Дивиак Саша

JavaScript и управление компьютерными экспериментами

ВВЕДЕНИЕ

Первая версия программного продукта xyzViewer и последовавшая за ней JavaxyZET позволяли визуализировать результаты моделирования в среде xyZET* на любой платформе, так как были написаны на языке Java. Дальнейшая техническая работа над xyzViewer привела к увеличению качества визуализации и решила многие проблемы совместимости с xyZET. Тщательное изучение JavaScript позволило внести ряд существенных усовершенствований и сделать работу с визуализатором более гибкой. Статья посвящена рассмотрению новых возможностей JavaxyZET.

ИСТОРИЯ СОЗДАНИЯ JAVAXYZET

Качество моделирования достаточно сложных систем не в последнюю очередь определяется мощностью компьютера. Программная среда xyZET одна из тех, что часто используется для моделирования. Это интерактивная графическая оболочка для моделирования физических систем, позволяющая визуализировать трехмерные объекты и конструкции. В качестве таких объектов, которые могут образовывать сложные ансамбли, служат частицы, чьи массы, заряды, начальные положения и скорости определяют их положение в пространстве. В частности, частицы могут быть соединены пружинами, и, таким образом, появляется возможность изучать не только жесткие конструкции. На частицы могут действовать силы инерции и внешние возмущения, а реакцию системы на них можно непосредственно наблюдать на экране. Модель, представляющая собой ансамбль взаимосвязанных частиц, позволяет изучать практически все механические и электрические явления (кинематику, законы сохранения моментов и энергии, законы Гука, гравитации, взаимодействия электрических зарядов, строить соответствующие поля и эквипотенциальные поверхности). Число учебных примеров изучения законов физики, построенных с помощью xyZET, уже превысило несколько сотен.

Для того чтобы наблюдать компьютерные эксперименты на любой платформе, и был разработан визуализатор в форме Java апплета. Апплет (xyzViewer) представлен кнопкой на гипертекстовых страницах электронного учебника. Нажатие на кнопку влечет появление специального окна с трехмерным изображением объекта. Управлять параметрами модели можно с помощью кнопок, движков и любых других управляющих элементов, которые располагаются тут же, на странице учебника. Две специальные программы позволяют получать дополнительную информацию о видимом в окне объекте в форме таблиц и графиков. На графиках можно видеть зависимости от времени скоростей, сил, ускорений и любых других переменных, связанных с объектом. Таблицы представляют способ визуализации переменных и в то же время служат для ввода новых значений.

Одновременно с разработкой xyzViewer были модифицированы и несколько учебников по физике. В основном, эту работу выполнили д-р. Г. Хартел

* См. статью Г. Хартела «Действительно ли моделирование дает возможность понять предмет лучше?», Компьютерные инструменты в образовании, № 2, 1999 г., а также сайт www.colos.ec-lyon.fr

(г. Киль) и члены ассоциации CoLoS из города Мурция (Murcia). Д-р. Хартел внес нужные изменения в учебник по механике, а его коллеги из Испании – в учебники по электричеству и явлениям резонанса. Техническая работа свелась к замене «TCP команд», которые использовались раньше при работе с xyZET, на апплеты, использующиеся в xyzViewer.

Позже программный код xyzViewer был переписан заново, чтобы обеспечить лучшую совместимость с xyZET. Расширились возможности визуализатора, и он стал поддерживать основные команды xyZET, была изменена также форма панелей управления, и, вместо xyzViewer, появился JavaxyZET. Принцип управления вычислительным экспериментом остался таким же, что и в xyzViewer.

