УДК 681.3

ОБ ОДНОМ ПОДХОДЕ К ОТОБРАЖЕНИЮ ВЕКТОРНОЙ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ В СРЕДЕ ИНТЕРНЕТ-БРАУЗЕРА

Н. А. Богомолов[1], А. Д. Ковалев[1], М. Н. Синицын[1]

Рассматривается подход к публикации в Интернет больших объемов числовой информации в графической форме. Приведено описание архитектуры разработанного авторами программного обеспечения (ПО), расширяющего возможности стандартного Интернет-браузера в части отображения графической информации. Включены примеры использования разработанного ПО, иллюстрирующие методику его применения.

Ключевые слова: графическое представление числовой информации, Интернет-браузеры, язык разметки HTML, гипертекстовые документы, растровые записи, векторные записи, визуализация.

1. Введение. В современном научном сообществе Интернет занимает прочные позиции как важное средство коммуникации, обмена научной информацией и публикации научных результатов.

Основным средством представления материалов в Интернет служит язык разметки HTML — гибкий инструмент, позволяющий "увязывать" в форме гипертекстового документа многошрифтовые текстовые фрагменты и растровые изображения. Научные результаты часто содержат много числовых материалов, которые необходимо представлять в графическом виде. Стандартный язык HTML не имеет устоявшихся встроенных средств описания векторных изображений. Поэтому, как правило, для публикации в Интернет научных результатов все графические материалы приходится представлять в растровой форме.

При публикации в Интернет числовых материалов в графической форме растровое представление графики часто становится неэффективным из-за большого объема пересылаемой по сети информации. Суммарный объем графического представления числовых данных может во много раз превосходить объем исходной числовой информации. Больший объем графического представления обусловлен прежде всего тем, что оно включает в себя дополнительные элементы оформления, повышающие наглядность отображения информации. Кроме того, часто для повышения наглядности одна и та же числовая информация может быть отображена в различной форме (например график, гистограмма, таблица значений, картограмма и т.д.). Это еще больше уменьшает отношение объема исходной числовой информации к объему растров, необходимых для ее наглядной визуализации.

Естественный путь снижения объемов пересылаемой по сети информации — это отправка потребителю информации не готовых изображений, а исходных числовых данных, и построение нужных растровых графических представлений непосредственно на компьютере клиента. Для этого необходимо дополнить стандартный Интернет-браузер программным обеспечением (ПО), расширяющим его возможности в части отображения векторной графики.

Надежным, хорошо зарекомендовавшим себя способом расширения функциональных возможностей стандартного Интернет-браузера является создание Java-апплета, который размещается на странице обычного HTML-документа подобно статическому растровому изображению. Однако в отличие от простого растрового изображения, Java-апплет является программой, которая может осуществлять генерацию изображений, в том числе и динамически по командам, подаваемым пользователем, осуществляющим просмотр гипертекстового документа. Использование Java-апплета для отображения графической информации позволяет гибко сочетать средства стандартного HTML по гипертекстовой организации публикуемых материалов с возможностью встраивания в них векторных изображений, в том числе и динамических.

Поэтому при выборе способа реализации ПО поддержки векторной графики в среде стандартного Интернет-браузера авторы остановили свой выбор на Java-апплете.

2. Архитектура разработанного Java-апплета. ПО для отображения векторной графики должно иметь многоуровневую архитектуру. При реализации верхнего уровня ПО, обеспечивающего пользовательский интерфейс и предварительную обработку данных для последующего отображения, целесообразно максимально использовать встроенные в Интернет-браузер языки описания сценариев JavaScript и VBScript.

[1] Научно-исследовательский вычислительный центр, Московский государственный университет им. М. В. Ломоносова, 119992, Москва; e-mail: nbogom@srcc.msu.su, kovalev@srcc.msu.su

Нижний уровень ПО должен обеспечивать то, что невозможно или нецелесообразно осуществлять средствами языков описания сценариев:

— построение изображений на основе “удобного” для широкого класса задач набора базовых графических примитивов;

— отслеживание перемещения курсора в реальном времени с целью создания “интерактивных” изображений, изменяющихся в зависимости от текущего положения курсора без нажатия кнопок мыши;

— динамическое изменение изображений для создания анимационных эффектов.

