариантами.

8. БИБЛИОТЕКА LEDA

Библиотека LEDA (Library of Efficient Data Types and Algorithms) [19, 20] разрабатывается совместно несколькими научными заведениями Германии с 1988-го г. и предназначена для создания программ для дискретной математики. В таких программах часто требуются различные структуры данных типа стеков, очередей, словарей, последовательностей и т.п. Не последнее место в этом списке отведено графам.

LEDA представляет собой набор классов и шаблонов классов на языке C++. Каждый класс моделирует некоторую структуру данных. Кроме того, существует много возможностей по образованию новых классов. Функции-компоненты классов реализуют алгоритмы, наиболее часто используемые для данного типа данных. Вводятся также удобные макроопределения, позволяющие расширять синтаксис C++ конструкциями типа forall.

Библиотека хорошо документирована. Для каждого класса приводится полная спецификация, включая метод реализации. Для алгоритмов даются теоретические оценки сложности.

В LEDA входят классы, предназначенные для визуализации графов, хотя, судя по демонстрационным примерам, это решение не претендует на универсальность.

Программы, использующие LEDA, можно компилировать для UNIX-платформ и для MS Windows 95/NT. LEDA распространяется как в виде исходных текстовых файлов на C++, так и в виде .lib-файлов для большинства современных компиляторов.

9. БИБЛИОТЕКА AGD

AGD [25] является расширением LEDA, предназначенным для рисования графов. Библиотека содержит достаточно много известных алгоритмов рисования, а также предоставляет удобные возможности для программирования новых алгоритмов.

В библиотеке описан один базовый класс, содержащий общие интерфейсные функции алгоритма рисования. Для написания новых алгоритмов создаются классы, производные от данного, и подменяются виртуальные функции. Ряд функций класса алгоритма обеспечивает потенциальную возможность пошаговой визуализации, если это предусмотрено самим алгоритмом.

Данная библиотека может использоваться программами, создаваемыми для различных платформ, в том числе Windows 95/NT. Под Windows можно использовать компилятор Borland 5.x или MS Visual C++ 6.x. Процесс инсталляции библиотеки достаточно прост.

В настоящий момент AGD находится в стадии развития. Следующие версии, как сообщают авторы, будут содержать больше алгоритмов рисования графов. Библиотека снабжена достаточно подробным описанием на 134 страницах и имеет все шансы оказаться полезной на практике.

10. БИБЛИОТЕКА FFGRAPH

Библиотека ffGraph [24], разработанная в University Passau в Германии, представляет собой библиотеку классов на C++, предназначенную для создания и изображения двумерных и трехмерных графов.

Основная цель библиотеки — создать удобный интерфейс для работы с графом, обеспечить возможность привязывать к элементам графа семантическую информацию, а также механизм визуализации.

Метки графов, вершин и дуг создаются как новые классы, которые могут помимо данных содержать также функции, позволяющие им взаимодействовать с другими элементами графа.

В библиотеку включено два алгоритма рисования [38] и трехмерный вариант force-метода [32].

К библиотеке прилагается небольшой редактор графов, работающий под Xwindow и демонстрирующий возможности библиотеки.

Подробная документация к библиотеке имеется в форматах postscript и HTML.

11. GRAPH DRAWING SERVER

Авторы Graph Drawing Server [35, 36] утверждают, что они разработали новую технологию, которую можно использовать для различных задач, связанных с анимацией алгоритмов на графах. Для демонстрации этой технологии, носящей название *Mocha*, был создан Graph Drawing Server.

Суть технологии заключается в том, что на компьютере пользователя запускается Java-апплет, который позволяет редактировать граф. Изменения, производимые в графе, передаются на сервер, где обрабатываются и где по ним генерируется дополнительная информация, которая передается обратно клиенту, транслируясь в некоторые действия по анимации. Таким

способом, например, можно вычислять выпуклую оболочку точек на плоскости. Пользователь вводит новые точки, а сервер считает их оболочку. При этом пользователь постоянно видит динамически обновляющуюся ломаную, ограничивающую выпуклую оболочку. Преимущество такого подхода заключается в том, что в качестве сервера может использоваться достаточно мощный компьютер. Таким образом любой пользователь с помощью Интернет сможет запускать и анимировать сложные графовые алгоритмы.

Строго говоря, Graph Drawing Server не является системой визуализации графов, однако заслужил здесь упоминание как средство схожей направленности.

12. GRAPH LAYOUT TOOLKIT И GRAPH EDITOR TOOLKIT

Graph Layout Toolkit и Graph Editor Toolkit (GLT и GET) [21] — продукты фирмы Tom Sawyer Software [23] — единственные полностью коммерческие разработки в области универсальных средств визуализации графов.

GLT представляет собой библиотеку на C++, включающую компоненты рисования графов, предназначенные для интеграции в любые приложения, использующие графический интерфейс.

В библиотеке реализовано четыре эффективных алгоритма рисования графов [22], способных размещать на плоскости графы, состоящие из нескольких сотен вершин, всего за несколько секунд. На рис. 6 показан результат работы этих алгоритмов для различных графов. GLT можно интегрировать в программы, разрабатываемые для различных платформ, включая Microsoft Windows, Unix, Apple Macintosh и OS/2.

GET является расширением GLT, предназначенным для визуализации графов в приложениях MS Windows, и построен на основе библиотеки MFC (Microsoft Foundation Classes).

Так как и GLT, и GET являются коммерческими разработками, они доступны только за определенную плату даже для использования в некоммерческих целях. Однако фирма Tom Sowyer Software бесплатно распространяет систему, работающую под MS Windows, которая создана специально для демонстрации возможностей GLT и GET. Кроме того, существуют варианты GLT и GET и демонстрационной системы на Java, а также набор ActiveX компонент, предоставляющий удобный API для интеграции в любые программы, работающие под Windows (рис. 7).