Дальнейшие разработки позволили достичь совместимости с ранними версиями электронных учебников, использовавших xyZET для моделирования. Существующая версия xyZET поддерживает такие взаимодействия с помощью апплета «TCP_command». Совместимость была достигнута с помощью нового класса, расширяющего основной класс визуализитора JavaxyZET с тем же именем («TCP_command»). Новый класс содержит интерпретатор (конвертор) исходных команд xyZET в команды JavaxyZET, что позволяет не корректировать ранее написанные учебники и их WEB-страницы. Теперь можно просто заменить старый «TCP_command» апплет новым JavaxyZET апплетом. Однако эта простая замена требует знания основных правил работы с файлами, принятыми в xyZET для написания электронных учебников.

Помимо этого, у визуализатора появились новые возможности. Часть из них – привычные для пользователей современные электронные учебники – заимствованы из других программных средств.

ДРУГИЕ ПОДХОДЫ

Очень интересен подход, использованный в так называемых «Физических Апплетах» («Physlets»), предложенный Вольфгангом Кристианом (Wolfgang Christian). «Физические Апплеты» – это небольшие, легко приспосабливаемые для нужд пользователя Java-апплеты, разработанные специально для обучения физике. Они могут быть использованы во многих WWW-приложениях. Их графика весьма проста, что позволяет сосредоточить внимание учащегося на главном и делает их относительно небольшими и гибкими. Все «физические апплеты» могут настраиваться и изменяться с помощью JavaScript. Это означает, что их можно модифицировать, внося небольшие изменения непосредственно в JavaScript, а не в текст, написанный на языке Java. Этими же средствами можно решить и задачу сбора и анализа данных, используя только взаимодействие между апплетами. Таким образом, управлять экспериментом можно, модифицируя HTML текст.

Еще одна интересная работа выполнена Франциско Эскумбре (Francisco Esquembre). Он предлагает выделить в программном продукте для компьютерного эксперимента три составляющие, взаимодействующие между собой:

- Модель: множество переменных и уравнений.
- Образ: визуализируемые данные.
- Управление: набор интересующих пользователя действий.

Однако обеспечение их взаимодействия требует значительных навыков программирования. С учетом этого им была разработана простая в использовании оболочка EJS (Easy Java Simulations), которая позволяет экспериментатору сосредоточить свое внимание на самой модели и не думать о визуализации и контроле эксперимента.

МОДИФИЦИРОВАННЫЙ JAVAXYZET

Модифицированный JavaxyZET, с учетом современных тенденций, оснащен дополнительными, публичными Java-функциями, которые можно вызывать прямо из JavaScript. Таким обра-

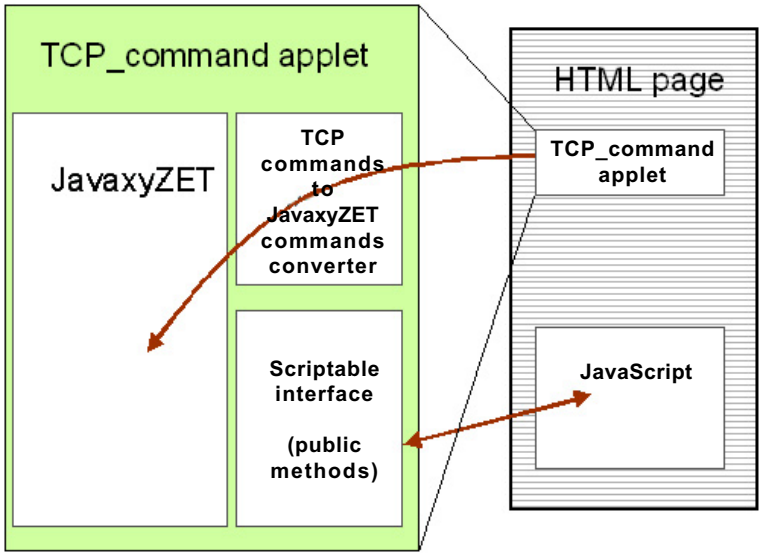


Рисунок 1

зом, можно обеспечить гибкость, предоставляя интерфейс к специальному набору команд. Далее мы рассмотрим, какие дополнительные возможности у нас появляются, при таком «сценарном» подходе (рисунок 1).