Созданный авторами Java-апплет MultiView (далее апплет) реализует функции нижнего уровня ПО. Он обеспечивает показ многослойных изображений, состоящих из векторных и растровых элементов. В функцию апплета также входит контроль за перемещением курсора с целью выявления попадания курсора в некоторые заранее заданные области изображения (активные зоны). Множество активных зон (также как и созданное средствами апплета изображение) может иметь многослойную организацию (т.е. активные зоны могут пересекаться и экранировать друг друга). Состав изображения и состав активных зон может динамически изменяться по командам, выдаваемым верхним уровнем ПО, который реализуется на языке JavaScript.

Разработанное ПО имеет следующую объектную структуру. Для хранения изображений служит объект типа слой графических элементов. Для хранения параметров графических элементов и числовых данных, которые могут быть использованы для генерации изображений, служит объект типа массив данных. Массивы данных с элементами типа число или строка предназначены для хранения исходных данных. Массивы данных с элементами такого типа, как битовая шкала, цвет, изображение строки текста и изображение многострочного многошрифтового текста, предназначены для параметризации графических элементов, входящих в состав слоя. Еще одним объектом, который может быть использован для хранения данных, является таблица поименованных объектов.

Для удобства хранения и доступа к перечисленным выше объектам они могут быть организованы в иерархическую структуру с помощью объекта типа группа. Группа представляет собой множество поименованных объектов. Элементом группы может быть слой графических элементов, массив данных, таблица поименованных объектов или группа. Любой объект, входящий в состав одной группы, может входить в состав произвольного числа групп. Изображение, которое апплет выводит на экран, предварительно строится в объекте типа виртуальный экран. Этот виртуальный экран является основным. Можно создать другие, вспомогательные экраны, на которых можно заранее готовить изображения для последующего быстрого показа. При создании вспомогательного виртуального экрана необходимо задать его размеры и имя. Основной виртуальный экран создается автоматически при загрузке апплета.

Подготовленные на виртуальных экранах растровые изображения и их фрагменты можно хранить в специально предназначенных для этого объектах — поименованных растровых изображениях. Динамически созданные средствами апплета поименованные растровые изображения по своему внутреннему устройству и возможностям использования аналогичны заранее подготовленным растровым изображениям в формате GIF, в частности, они могут иметь прозрачные области.

Изображение, которое создается с помощью апплета, является иерархической структурой и состоит из следующих базовых элементов:

— прямоугольников,

— односвязанных полигонов,

— ломаных линий,

— секторов эллипсов,

— фрагментов многошрифтового форматированного текста,

— текстовых строк, размещенных вдоль произвольных ломаных линий,

— растровых изображений.

Отдельные элементы изображения организованы в объекты типа слой. Слой — это множество элементов изображения, обладающих некоторыми параметрами, общими для всех элементов, входящих в состав слоя (например, толщина и цвет границ, цвет фона).

Слои могут быть организованы в объекты типа группа. Группы не содержат параметров отображения и служат лишь для управления составом изображения и последовательностью выводимых на экран слоев.

Один элемент может входить одновременно в произвольное количество слоев, а один слой, в свою очередь, в произвольное количество групп. Такая организация позволяет гибко управлять составом “сложного” (состоящего из сотен и тысяч элементов) изображения, манипулируя относительно “небольшим” количеством слоев и групп слоев.

Подобная же гибкость допускается и при описании объектов типа активная зона изображения —

областей изображения, чувствительных к положению курсора. Апплет следит за перемещением курсора и позволяет получать информацию обо всех активных зонах, в которых в данный момент времени находится курсор. Для задания активных зон также используются элементы изображения. Активной считается область, которую занимает на экране соответствующий элемент изображения. Для задания множества активных зон используются элементы изображения, составляющие слои и группы слоев. Элементы, входящие в состав слоев, принадлежащих одной группе, образуют множество зависимых друг от друга областей изображения. В таком множестве в каждый момент времени может быть активной только одна зона (соответственно, один элемент слоя). Если активные зоны в таком множестве геометрически пересекаются, то всегда однозначно определено, какая зона экранирует другие зоны (т.е. расположена сверху). Элементы, входящие в состав слоев, принадлежащих разным группам, не зависят друг от друга (т.е. их зоны могут быть одновременно активными несмотря на взаимное геометрическое пересечение).