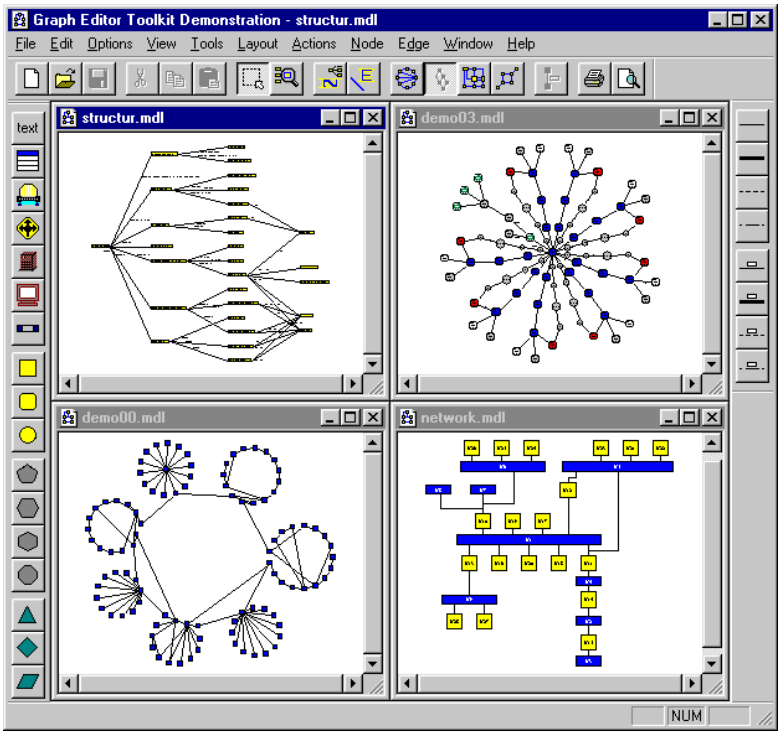

Рис. 6. Четыре стиля автоматического расположения графов в библиотеке GLT

Оставшаяся часть этого пункта посвящена описанию системы, демонстрирующей возможности библиотек. Система представляет собой универсальный графовый редактор с некоторыми дополнениями, ориентированными на работу с графами, представляющими компьютерные сети. Из описания не ясно, является ли эта ориентация базовой для самих библиотек или она возникает как пример их адаптации к конкретной задаче.

Система имеет удобный интерфейс и сравнительно хорошие графические возможности, а также полноценную help-поддержку.

Для редактирования графа используются четыре основных режима: select, zoom, create nodes, create edges и два дополнительных: create bends и delete bends. Кроме основной панели инструментов в системе есть еще две: одна для выбора стиля новой вершины, другая для выбора стиля новой дуги.

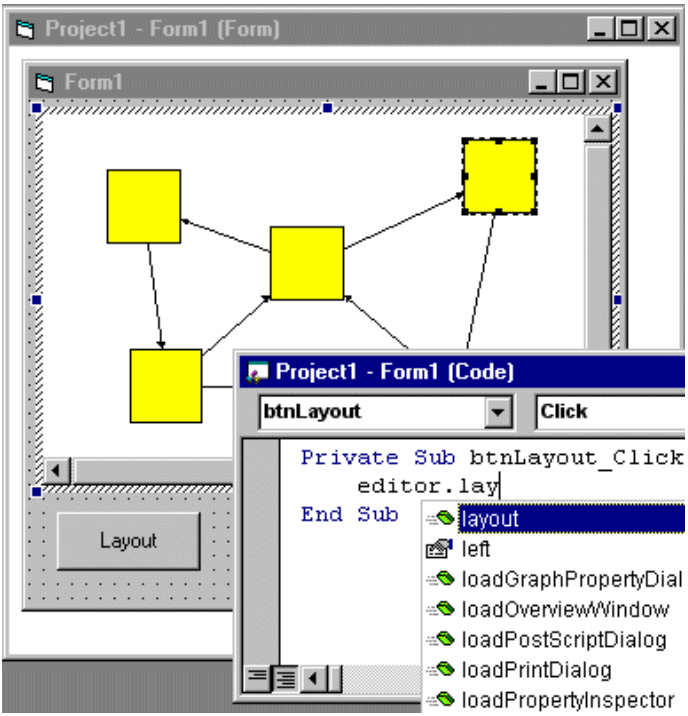

Рис. 7. Использование Graph Editor Toolkit for ActiveX

Вершины бывают пяти видов:

1) текстовые — автоматически подстраиваются под размер текста, рисуются прямоугольниками;
2) табличные — представляют собой таблицы с одной строкой заголовка и несколькими строками текста в несколько линий и рядов, число которых можно изменять; текст каждого элемента вводится отдельно;
3) вершины с изображениями — содержат картинку (нужно указать файл) и подпись под ней; размеры не меняются;
4) bus — изображаются в виде горизонтальной полосы, размеры которой можно менять; особенность в том, что дуги, входящие снизу и сверху, направлены не к центру вершины, а перпендикулярно ее границе; текст не выводится;

5) обычные вершины — имеют форму, выбираемую из 10 вариантов; можно изменять размеры, цвет внутренности и шрифт текста.

Для дуг задаются стиль линии (из 9 вариантов), ее толщина (1, 2, 3) и цвет. Кроме того, каждая дуга может иметь несколько текстовых меток. При этом метки можно произвольно перемещать отдельно от дуг, но каждая метка имеет несколько атрибутов, которые учитываются алгоритмами рисования.

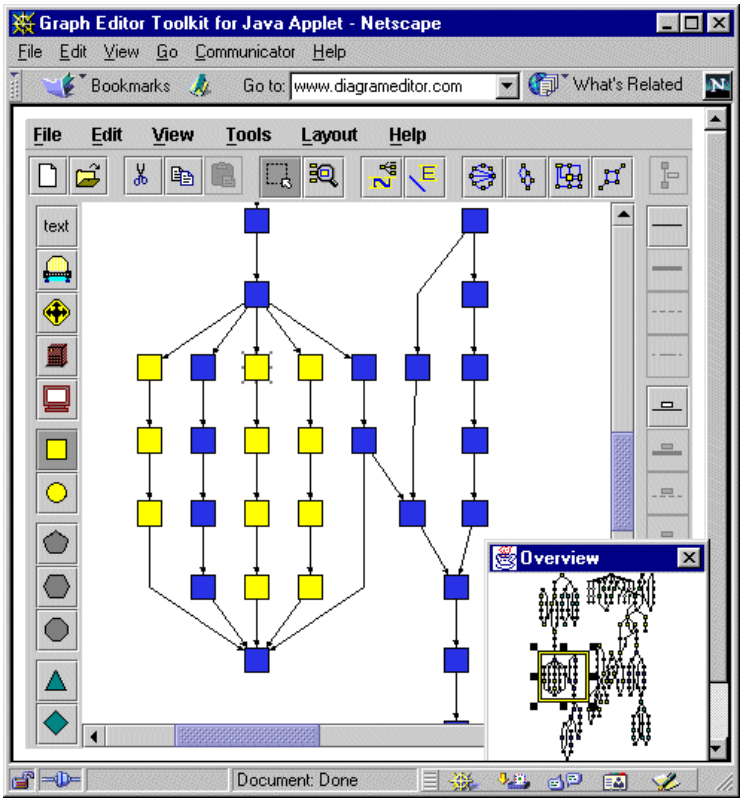

Рис. 8. Использование обзорного окна для навигации

В системе есть возможность произвольного масштабирования изображения графа. Для лучшей ориентации в структуре графа можно открыть обзорное окно (рис. 8). Существует также возможность «сворачивания» отдельных фрагментов графа, т.е. временной замены их псевдовершинами