Основное преимущество использования всех доступных функций в сценарном подходе заключается в том, что достаточно одного апплета (JavaxyZET) на HTML-странице, а не нескольких, как это было раньше. В предыдущем подходе изменение параметров модели было связано с активизацией соответствующего апплета и передачей данных в JavaxyZET с помощью механизма передачи параметров, существующего у апплетов. Теперь, если требуется менять параметры, предлагается редактировать HTML страницу.

Применение доступных из JavaScript публичных функций – более мощное средство, так как параметры могут меняться с помощью процедур, написанных пользователем. Это позволяет пользователю, имеющему необходимые навыки программирования, самому определять структуру и поведение модели прямо на HTML-странице и расширять возможности среды моделирования. Приведенные ниже примеры иллюстрируют этот подход.

Среда xyZET позволяет создавать описание модели в интерактивном режиме и сохранять его в файле, который может

```
function addPlane (x1, y1, z1, xm, ym, zm, xn, yn, zn, m, n, mass, charge,fixed){
// points T1, Tm and Tn define the plane
// fixed = 0..all particles free, 1.. all particles fixed
   var i,j, x,y,z, dxm, dym, dzm, dxn, dyn, dzn;
   dxm = (xm-x1)/(m-1);
   dym = (ym-y1)/(m-1);
   dzm = (zm-z1)/(m-1);
   dxn = (xn-x1)/(n-1);
   dyn = (yn-y1)/(n-1);
   dzn = (zn-z1)/(n-1);
   for (i=0;i<m;i++){
     for (j=0;j<n;j++) {
       x= x1 + dxm*i + dxn*j;
       y= y1 + dym*i + dyn*j;
       z= z1 + dzm*i + dzn*j;
       document.xyz.addParticle(x,y,z,mass,charge,fixed);
     }
   }
}
```

Программа 1

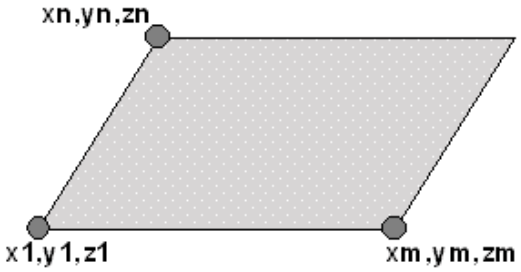


Рисунок 2

использоваться неоднократно. Использование Javascript теперь разрешает создавать даже более сложные, чем в xyZET, структуры и модифицировать их, меняя только соответствующие параметры алгоритмов, созданных для их описания.

Приведенная ниже JavaScript функция описывает плоскость, образованную частицами, массы и заряды которых заданы. Специальный признак указывает, могут ли частицы перемещаться или их положение фиксировано на плоскости (программа 1).

Координаты x1,y1,z1; xm,ym,zm; xn,yn,zn определяют положение плоскости в пространстве, на которой и располагаются все остальные частицы (рисунок 2).

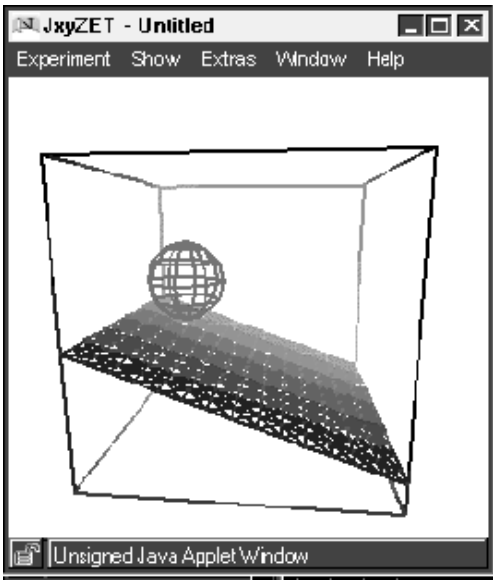


Рисунок 3

Интерфейс для JavaxyZET апплета определяется формой оператора вызова функции

```
addParticle(x,y,z,mass,charge,fixed);
```