Разработанное ПО, работающее на клиентской машине, обладает следующими основными функциональными возможностями:

— статическое построение изображений по заданным параметрам вызова апплета;

— динамическое перестроение изображения под управлением сценариев на JavaScript;

— чтение с Интернет-сервера специально подготовленных файлов данных, содержащих описание пользовательского интерфейса, бизнес-логику приложения, координаты векторных объектов и другую необходимую для построения изображений информацию;

— реализация на клиентской машине пользовательского интерфейса и бизнес логики в соответствии с полученной от Интернет-сервера информацией.

Разработанный апплет позволяет разместить требуемое изображение на странице HTML-документа без дополнительного программирования на JavaScript. Для размещения апплета на Web-странице необходимо использовать тег APPLET. Параметризация апплета осуществляется с помощью параметров двух типов — общих и индивидуальных. Параметры, общие для любого апплета (задающие размеры изображения, сетевой адрес программного кода апплета и т.д.), задаются в форме атрибутов тега APPLET. Параметры, индивидуальные для каждого конкретного апплета, задаются стандартным образом с помощью произвольного количества тегов PARAM, размещаемых между открывающим и закрывающим тегами APPLET. Построенное таким образом изображение может состоять из произвольного количества элементов, организованных в произвольное количество слоев. Такое задание изображений может рассматриваться как расширение языка HTML для описания сложных векторных изображений.

Статическое построение целесообразно использовать только для сравнительно простых изображений. Сложные изображения более эффективно строить на основе получаемых от сервера упакованных файлов данных. Эти файлы могут быть заранее подготовлены специальными программными средствами на инструментальной машине и переписаны на сервер либо генерироваться непосредственно на сервере динамически (по запросам пользователя) серверным программным обеспечением. При необходимости автономной работы без подключения к Интернет файлы данных могут располагаться в произвольных каталогах пользовательской машины. При этом никакого серверного программного обеспечения не требуется.

Для динамического перестроения изображения под управлением сценариев на JavaScript в апплете реализованы следующие основные группы методов:

— изменение состава групп и слоев изображения, изменение атрибутов отображения его элементов;

— получение информации об элементах изображения, выбранных курсором (об активных зонах);

— работа с виртуальными экранами;

— работа с динамически создаваемыми растровыми изображениями;

— получение информации о состоянии кнопок мыши и клавиатуры;

— некоторые базовые методы редактирования векторных изображений;

— работа с массивами данных (операции над векторами чисел для вычисления производных векторов данных на основе базовых);

— работа с таблицами поименованных объектов;

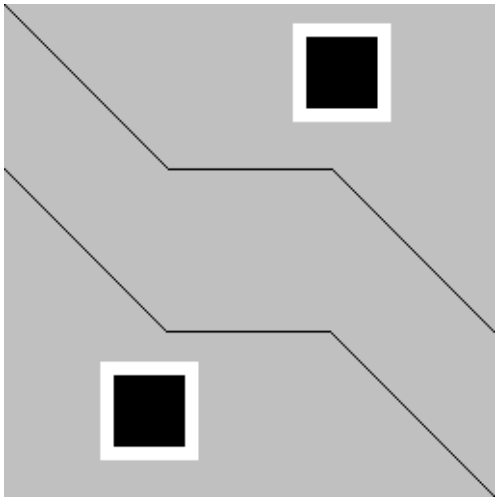
— отображение ниспадающих иерархических меню и вызовов методов, реализующих необходимые действия;

— системные управляющие методы.

Особенностью реализации чтения файлов данных с сервера является кэширование и возможность упреждающего чтения. Во время одного сеанса работы пользователя все данные читаются с сервера и распаковываются только один раз. При повторном обращении к этим данным они берутся из уже построенных и запомненных объектов. Это существенно уменьшает трафик и повышает скорость реакции

программы на действия пользователя. При необходимости апплет может заранее читать файлы данных в фоновом режиме, во время просмотра пользователем уже построенного изображения.

