Министерство образования Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ 2003»

ТРУДЫ 4-й МЕЖДУНАРОДНОЙ
НАУЧНО-ТЕХНИЧЕСКОЙ КОНФЕРЕНЦИИ

24 - 28 июня 2003 года

Санкт-Петербург

2003

А.С. Кюршунов

РАЗРАБОТКА ИНТЕРАКТИВНЫХ КОМПЬЮТЕРНЫХ МОДЕЛЕЙ В ПОДДЕРЖКУ ДИСТАНЦИОННОГО ОБУЧЕНИЯ ФИЗИКЕ

Петрозаводск, КГПУ

1. Введение

В настоящее время актуальной является проблема разработки учебно-методических материалов для дистанционного обучения (ДО). Рассмотрим данный аспект применительно к курсу физики. Этот интерес усилился в связи с появлением сервиса WWW сети Internet.

Одним из основных методов изучения явлений в естественнонаучных дисциплинах является эксперимент. При переходе к дистанционной форме обучения физике возникает вопрос о том, что же может заменить • экспериментальные работы, ведь в условиях ДО физике обучаемые лишены доступа в лаборатории учебного заведения. Отказ от экспериментальной работы может привести к значительному снижению качества обучения, поскольку физика в своей основе является экспериментальной наукой. Данную проблему можно решать несколькими способами.

Во-первых, можно ограничиться средствами доступными обучаемым - выполнение экспериментов по принципу "Outdoors"[l]. Кроме того, выполнение экспериментов "Outdoors" исполняемых по форме "Откры-

той платформы" [2] с использованием интерактивных компьютерных моделей (в качестве мотивационного средства или средства проверки верности выводов по эксперименту) позволит повысить интерес обучаемых к изучению физики и организовать сетевое взаимодействие в распределённых учебных коллективах. Во-вторых, в курсе физики рассматриваются явления, для изучения которых необходимо дорогостоящее оборудование или их проведение сопряжено с риском для жизни - в подобных случаях наиболее доступным способом является использование интерактивных компьютерных моделей (ИКМ). Следовательно, в ДО физике целесообразно включить ИКМ.

Разработка ИКМ для ДО физике включает два аспекта - методический и технический. Методический аспект определяется содержанием учебного курса, выбором физических явлений для которых возможно создание методически — обоснованных ИКМ (в поддержку натурного эксперимента и для замены экспериментов, для которых невозможно подобное выполнение), отбором наиболее продуктивных моделей обучения. Технический аспект определяется необходимостью выбора средств разработки ИКМ с помощью технологий, учитывающих особенности Internet и возможности взаимодействия распределённых групп. Рассмотрим техническую сторону разработки ИКМ для ДО физике.

К *настоящему* времени разработано большое количество компьютерных программ моделирования таких процессов, как гармонические колебания, движения тел в гравитационном, электрическом и магнитном полях, явлений броуновского движения и диффузии, радиоактивного распада и т.н. Большинство таких программ реализовано для использования пользователем локально и в связи с этим не могут быть применены в электронных гипертекстовых учебниках по соответствующим темам дистанционного курса физики. Необходимо иметь программы, созданные средствами, ориентированными на использование в Internet и одновременно обладающие графическими возможностями. Среди разнообразия подобных средств можно выделить Java-технологию и Flash-анимацию.

В статье приводятся примеры Java-апплета моделирующего движение тела, брошенного под углом к горизонту и Flash-ролика моделирующего движение человека по плоту.

2. Инструментальные средства разработки ИКМ

В настоящее время в рамках традиционного обучения используется большое количество компьютерных демонстраций и лабораторных практикумов. Почти все они ориентированы для использования на компьютерах пользователей. Поэтому они непригодны для условий ДО, так как разработаны для конкретных операционных платформ и являются отдельной средой по отношению к обучающей (для ДО средой является Internet). Это обуславливает необходимость в качестве средств разработки ИКМ для ДО выбирать языковые средства, использующие возможности Internet-технологий, являющиеся платформенно независимыми, обладающие графическими возможностями и позволяющие организовать единый Web-документ. Проведем краткий анализ таких средств.

Для публикации информации повсеместного распространения через сеть Internet необходимо иметь язык, понятный всем компьютерам. Таким языком, используемым в WWW, является HTML. С помощью этого языка возможно:

1) представление текстовой информации для учащихся дистанционно (описание теоретических основ исследуемого явления, описание исследовательской работы и принципа работы с моделью);

2) организация форм для обмена информацией между пользователем, компьютерной моделью и сервером ДО;

3) встраивание Java-апплетов и Flash-анимации в Web-страницы, на основе которых формируются компьютерные модели.