для обеспечения лучшей видимости структуры графа (рис. 9). Вершины графа также могут содержать другие графы внутри себя (рис. 10).

Параметры вершин, дуг и меток дуг редактируются в специальном окне "Object Properties". При этом можно изменять параметры сразу нескольких выделенных объектов.

В системе реализованы операции cut/copy/paste для групп объектов, однако операции undo и redo отсутствуют.

Четыре алгоритма рисования, представленные в GLT, присутствуют в системе как четыре стиля. В зависимости от выбранного в данный момент стиля изменяется специфика редактирования графа, например: если выбран стиль orthogonal, то конечные звенья дуг всегда входят в вершины под прямым углом. Во всех других стилях они ориентируются на центр вершины.

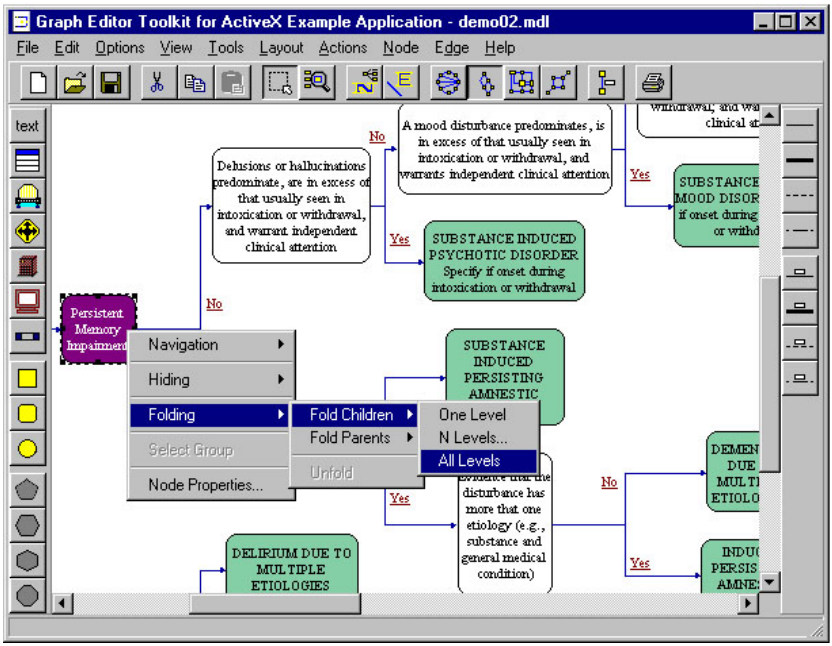

Рис. 9. Применение механизма упрощения диаграмм путем «сворачивания» фрагментов

Все четыре алгоритма можно запускать на произвольном графе. При запуске алгоритма соответственно меняется текущий стиль. Каждый алго-

ритм имеет некоторое множество параметров, которые можно регулировать. Существует специальный алгоритм для размещения меток дуг.

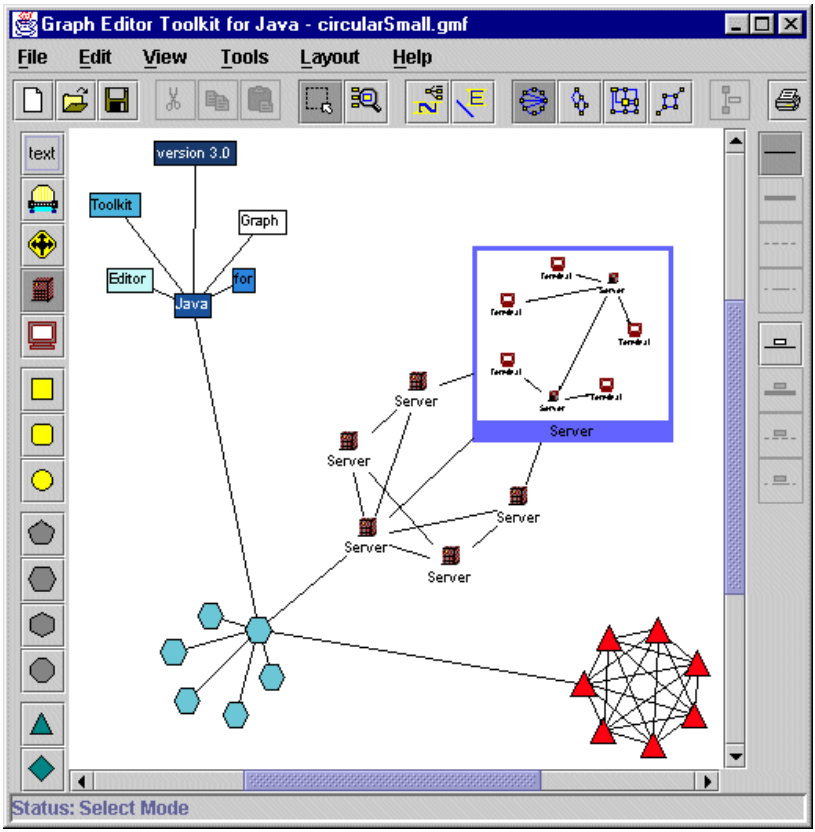

Рис. 10. Пример, показывающий, что в системах, построенных на основе библиотек vGET и GLT, вершины графа также могут быть графами

13. СИСТЕМА SMARTDRAW

Системы визуализации графовых объектов, как правило, ориентированы на применение в профессиональной среде, т.е. их пользователи — программисты и математики, изучающие те или иные структуры данных или

алгоритмы их обработки. Однако с необходимостью построения изображений графов и похожих на них объектов сталкиваются и рядовые пользователи. Многие книги, статьи и другие издаваемые материалы часто содержат иллюстрации в виде диаграмм, по сути дела являющихся графами. Особенно это характерно для научных публикаций.