одной из публичных функций TCP_command апплета (расширение JavaxyZET). Префикс «document.xyz» указывает на апплет, размещаемый на

Type of created object	**JavaScript function**
Круг на плоскости, образованный частицами	function **addCircularPlane** (x0, y0, z0, radius, m,n, mass, charge, fixed) Параметры: fixed = 0..все частицы свободны, 1.. все частицы фиксированы
Открытый цилиндр, образованный частицами	function **addTube** (x0, y0, z0, radius, height, m,n, mass, charge, fixed) Параметры: fixed = 0..все частицы свободны, 1.. все частицы фиксированы
Многоугольник, заполненный частицами	function **addPlane** (x1, y1, z1, xm, ym, zm, xn, yn, zn, m, n, mass, charge,fixed) Параметры: точки T1, Tm and Tn определяют плоскость fixed = 0..все частицы свободны, 1.. все частицы фиксированы
Многоугольник, заполненный частицами, соединенными пружинами	function **addConnectedPlane** (x1, y1, z1, xm, ym, zm, xn, yn, zn, m, n, k, mass, charge, fixed) Параметры: точки T1, Tm and Tn определяют плоскость fixed = 0..все частицы свободны, 1.. все частицы фиксированы k .. жесткость пружин

Таблица 1

```
<html>
<head>
. . . . . .
<script language=»javascript1.2" src=»extended_xyz.js»></script>
</head>
. . . . . .
```

Программа 2

HTML-странице документа с именем «xyz».

Простым изменением параметров можно создать плоскость с сотнями или тысячами частиц. Ограничением сверху для выбора числа частиц может быть только уменьшающаяся скорость моделирования.

На рисунке 3 вы видите трехмерное окно, построенное с помощью аналогичной Javacript функции. Мячик отскакивает от плоскости, образованной частицами, соединенными пружинами.

Пользователь может написать свою специальную библиотеку таких JavaScript функций и с их помощью создавать достаточно сложные объекты. В свою очередь, последние могут служить строительными блоками еще более сложных объектов.

Пример такой библиотеки (файл «extended_xyz.js») был разработан специально для демонстрации возможностей нового подхода и содержит функции, перечисленные в таблице 1.

Для использования этих функций на HTML странице надо размесить текст программы 2.

Аналогично разрабатываются и другие проблемно-ориентированные библиотеки функций.

Возможности JavaScript можно использовать и для построения алгоритмов управления экспериментом. Это означает, что мы можем сами расширять возможности эксперимента, предложенного авторами программного средства.

Следующий пример показывает, как ввести управление и сделать поведение модели более сложным. Предположим, что в исходном эксперименте мы наблюдаем за движением одного шарика, упруго отскакивающего от невидимой стены. Мы же хотим, чтобы при ударе о стенку, появился бы еще один, меньших размеров шарик. Это означает, что не только параметры, но структура модели может быть изменена, при наступлении заданных событий (рисунок 4).