В составе страниц HTML для увеличения функциональности и возможностей взаимодействия с пользователем используется язык JavaScript. Составленные на этом языке сценарии выполняются в результате наступления каких-либо событий, инициированных действиями пользователя. Данная возможность позволяет создавать динамичные Web - документы, способные взаимодействовать с Java-апплетами и Flash-роликами.

Мощным современным средством разработки является Java-технология. Достоинствами этой технологии являются её платформенная независимость и сетевая направленность. Она позволяет' создавать апплеты - приложения встраиваемые в Web - документы. Java API (Application Program Interface) дает программисту независимый от операционной среды доступ к необходимым для создания сложных Internet приложений средствам, таким как сетевые сокеты, потоки и графическая оконная система. Кроме того, Java позволяет создавать клиент - серверные приложения, реализовывает соединение с сетевыми базами данных с помощью интерфейса JDBC (Java DataBase Connectivity) |3|. Для предоставления услуг на Web-сервере предназначены ссрвлсты и JSP (Java Server Pages) |4|.

Однако, создание ИКМ изобилующих динамичными графическими объектами, средствами языка Java, является трудоёмким. В подобных ситуациях целесообразно воспользоваться специализированным средством - Flash. Данное средство позволяет создавать интерактивные видео ролики для Web. поддерживаемые различными браузерами многих популярных операционных систем, а встроенный язык ActionScript по своим возможностям незначительно уступает языку Java. Основным недостатком является неспособность Flash-анимации изображать графики функциональных зависимостей.

3. Пример использования языка программирования Java

В курсе физики, после рассмотрения раздела основ кинематики и темы сила тяжести из раздела основ динамики, традиционно изучают движение тел под действием силы тяжести с начальной скоростью направленной под утлом к горизонту. Разработанная ИКМ позволяет изучить характер движения, его траекторию и особенности зависимости траектории движения тела от начальных условий.

Математической основой создания модели являечся следующая система уравнений [6]:

$$x = x_0 + v_{0x}t; \quad y = y_0 + v_{0y}t - \frac{gt^2}{2};$$

$$v_x = v_0\cos(\alpha_0); \quad v_y = v_0\sin(\alpha_0) - gt;$$

$$g_n = g\cos(\alpha); \quad g^2 = g_n^2 + g_\tau^2; \quad R_{кр} = \frac{v^2}{g_n}$$

где x, y координаты положения тела в момент времени t. v_x, v_y проекции вектора скорости и g_n, $g_т$ нормальное и тангенциальное ускорения в данный момент времени, a - угол наклона вектора скорости к горизонту, REP - радиус кривизны и точке траектории. Вид и особенности чраектории полёта тела определяются начальными значениями положения тела, скоростииуглабросания(соответственно:Xo,yo,Vo,(Xo).

Математическая часть модели осуществлена в классе Body, где реализованы следующие методы:

— Body() — конструктор класса, осуществляющий инициализацию нолей класса отвечающих за начальные параметры модели;

— setVelocity() - метод, устанавливающий скорость бросания;

— setAlfa()- метод, устанавливающий угол бросания;

— setY() - метод, устанавливающий начальную высоту бросания тела;

— getTime() - метод, возвращающий время по заданной координате x;

- getVelocity() - метод, возвращающий значение скорости тела, в точке траектории движения;

getAlfa() метод, возвращающий значение угла наклона вектора скорости, в точке траектории;

— getRadius() - метод, возвращающий значение радиуса кривизны траектории полёта тела в точке;

— gelY() - метод, вычисляющий координату у положения тела по известной координате x.

Класс Motion реализует графическое изображение модели и её взаимодействие с пользователем.