Необходимостью быстро и удобно создавать такого рода иллюстрации обусловлено существование класса систем графовых редакторов, нацеленных на непрофессионального пользователя. Типичным представителем этого класса является система SmartDraw [40]. На рис. 11 приведено окно системы с примером редактируемой диаграммы.

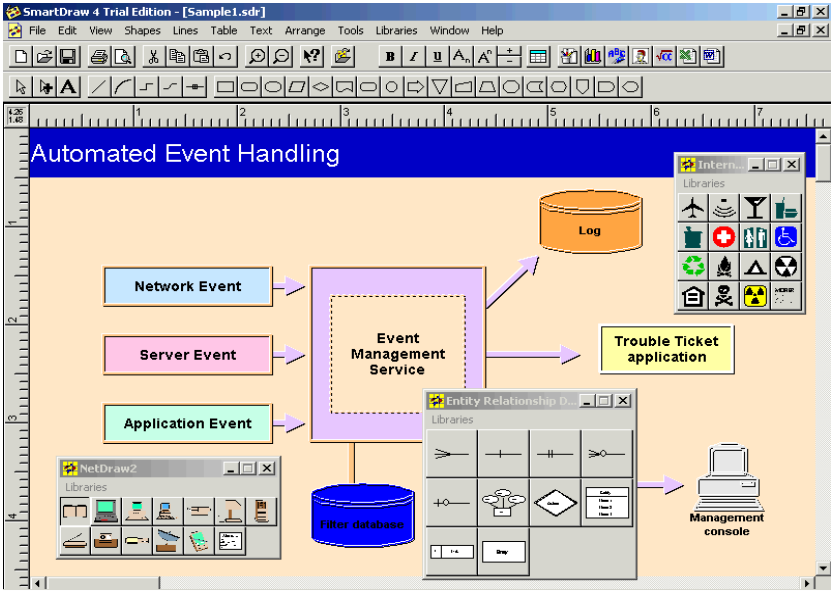

Рис. 11. Система SmartDraw

Следует отметить, что для систем этого класса понятие графа как математического объекта отсутствует. Если в математике графы являются удобным механизмом представления структур данных, то здесь они возникают как удобный механизм представления изображения, позволяющий редактировать его более удобным способом, чем в графическом редакторе. С другой стороны, SmartDraw это и есть наполовину графический редактор, наполовину система визуализации графов.

В SmartDraw диаграммы составляются из двух видов объектов — форм и линий, что соответствует вершинам и дугам в графе. В то же время каждая форма сама задается набором линий, из которых состоит ее контур. Формы, как правило, соединяются друг с другом линиями, используя так называемые *точки присоединения*. На линиях и контурах форм задается некоторое число таких точек. Если нужно соединить вершину и форму, то следует совместить какую-либо точку соединения на линии с какой-либо точкой соединения на нужной форме. Такой механизм соединения весьма удобен по той же причине, что и прямоугольная сетка для расположения объектов. Оба механизма позволяют симметрично располагать объекты без дополнительного выравнивания. Если симметрии не требуется, то для любой линии и формы можно задать свободный режим сцепления. Тогда точки соединения других объектов можно совмещать с любой точкой данной линии или формы.

В системе используются 24 вида линий и неограниченное количество форм, составленных из этих линий. Формы группируются в библиотеки, позволяющие выбирать и создавать новые формы и целые подграфы. Для каждой линии и формы отдельно задаются все необходимые атрибуты, такие как цвет, толщина линии и т. д. Допускается также размещение отдельно созданных изображений внутри форм и вставка в рисунок надписей. На рис. 12 приведены примеры диаграмм, созданных с помощью системы SmartDraw.

Система SmartDraw предоставляет также ряд дополнительных возможностей, полезных при подготовке иллюстраций, например выбор стиля всего изображения, позволяющий создавать выпуклые формы (см. рис. 11). SmartDraw отлично документирована и имеет систему подсказок, позволяющих изучить работу за очень короткое время.

14. СИСТЕМА HIGRES

Система Higres [41, 42] создавалась автором настоящего обзора в 1996—1999 гг. Данная система предназначена для создания и визуализации иерархических графовых моделей, представляющих собой иерархические графы с заданной семантикой.