3. Примеры и рекомендации по использованию апплета. Ниже приведен HTML-код страницы, позволяющей получить на экране следующее изображение:



Изображение задается статически, т.е. непосредственно в параметрах апплета.

```
<html>
<head>
<title> Пример статического задания параметров апплета multy_view</title>
</head>
<body>
<center>
<applet code= multy_view.class name= MultyView
    width=255 height=255 archive= multy_view.zip>
  <param name=bgcolor value=#F0F0F0>
  <param name=layers_all value="lines rectangles">
  <param name=layers_draw value="#lines #rectangles" >

  <param name=lines value="draw_layer=0x300
                           border_color_layer=#A0A0A0">
    <param name=lines_object_0 value="line 0 0  85 85  170 85  255 170">
    <param name=lines_object_1 value="iline 254 254  -85 -85  -85 0  -85 -85">

  <param name=rectangles value="draw_layer=0xD00
                                border_color_layer=#FFFFFF
                                border_width_layer=7
                                fill_color_layer=#FFFF00">

   <param name=rectangles_object_0 value="rect 150 10 50 50">
   <param name=rectangles_object_1 value="rect 50 185 50 50 ">

</applet>
</center>
</body>
</html>
```

Для динамического задания того же изображения (с использованием программы на языке JavaScript, работающей после завершения загрузки страницы) можно использовать следующий код на HTML.

```
<html>
```

```
<head>
<title> Пример динамического задания параметров апплета multy_view</title>
<script>
function PreparePicture()
{
    var MV = document.applets.MultyView ;
    if( MV == null ) return;
    MV.open_change_mode(0);

    MV.add_layer( "lines",
"draw_layer=0x300 border_color_layer=#A0A0A0" );
    MV.set_layer_element_num( 0, "object",
"line 0 0  85 85  170 85  255 170" );
    MV.set_layer_element_num( 1, "object",
"iline 254 254  -85 -85  -85 0  -85 -85" );

    MV.add_layer( "rectangles",
"draw_layer=0xD00 border_width_layer=7 border_color_layer=#FFFFFF "+
"fill_color_layer=#FFFF00" );
    MV.set_layer_element_num( 0, "object", "rect 150 10 50 50" );
    MV.set_layer_element_num( 1, "object", "rect 50 185 50 50" );

    MV.set_layers_draw( "#lines #rectangles" );
    MV.close_change_mode( 5 );
}
</script>
</head>
<body onLoad="PreparePicture();">
<center>
<applet
    code= multy_view.class   name= MultyView
    width=250  height=250  archive=multy_view.zip mayscript>
        <param name=bgcolor value=#F0F0F0>
</applet>
</center>
</body>
</html>
```

Ниже приведен код на HTML примера динамической подкачки изображения по запросу пользователя. Загрузка изображения в апплет осуществляется при активизации ссылки на файл picture.htm, который и содержит код задания параметров апплету. По завершении загрузки изображения окно с файлом picture.htm автоматически закрывается.

```
<html>
<head>
<title> Пример динамической загрузки параметров апплета multy_view
        по запросу пользователя</title>
<script>
var MVReady = false;
function SetReadyMode ()
{
  MVReady = true;
}

function LoadPicture()
{
  if( MVReady )
```

```
    window.open('picture.htm', '', 'width=50,height=20' );
}
</script>
</head>
<body onLoad="SetReadyMode();">
<center>
<applet
    code= multy_view.class   name= MultyView
    width=250  height=250  archive=multy_view.zip mayscript>
        <param name=bgcolor value=#F0F0F0>
</applet>
<p><a href=# onCLick="LoadPicture()">Загрузка изображения</a></p>
</center>
</body>
</html>
```