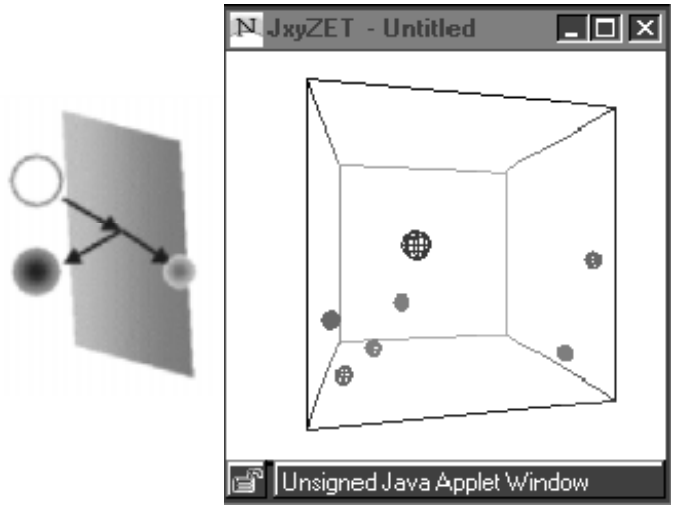


Рисунок 4

Рассмотрим как взаимодействуют два параллельных процесса – JavaScript алгоритм, описывающий изменения, и JavaxyZET, реализующий основной эксперимент. Описание основной JavaScript-функции приведено в программе 3.

Эксперимент начинается с вызова JavaScript-функции **`openExperiment()`**, перезапускающей оболочку JavaxyZET, которая и открывает окно с трехмерным

```
1    <script language="javascript1.2" >
2    function openExperiment() {
3    document.xyz.clearExperiment();
4    document.xyz.setAppletVisible(true);
5      window.setInterval("doStep()", 1); // do step every 1 milisecond)
6    }

7    function doStep(){
8      . . . user- defined algorythm
9      document.xyz.stepForward();
10   }
11   </script>
```

Программа 3

изображением. Это достигается вызовом двух методов из расширения JavaxyZET – **clear Experiment()** и **setAppletVisible()**.

После этого периодически вызывается JavaScript-функция **doStep**, в нашем случае каждую миллисекунду.

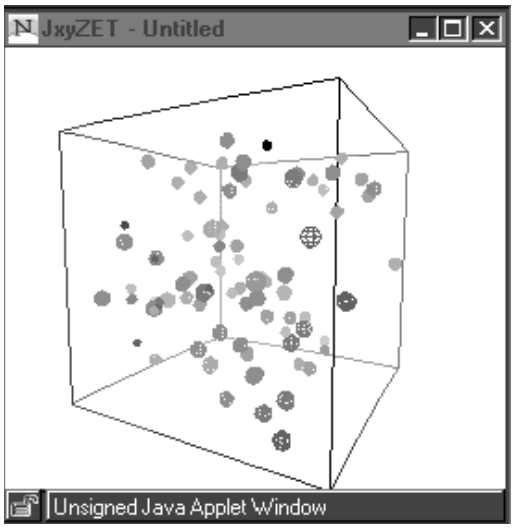
Написанная пользователем функция **doScript()** содержит нужный алгоритм и прерывает процесс моделирования с помощью метода **stepForward()** расширения JavaxyZET.

Функция прерывания (в нашем случае **doStep**) содержит алгоритм, меняющий параметры и структуру модели. Он следит за положением шарика (встреча с невидимой стенкой) и меняет условия проведения эксперимента – добавляет шар с немного уменьшенной скоростью. Текст функции приведен в программе 4.

Этот пример показывает, как можно управлять экспериментом, и одновременно дает примеры функций (**getParticlePositionX**, **setParticleVelocity**, **setParticleSiz**), позволяющих менять параметры модели. Можно написать и множество других аналогичных функций, отслеживающих события и меняющих массу зарядов, цвета частиц.

```
function doStep(){
 x = document.xyz.getParticlePositionX(i); // get the position of bouncing ball
 y = document.xyz.getParticlePositionY(i);
 z = document.xyz.getParticlePositionZ(i);
 vy = document.xyz.getParticleVelocityY(i);
 vz = document.xyz.getParticleVelocityZ(i);
 if (x> xmax) {
  // the bouncing ball collided against «invisible wall»
  document.xyz.setParticleVelocityX(i,-vx);
  //...at the same position a new particle is generated
  document.xyz.addParticle(x,y,z,1,0,0);
  j = document.xyz.getNumParticles()-1;
   document.xyz.setParticleVelocity(j,0.8* vx, 0.8* vy,0.8* vz); // decrease speed
  document.xyz.setParticleSize(j,60);
  document.xyz.setParticleColor(j,70,255,70); // the created ball is green
 };
 document.xyz.stepForward();
}
```

Программа 4



Number of U235 nuclei	66
Number of U239 nuclei	5
Number of neutrons	7
Released energy	800

Рисунок 5

Достигнутая гибкость модифицированного JavaxyZET позволяет ставить эксперименты в других научных областях, так как теперь мы можем создавать свои новые алгоритмы с помощью JavaScript и не ограничиваться только существующими в JavaxyZET. Следующий пример показывает, как использовать трехмерные объекты для моделирования ядерных реакций.