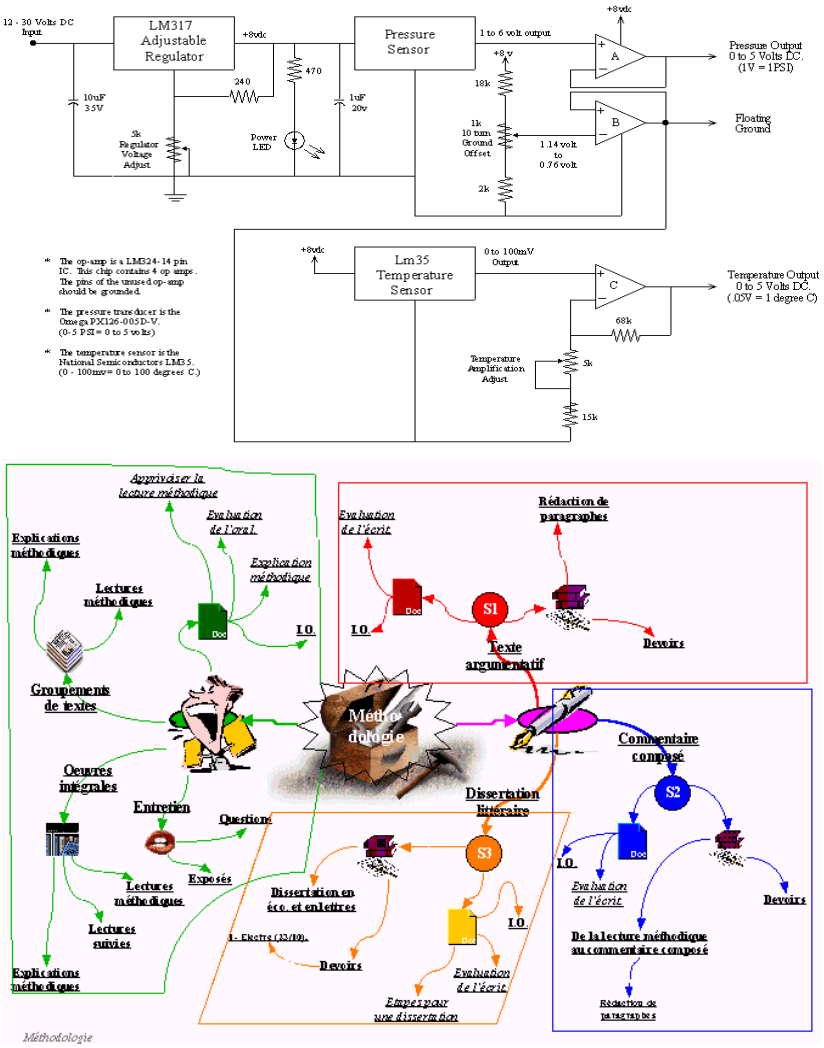

Рис. 12. Примеры изображений графов, построенных с помощью системы SmartDraw

Иерархический граф состоит из вершин, дуг и фрагментов. Вершины и дуги представляют собой обычный граф, который может быть ориентиро-

ванным или неориентированным. Каждый фрагмент ассоциируется с набором вершин, которые ему принадлежат. Фрагменты могут быть вложенными, если набор вершин одного является подмножеством набора вершин другого. Иначе пересекаться они не могут. Набор всех вершин определяет *главный* фрагмент иерархического графа. Более точное определение иерархических графов и других понятий, связанных с ними, можно найти в [2]. Различным вопросам визуализации иерархических графов посвящены работы [4—7]. Пример иерархического графа, созданного в системе Higres, приведен на рис. 13. Такие графы используются во внутреннем представлении IF-1 языка SISAL [3].

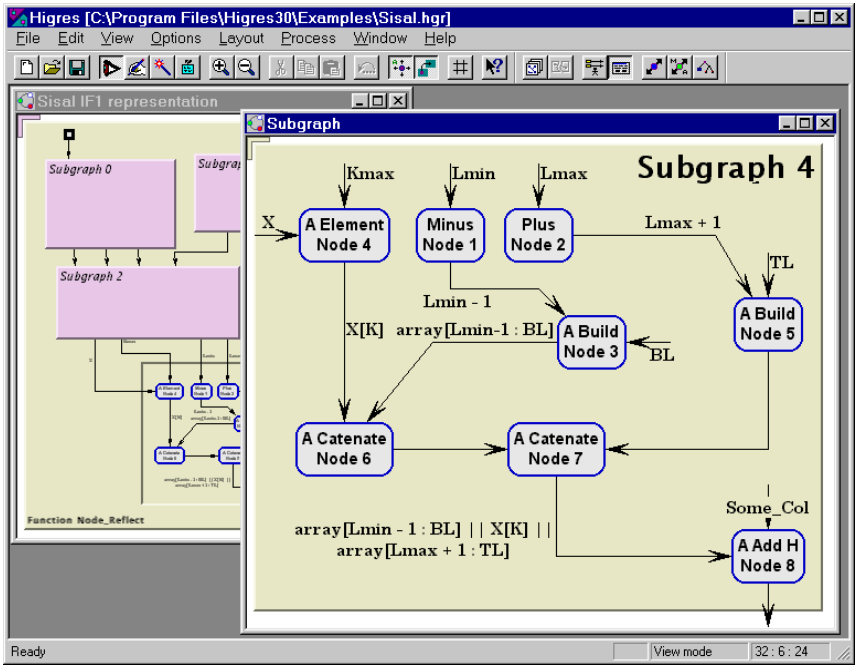

Рис. 13. Иерархический граф в системе Higres

Семантика иерархического графа представляется в системе Higres с помощью типов объектов. Каждый объект в графе принадлежит к какому-нибудь типу. Для каждого типа определяется набор меток. Каждая метка имеет тип данных, имя и несколько других параметров. Набор значений ассоциируется с каждым объектом графа в соответствии с набором меток,

определенным для типа этого объекта. Вместе с разделением на типы эти значения задают семантику графа. Таким образом, пользователь может определять семантику, добавляя новые типы и их метки.

В системе Higres каждый фрагмент визуально задается прямоугольником. Все вершины, принадлежащие фрагменту, располагаются внутри этого прямоугольника. Фрагменты так же, как и вершины, никогда не накладываются друг на друга. Каждый фрагмент может быть либо *закрытым*, либо *открытым*. В первом случае содержимое фрагмента скрыто от пользователя, во втором — видно внутри прямоугольника, представляющего данный фрагмент. Для каждого фрагмента можно открыть отдельное окно, в котором будет видно содержимое только этого фрагмента и его открытых подфрагментов.

Большая часть атрибутов объекта определяется его типом. Следовательно, семантически близкие объекты имеют сходное визуальное представление. Для визуализации меток объекта пользователь задает некоторый шаблон текста, в который вставляются значения меток каждого конкретного объекта.

Дополнительные возможности визуализации включают:

- различные формы и стили вершин;
- дуги в виде ломаных линий и гладких кривых;
- различные стили линий и стрелок дуг;
- выбор цвета для всех элементов графа;
- возможность задавать произвольный масштаб изображения;
- возможность перемещать текст дуги вдоль ее линии;
- позиционирование внешнего текста вершины в любом месте около ее границы;
- выбор шрифта для всех элементов графа;
- два формата графического вывода;
- широкий набор опций визуализации.

В Higres встроено три алгоритма размещения графов на плоскости. Первый — известный метод сил, второй — несколько улучшенная версия первого, третий — алгоритм размещения деревьев с помощью уровней. Все алгоритмы реализованы во внешних модулях, прилагаемых к системе.

Одним из достоинств системы является удобный интуитивный пользовательский интерфейс. Главное окно системы содержит меню, несколько панелей инструментов и панель состояния. Внутри главного окна можно открывать произвольное количество окон фрагментов, используя интерфейс MDI.

При работе с системой используются два основных режима: просмотра и редактирования. В первом из них можно просматривать граф, открывая и закрывая фрагменты и их окна, а также скроллируя граф с помощью линеек прокрутки или мыши.