Содержимое файла picture.htm, динамически загружаемого при нажатии кнопки мыши на тексте "Загрузка изображения".

```
<html>
<head>
<title> Пример динамически загружаемого изображения </title>
<script>
function PreparePicture()
{
    var MV = self.opener.window.document.applets.MultyView;
    if( MV == null ) return;
    MV.open_change_mode(0);

    MV.add_layer( "lines",
"draw_layer=0x300 border_color_layer=#A0A0A0" );
    MV.set_layer_element_num( 0, "object",
"line 0 0  85 85  170 85  255 170" );
    MV.set_layer_element_num( 1, "object",
"iline 254 254  -85 -85  -85 0  -85 -85" );

    MV.add_layer( "rectangles",
"draw_layer=0xD00 border_width_layer=7 border_color_layer=#FFFFFF "+
"fill_color_layer=#FFFF00" );
    MV.set_layer_element_num( 0, "object", "rect 150 10 50 50" );
    MV.set_layer_element_num( 1, "object", "rect 50 185 50 50" );

    MV.set_layers_draw( "#lines #rectangles" );
    MV.close_change_mode( 5 );
  self.close();
}
</script>
</head>
<body onLoad="PreparePicture();">
</body>
</html>
```

Одна из сложностей написания JavaScript-программ, осуществляющих динамическую генерацию изображений и обеспечивающих реакции на действия пользователя (нажатие кнопок мыши, клавиш на клавиатуре и попадание курсором в активные зоны изображения), связана с необходимостью синхронизации JavaScript-программ с работой апплета. Явные средства синхронизации практически отсутствуют. Хотя анализ перечисленных выше действий пользователя осуществляет сам апплет, последний не может

активизировать JavaScript-программы по собственной инициативе. Поэтому при написании JavaScript-программ необходимо использовать ряд перечисленных ниже правил и приемов.

1) Вызов методов апплета возможен только после полного завершения его инициализации. Момент готовности апплета можно определить по наступлению события OnLoad, собственную реакцию на которое можно разместить в теге BODY. Однако нужно иметь в виду, что на страничке с апплетом не следует использовать HTML-слои, так как при их использовании событие OnLoad может наступить до завершения инициализации апплета.

2) Для написания интерактивных JavaScript-программ, реагирующих на события, наступление которых определяется апплетом, необходимо организовывать JavaScript-потоки, ведущие регулярный опрос состояния апплета и осуществляющие нужные JavaScript-реакции на эти события.

3) Для написания JavaScript-потоков удобно использовать оператор setTimeout языка JavaScript. Ниже приведен фрагмент HTML-документа, в котором используется поток с именем MultiView, обслуживающий апплет.

```
<head>
...
<script>
function run()
{
        if( document.applets.MultiView.get_selected_in_layer( "layer0", 0 ) >= 0 )
        { // реакция на попадание курсора а активную зону слоя "layer0"
        ...
        }
// очередной запуск цикла опроса состояния апплета через 200 миллисекунд
        setTimeout( "run()", 200 );
}
</script>
...
</head>
<body E OnLoad="run();" ... >
...
<applet name=MultiView mayscript ... > ... </applet>
...
</body>
</html>
```

Подробное описание разработанного апплета и интерактивный электронный учебник по его использованию расположены на Интернет-сайте НИВЦ МГУ (http://www.srcc.msu.su/applet_book).

Описываемый апплет использовался, в частности, в качестве ядра клиентского программного обеспечения при подготовке атласа “Регионы России”, являющегося фрагментом Интернет-сайта Национального информационного агентства “Природные ресурсы” (http://www.priroda.ru), предназначенного для отображения информации о природных ресурсах и охране окружающей среды в субъектах Российской Федерации.

Работа выполнена при поддержке РФФИ, проекты № 98–07–90018, 01–07–90173 и 02–07–90236.

СПИСОК ЛИТЕРАТУРЫ

1. *Мейджер Дж.* JavaScript: основы программирования. Киев: Издательская группа BHV, 1997.
2. *Браун М., Ханикат Д.* HTML 3.2 в подлиннике. СПб.: BHV–Санкт-Петербург, 1998.
3. *Джамса К.* Библиотека программиста JAVA. Минск: ООО “Попури”, 1996.

Поступила в редакцию
30.09.2002