Приведём фрагмент апплета.

```
import java.applet.*;
import java.awt.*;
/*Класс апплета реализующий изображение модели и ее взаимодействие с пользователем */
public class Motion extends Applet}
Image Screen;//Объект класса Image, содержит изображение модели
double Lmax;
Body ResearchBody; //Экземпляр класса Body
int Xmax; int Ymax; int Xmin; int Ymin; /*максимальные и минимальныезначенияэкранныхкоординатрисования*/
double Rxy; /* переменная Rxy предназначена для перехода от системы координат модели в координаты графического контекста апплега и наоборот*/
```

```
public int mouseX, mouseV; /*переменные для хранения местоположение курсора */
public boolean visualXY, visualV, visualA, visuaIGn,
visualGt, visuaIG, visualR; /*переменные определяющие визуализации: объектов модели*/
/*четыре метода преобразования графических координат в координаты модели и наоборот*/
private int LtoX(double L) (return (int) Math.round(L*Rxy+Xmin);}
private double XtoL(int X){return (X-Xmin)/Rxy;}
private int StoY(double S){retum (int) Math.round(Ymax-S*Rxy); }
private double YtoS(int Y){return (Ymax-Y)/Rxy;}
//метот изображения произвольного вектора вектора
private void drawVector(Graphics g, int x, int y, double r,
double a, String s){ ...}
//метод изображающий траекторию полёта
private void FlightPath(Graphics g){
g.setColor(Color.blue);
g.clipRect(Xmi+ l,Ymin+l,Xrnax-Xmin-l,Ymax-Ymin-l);
int x0=Xmin; int y0=Ymax;
for(intx=Xmin;x<=Xmax;x++){
int y=StoY(ResearchBody.getY(XtoL(x)));
g.drawLine(x0,y0,x,y); x0=x; y0=y;}
g.clipRect(0,0,size().width,size().height);}
//метод рисующий координатные оси
private void setAxis(Graphics g){...}
//изображение траектории полёта и координатных осей
private void drawSereen(){
setAxis(Screen.getGraphics());
FlightPath(Screen.getGraphics());}
//метод инициализации апплета
public void init(){
ResearchBody = new Body();
Dimension d = size();
Xmin= 40; Ymin - 20; Xmax = d.width-40;
Ymax = d.height-30; Rxy=(Xmax-Xmin)/Lmax;
Screen - createImage(d,width,d.height);
drawScreen();}
//конвертирование числа в строку
private String DoubleToStr(double num){String S=String.valueOf(num);
return S.substring(0, (num>0)?5:6);}
/*метод изображающий параметры модели соответствующие секущей горизонтальной координате */
public void drawDynObjects(Graphics g){...}
/*метод вызываемый апплетом для перерисовывания графического контекста*/
public void paint(Graphics g){
```

```
g.drawlmage(Screen,0,0,this); drawDynObjects(g);}
//метод обрабатывающий событие передвижения мыши
public boolean mouseMove(Event e, int x, int y){
mouseX-x; mouseY=StoY(ResearchBody.getY(XtoL(x)));
repaint();returntrue;}
//методобработкинажатиямыши
public boolean mouseDown(Event e, int x, int y){
drawDyn()bjects(Screen.gctGraphics());
repaint(); return true;)
/*метод предназначенный для изменения начальных параметром модели,
из html-документа*/
public void setParameters(String L, String x, String v, String a, String
v){...}.
```

Рассмотрим принцип функционирования апплета Motion.java. В данном апплете реализован метод обработки события движения мыши mouseMove(), который позволяет апплету определять текущее местоположение курсора. Поскольку высота положения тела однозначно определяется координатой x, то в переменную mouseX передаётся эта координата, а переменной mouseY присваивается значение координаты, соответствующей высоте полёта тела. Далее вызывается метод paint(), который осуществляет рисование в графическом контексте апплета изображения g.drawImage(Screen,0,0,this), где Screen является объектом класса Image, предназначенным для рисования статичного изображения, которым является рисунок координатных осей и траектории полёта. Затем вызывается метод drawDynObjects(g) который получает ссылку на графический контекст апплета и рисует на нём динамичное изображение, которое представляет из себя совокупность векторов (вектор скорости, нормального и тангенциального ускорения), угла наклона вектора скорости и текущих координат, а также и радиуса кривизны траектории в данной точке. Второй метод обработки события перехватывает нажатие на кнопку мыши mouseDown(), в котором вызывается метод drawDynObjects(), но рисование осуществляется в графическом контексте объекта Screen, таким образом динамичное изображение, соответствующее данному положению курсора мыши, дублируется в статичном, что позволяет пользователю зафиксировать текущие параметры модели, для их анализа и сравнения с параметрами модели соответствующие другим точкам траектории полёта тела брошенного под углом к горизонту.

Поскольку изучение траектории полёта тела, соответствующей одному набору начальных параметров (условий) не представляет большого интереса, то был реализован метод setParameters(), который получает от пользователя из Web-документа новые начальные условия модели, такие как начальная скорость, угол наклона вектора скорости и начальная координата по оси ординат. Эти данные задаются в соответствии с системой "СИ", принятой за основу в математической модели, поэтому в данном методе осуществляется пересчёт переменной Rxy. Далее вызывают-

ся методы selAxis() и FlightPath(), которые рисуют в графическом контексте объекта Screen координатные оси и новую траекторию полета, соответственно.