В режиме редактирования левая кнопка мыши используется для выделения объектов. Нажатие правой кнопки вызывает контекстное меню, в котором можно выбрать команду для исполнения над выделенными объектами или над всем графом. С помощью этого меню можно также создавать новые вершины, фрагменты и дуги (рис. 14).

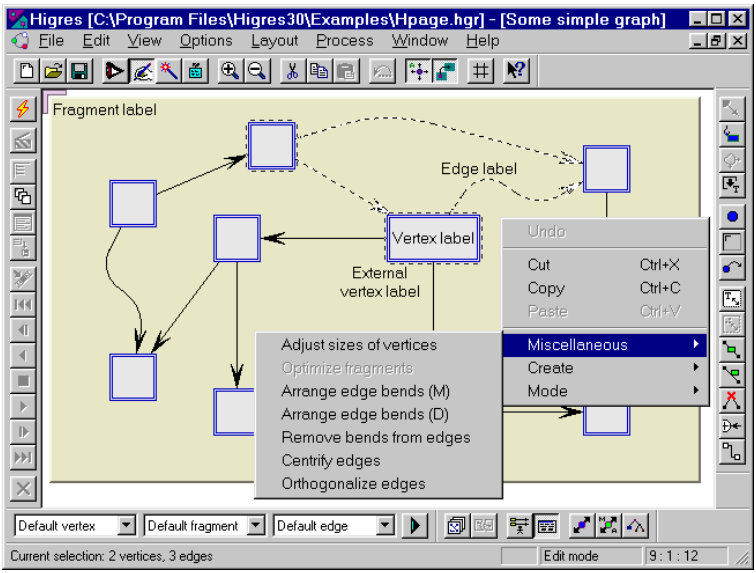

Рис. 14. Редактирование графа в системе Higres

С помощью левой кнопки мыши можно перемещать выделенные группы объектов. Таким образом, все операции редактирования собраны в одном режиме. В системе предусмотрены два дополнительных режима, позволяющие в некоторых случаях облегчить процесс редактирования: режим создания объектов и режим редактирования меток.

Дополнительные интерфейсные возможности включают:

- операции cut/copy/paste;
- практически неограниченное количество уровней undo;
- оптимизированный вывод изображения на экран;

- автоматическое устранение наложений объектов;
- автоматическую подстройку размеров вершин;
- прямоугольную сетку;
- опции, позволяющие настраивать пользовательский интерфейс;
- справочную систему для каждого пункта меню, блока диалога и режима редактирования.

В системе существует возможность расширения путем добавления новых модулей двух типов: во-первых, содержащих алгоритмы размещения графов на плоскости, и во-вторых, таких, с помощью которых можно производить семантическую визуальную обработку графовых моделей, т.е. выполнять любые алгоритмы на графах, визуально наблюдая результат каждого шага алгоритма.

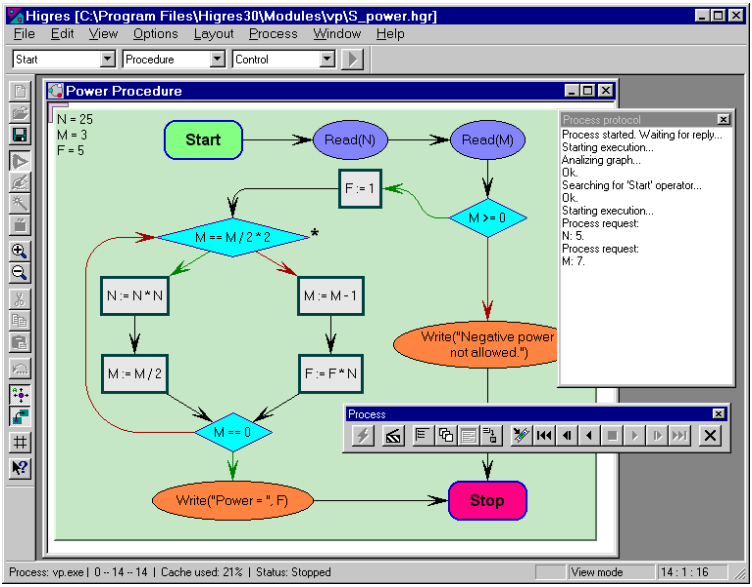

Рис. 15. Выполнение алгоритма, реализованного в виде внешнего модуля к системе Higres

Для запуска алгоритма в системе пользователь должен выбрать соответствующий внешний модуль. Результат работы модуля можно наблюдать непосредственно во время его работы. Кроме того, в системе предусмотре-

на возможность «промотки назад» и пошагового выполнения. В отдельном окне показывается протокол исполнения алгоритма и его текстовый вывод. На рис. 15 показан пример исполнения внешнего модуля. В данном случае граф представляет собой схему программы быстрого возведения в степень, а внешний модуль интерпретирует эту схему, т.е. выполняет программу по шагам, выводя результат в специальное окно.

Анимация алгоритмов может быть использована для их тестирования и отладки, образовательных целей, а также для изучения итеративных процесов, возникающих, например, в некоторых методах рисования графов.

К системе прилагается специальная библиотека на языке C++, предназначенная для написания внешних модулей и включающая функции для модификации графа, а также для взаимодействия с системой. Для написания внешних модулей с помощью данной библиотеки необязательно знать детали внутреннего представления графа, что облегчает работу с ней.

15. ЗАКЛЮЧЕНИЕ

Проведенный обзор систем визуализации графов показывает, что существует достаточно широкий спектр инструментов визуализации, причем, по всей видимости, проблема визуализации настолько разнообразна, что на сегодняшний день никто не ставит задачу создания универсального средства для этих целей. Каждая система имеет свою специализацию, даже если она не является узкоспециализированной в обычном понимании этого слова. Так, например, система VCG ориентирована на визуализацию графов, получаемых в компиляторах, Graphlet — на испытание алгоритмов расположения графов, Higres — на визуальную обработку иерархических графовых моделей, SmartDraw — на подготовку иллюстраций.

Кроме того, многие системы имеют ограничения на структуру графа либо ориентацию на определенный тип графа. Так, например, daVinci фактически работает только с ациклическими графами, Hirges ориентирован на иерархические графы.

В заключение отметим ряд открытых проблем в данной области и дадим некоторый комментарий по этому поводу.

Почти все универсальные системы и библиотеки созданы в университетской среде. Это обстоятельство накладывает определенный отпечаток на сущность данных систем или, вернее сказать, позволяет проследить некоторые тенденции в расстановке приоритетов, учитывавшихся при их создании. Фактически можно сказать, что процесс создания большинства систем

носил несколько стихийный характер и практические цели не всегда учитывались.