В эксперименте участвуют ядра урана U235 (оранжевые шары), несколько ядер U239 (красные шары) и один нейтрон (белый) с заданной скоростью (рисунок 5). При столкновении нейтрона с ядром U235 ядро делится на два меньших и возникают два дополнительных нейтрона. Дополнительно высвобождается энергия. Ядра U239 работают как поглотитель. Как только нейтрон сталкивается с ядром U239, тот его поглощает. Необходимо найти сбалансированное число ядер U235 и U239 так, чтобы цепная реакция не прекращалась из-за отсутствия нейтронов и

```
                                    function doStep(){
   for (i=0;i<num;i++) {
    if (document.xyz.getParticleCollided(i) != null){
      if(document.xyz.getParticleStatus(i) == 235) {
       // collision with a U235 nucleus
       // the nucleus changes in waste product (142 Ba and 92 Kr)
     document.xyz.setParticleStatus(i,142); // this is now 142 Ba nucleus
      document.xyz.addParticle(x,y, z-40,1,0,1);
       j = document.xyz.getNumParticles()-1;
      document.xyz.setParticleStatus(j,92); // this is 92 Kr nucleus
       //...at the same position a two new neutrons are generated
       createNeutron(x+50 ,y,z+50, vx,vy,vz);
       createNeutron(x-50, y,z+50, -vx,-vy,-vz);
      }
      else
      if(document.xyz.getParticleStatus(i) == 239) {
        // collision with a U239 nucleus which absorbs the neutron
        p = document.xyz.getParticleCollided(i);
        document.xyz.removeParticle(p);
      }
     }
   }
   document.xyz.stepForward();
}
```

Программа 5

не становилась чрезвычайно быстрой (неконтролируемой реакцией).

Соответствующий алгоритм, написанный с помощью JavaScript, достаточно сложен, чтобы приводить его в этой статье, поэтому мы ограничиваемся его упрощенной версией (программа 5).

Основные функции, используемые для динамического изменения структуры модели, – `это addParticle()`, `and removeParticle()`. Алгоритм также отслеживает все столкновения с помощью функции `getParticleCollided()` и изменения параметров частиц (`getParticleStatus()` и `setParticleStatus()`).

ЗАКЛЮЧЕНИЕ

JavaScript может существенно помогать видоизменять ход эксперимента, уже подготовленного и визуализируемого каким-либо программным продуктом. Возможность использовать преимущества объединенных в единое целое Java – JavaScript особенно важна для постоянно экспериментирующих пользователей и увеличивает повторное использование уже разработанных библиотек. Ограничения при таком подходе, в основном, проявляются не при описании сложных экспериментов, а при их проведении, при отсутствии достаточной мощности у компьютера.

Литература.

1. H. Haertel. Physik 3D – Mechanik, xyZET, Ein Simulationsprogramm zur Physik. Begleitbuch, Springer Verlag Berlin, 2000.
2. Francisco Esquembre. Easy Java Simulations. Progress report, Kiel, 2000.
3. W. Christian, M.Belloni: Physlets. Teaching with Interactive Curricular Material. Prentice Hall Series in Educational Innovation, 2000.
4. H.M.Deitel, P.J.Deitel, T.R.Nieto. Internet and WWW, How to Program. Prentice Hall, 2000.

Саша Дивиак,
Университет города Любляны (Словения),
факультет информатики.