Академический подход к проблеме породил дисбаланс между количеством разработанных методов визуализации графов и созданными инструментальными средствами. Существует достаточно много хороших алгоритмов рисования графов, которые до сих пор реализованы только в экспериментальных системах, созданных для тестирования этих алгоритмов, либо вообще нигде не реализованы.

Другой стороной проблемы является ориентация почти всех рассмотренных систем на работу под операционной системой UNIX. При этом традиционно заявление: "наша система легко переносима под MS Windows" или "наша система уже перенесена под MS Windows". В первом случае обычно оказывается, что осуществить перенос на практике совсем не легко, во втором, что после такого переноса система теряет ряд возможностей, набирает дополнительные ошибки, а главное, продолжает выглядеть как стандартная XWindow-система с характерным интерфейсом. Пользователь MS Windows, привыкший к совершенно другим стандартам, не станет без крайней необходимости использовать такую систему. Таким образом, переносимость системы в данном случае есть только повод вписать лишнюю строчку в список ее достоинств.

Операционная система UNIX является базовой для подавляющего большинства западных университетов, поэтому нет ничего удивительного в том, что на нее ориентированы все их разработки. Однако абстрактный пользователь системы визуализации графов вовсе не обязан принадлежать к университетской среде. Более того, тот факт, что фирма Tom Sawyer Software, будучи коммерческой организацией, разрабатывает свои продукты в первую очередь для MS Windows, показывает, что ориентацию на UNIX можно записать только в недостатки. К сказанному можно лишь добавить, что, по мнению автора, операционные системы MS Windows по всем параметрам более пригодны для целей визуализации, чем любой вариант XWindow, работающий на том же оборудовании.

Если говорить о конкретных недостатках существующих систем визуализации графов, то можно выделить следующие:

1. Недостаточные графические возможности, не позволяющие получать качественные изображения графов. Сюда входит и ограниченный набор визуальных параметров элементов графа, и грубые вычислительные методы, приводящие к искажениям изображения.
2. Неудобный интерфейс графовых редакторов: интенсивное использование режимов, текстовое задание параметров элементов графа,

отсутствие функций undo и redo, общая медлительность интерфейса, особенно сильно сказывающаяся при манипуляциях с мышью и т.д.

3. Многие системы не имеют встроенной help-поддержки, а в некоторых случаях документация вообще отсутствует.
4. Нестабильная работа доступных версий.

Конечно, нельзя сказать, что все эти недостатки присущи каждой системе, но вполне можно утверждать, что каждая система имеет свой набор недостатков из этого списка.

Из всех средств визуализации графов достаточно широкое распространение в мире получили только daVinci, LEDA, продукция Tom Sawyer и SmartDraw. При этом система daVinci к настоящему времени несколько устарела (хотя работы по ней, прерванные в 1995 г., недавно возобновились), кроме того, данная система не является универсальной по типу изображения графа, поддерживая только уровневое представление. Библиотека LEDA представляет хорошую основу для построения графовых моделей и алгоритмов, но не содержит хороших универсальных компонентов визуализации.

Таким образом, на сегодняшний день для решения задачи визуализации графов, как в коммерческих, так и в университетских проектах, в большинстве случаев либо создается собственная система визуализации, что ведет к большим дополнительным затратам времени и средств, либо используются компоненты Graph Layout Toolkit и Graph Editor Toolkit, поставляемые фирмой Tom Sawyer Software. Для создания рисунков диаграмм используется в основном система SmartDraw или ее аналоги.

СПИСОК ЛИТЕРАТУРЫ

1. **Касьянов В.Н.** Оптимизирующие преобразования программ. — М: Наука, 1988.
2. **Касьянов В.Н.** Иерархические графы и графовые модели: вопросы визуальной обработки // Проблемы систем информатики и программирования. — Новосибирск, 1999. — С. 7—32.
3. **Густокашина Ю.В., Евстигнеев В.А.** IF1 — промежуточное представление SISAL-программ // Проблемы конструирования эффективных и надежных программ. — Новосибирск, 1995. — С. 70—78.
4. **Eades P., Feng Q.W.** Drawing clustered graphs on an orthogonal grid // Proc. of Graph Drawing 97. — Berlin a.o.: Springer Verlag, 1997. — P. 182—193. — (Lect. Notes in Comput. Sci.; Vol. 1353).

5. **Eades P., Feng Q.W.** Multilevel visualization of clustered graphs // Proc. of Graph Drawing 96. — Berlin a.o.: Springer Verlag, 1996. — P. 101—112. — (Lect. Notes in Comput. Sci.; Vol. 1190).
6. **Eades P., Feng Q.W., Lin X.** Straight-line drawing algorithms for hierarchical graphs and clustered graphs // Proc. of Graph Drawing 96. — Berlin a.o.: Springer Verlag, 1996. — P. 113—128. — (Lect. Notes in Comput. Sci.; Vol. 1190).
7. **Feng Q.W., Cohen R., Eades P.** How to draw a planar clustered graph // Proc. COCOON '95. — Berlin a.o.: Springer Verlag, 1995. — P. 21—31. — (Lect. Notes in Comput. Sci.; Vol. 959).
8. **Di Battista G., Eades P., Tamassia R., Tollis I.G.** Algorithms for drawing graphs: an annotated bibliography // Comput. Geom. Theory Appl. — 1994. — Vol. 4.
9. **Tamassia R.** Graph drawing // CRC Handbook of Discrete and Computational Geometry / Ed. by J.E. Goodman, J. O'Rourke. — CRC Press, 1997.
10. **Himsolt M.** GraphEd: a graphical platform for the implementation of graph algorithms (extended abstract and demo) // Proc. of Graph Drawing 94. — Berlin a.o.: Springer Verlag, 1994. — P. 182—193. — (Lect. Notes in Comput. Sci.; Vol. 894).
11. **Himsolt M.** The Graphlet system (system demonstration) // Proc. of Graph Drawing 96. — Berlin a.o.: Springer Verlag, 1996. — P. 233—240. — (Lect. Notes in Comput. Sci.; Vol. 1190).
12. **Система** Graphlet доступна по адресу: http://www.fmi.uni-passau.de/Graphlet/
13. **Sander G.** Graph layout through the VCG tool // Proc. of Graph Drawing 94. — Berlin a.o.: Springer Verlag, 1994. — P. 194—205. — (Lect. Notes in Comput. Sci.; Vol. 894).
14. **Система** VCG доступна по адресу: http://www.cs.uni-sb.de/RW/users/sander/html/gsvcg1.html
15. **Frohlich M., Werner M.** Demonstration of the interactive graph visualization system daVinci // Proc. of Graph Drawing 94. — Berlin a.o.: Springer Verlag, 1994. — P. 266—269. — (Lect. Notes in Comput. Sci.; Vol. 894).
16. **Система** daVinci доступна по адресу: http://www.informatik.uni-bremen.de/~inform/forschung/daVinci/daVinci.html
17. **Lauer H., Ettrich M., Soukup K.** GraVis — system demonstration // Proc. of Graph Drawing 97. — Berlin a.o.: Springer Verlag, 1997. — P. 344—349. — (Lect. Notes in Comput. Sci.; Vol. 1353).
18. **Система** GraVis доступна по адресу: http://www-pr.informatik.uni-tuebingen.de/GraVis/index.html/
19. **Mehlhorn K., Naher S.** LEDA: A platform for combinatorial and geometric computing // Communs. ACM. — 1995. — Vol. 38. — P. 96—102.
20. **Библиотека** LEDA доступна по адресу: http://www.mpi-sb.mpg.de/LEDA/

21. **Madden B., Madden P., Powers S., Himsolt M.** Portable graph layout and editing // Proc. of Graph Drawing 95. — Berlin a.o.: Springer Verlag, 1995. — P. 385—395. — (Lect. Notes in Comput. Sci.; Vol. 1027).
22. **Dogrusoz U., Madden B., Madden P.** Circular layout in the graph layout toolkit // Proc. of Graph Drawing 96. — Berlin a.o.: Springer Verlag, 1996. — P. 92—100. — (Lect. Notes in Comput. Sci.; Vol. 1190).
23. **Tom** Sawyer Software, 804 Hearst Avenue, Berkeley, CA 94710, http://www.tomsawyer.com
24. **Библиотека** ffGraph доступна по адресу: http://www.fmi.uni-passau.de/~friedric/ffgraph/main.shtml
25. **Библиотека** AGD доступна по адресу: http://www.mpi-sb.mpg.de/AGD/
26. **Система** VGJ доступна по адресу: http://www.eng.auburn.edu/department/cse/research/graph_drawing/graph_drawing.htm
27. **Barghouti N., Mocenigo J., Lee W.** Grappa: a graph package in Java // Proc. of Graph Drawing 97. — Berlin a.o.: Springer Verlag, 1997. — P. 336—343. — (Lect. Notes in Comput. Sci.; Vol. 1353).
28. **Bruss I., Frick A.** Fast Interactive 3-D Graph visualization // Proc. of Graph Drawing 95. — Berlin a.o.: Springer Verlag, 1995. — P. 99—110. — (Lect. Notes in Comput. Sci.; Vol. 1027).
29. **Patrignani M., Vargui F.** 3DCube: a tool for three dimentional graph drawing // Proc. of Graph Drawing 97. — Berlin a.o.: Springer Verlag, 1997. — P. 284—290. — (Lect. Notes in Comput. Sci.; Vol. 1353).
30. **Tunkelang D., Byrd R., Cooper J.** Lexical navigation: using incremental graph drawing for query refinement // Proc. of Graph Drawing 97. — Berlin a.o.: Springer Verlag, 1997. — P. 316—321. — (Lect. Notes in Comput. Sci.; Vol. 1353).
31. **Eades P., Cohen R., Huang M.L.** Online animated graph drawing for Web navigation // Proc. of Graph Drawing 97. — Berlin a.o.: Springer Verlag, 1997. — P. 332—335. — (Lect. Notes in Comput. Sci.; Vol. 1353).
32. **Eades P.** A heuristic for graph drawing // Congressus Numerantium. — 1984. — N42. — P. 149—160.
33. http://www.cs.uni-sb.de/RW/users/sander/html/gstools.html
34. http://www.uni-passau.de/~himsolt/graphdrawing.html
35. **Bridgeman S., Garg A., Tamassia R.** A graph drawing and translation service on the WWW // Proc. of Graph Drawing 96. — Berlin a.o.: Springer Verlag, 1996. — P. 45—52. — (Lect. Notes in Comput. Sci.; Vol. 1190).
36. **Graph** Drawing Server находится по адресу: http://loki.cs.brown.edu:8081/graphserver/home.shtml
37. **Garg A., Tamassia R.** GIOTTO3D: a system for visualizing hierarchical structures in 3D // Proc. of Graph Drawing 96. — Berlin a.o.: Springer Verlag, 1996. — P. 193—200. — (Lect. Notes in Comput. Sci.; Vol. 1190).

38. **Sugiyama K., Tagawa S., Toda M.** Methods for visual understanding of hierarchical systems // IEEE Trans. Syst. Man Cybern. — 1981. —Vol. SMC-11, N 2. — P. 109—125.
39. **Система** Y Project доступна по адресу: http://www-pr.informatik.uni-tuebingen.de/yfiles/
40. **Система** SmartDraw доступна по адресу: http://www.smartdraw.com
41. **Lisitsyn I.A., Kasyanov V.N.** Higres — visualization system for clustered graphs and graph algorithms // Proc. of Graph Drawing 99. — Berlin a.o.: Springer Verlag, 1999. — P. 82—89. — (Lect. Notes in Comput. Sci.; Vol. 1731).
42. **Система** Higres доступна по адресу: http://lis.iis.nsk.su/higres.

И. А. Лисицын

СИСТЕМЫ ВИЗУАЛИЗАЦИИ И РЕДАКТИРОВАНИЯ ГРАФОВЫХ ОБЪЕКТОВ

ОБЗОР

Препринт
76

Рукопись поступила в редакцию 24.06.2000
Рецензент В. А. Евстигнеев
Редактор З. В. Скок

Подписано в печать 27.11.2000
Формат бумаги 60 × 84 1/16 Объем 2.4 уч.-изд.л., 2.6 п.л.
Тираж 50 экз.

НФ ООО ИПО “Эмари” РИЦ, 630090, г. Новосибирск, пр. Акад. Лаврентьева, 6