Для работы апплета создадим HTML-документ который позволит управлять параметрами модели. Приведём функциональную часть документа:

```
<applet code="Motion.class" align="baseline" width="420" height="300"
name="Motion"></applet>
<form name="Param">Визуализация объектов
<input type= "checkbox" name="vXY" onClick= "document. Motion.visualXY =!document. Motion.visualXY;">координатные оси
<input type= "checkbox" name="vV" onClick= "document.Motion.visualV=!document.Motion.visualV">Скорость v
<!-- Аналогично для "угол между OX и v", "ускорение g", "g тангенциальное" "g нормальное" "радиус кривизны"-->
Параметры модели
ширина:<input type="text" size="4" name="L" value="10">м.
координата Xo:<input type="text" size="4" name="x0" value="0">м.
координата Yo:<input type="text" size="4" name="y0" value="0">м.
начальная скорость:<input type="text"size="4"narne="v0"value="9">м/с.
угол к горизонту:<input type="text"size="4"name="a0"value="45">град.
<input typc="button" value="Обработка данных" align="right" onclick="document.Motion.setParam( document.Param.L.value, document.Param.x0.value, document.Param.y0.value, document.Param.a0.value, document.Param.v0.value);"></form>
```

Результат работы апплета представлен на рисунке 1.

Данная компьютерная модель позволяет пользователю устанавливать следующие параметры модели:

— диапазон отображения, по горизонтальной оси - поле формы "ширина";

— начальное положение движущегося тела "координата Yo";

— скорость бросания - "начальная скорость";

— и угол бросания - "угол к горизонту".

Кнопка "Обработка данных" передаёт параметры модели в Java-апплет, который рисует траекторию движения тела. Поле формы, именуемое "Визуализация объектов", позволяет выбирать те объекты, над которыми необходимо проводить исследование. Далее можно перемещать курсор мыши, в области апплета, и наблюдать характер изменения исследуемых объектов. Нажатие клавиши мыши в области апплета фиксирует текущее изображение объектов, что позволяет сравнивать состояния объектов, в различных точках траектории движения тела.

Так на рисунке 1, для наглядности, показаны три положения тела, каждое положение характеризуется собственным набором изображаемых объектов.

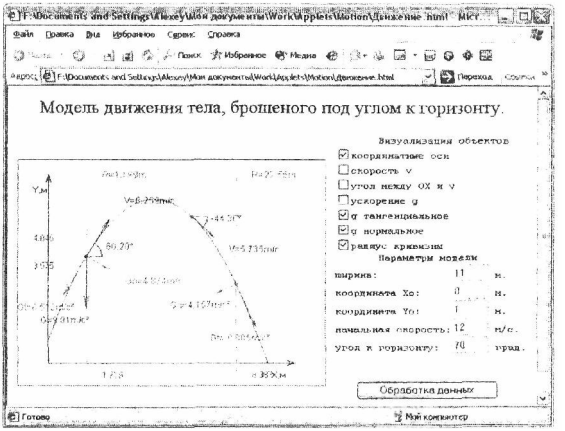

Рис.1.

4. Пример использования Flash-анимации

Рассмотрим задачу о движении центра масс. Заданная ситуация заключается в следующем. Через весь плот проходит человек с некоторой скоростью, требуется выяснить смещение плота, если известны массы человека и плота, длина плота. В основе компьютерной модели лежит следующая система уравнений:

$$x_{hr} = vt; \quad x_h = x_r + x_{hr}; \quad x_c = \frac{x_h m_h + x_r m_r}{m_h + m_r},$$

где x_{hr} - координата положения человека относительно плота, v - скорость передвижения человека относительно плота, x_c - координата центра масс системы, Xh - координата положения человека, m_h - масса человека, m_r - масса плота.

Основные действия по созданию Flash-ролика состоят в следующем. Во-первых, необходимо создать видео ролик ("Movie Clip" с именем human) с изображением человека и текстовым полем, связанным с переменной x_h. Во-вторых, необходимо создать видео ролик с изображением плота (raft) и секстовым полем, связанным с переменной x_r. Далее формируем видео ролик именуемый center - отображающий центр масс системы тел и содержащий текстовое поле, связанное с переменной x_c. Теперь необходимо создать видео ролик содержащий текстовые поля задающие значения параметров модели (связанные с переменными mass_h, mass_r, L_r) и кнопки управления моделью ("Play", "Stop", "Refresh"), данному видеоролику назначается ActionScript выполняющий реализацию модели. Приведём фрагмент программы.

```
onClipEvent(load) {//метод вызываемый после загрузки видеоролика
//инициализация параметров модели
 var mass_h = 70; //масса человека
 var mass_r = 200; //масса плота
```

```
var L_r = 10; //длина илота
var v = 1; //скорость передвижения человека относительно плота
var dt = 0.2; //приращение времени
var x_h = 0; //координата положения человека
var x_r = 0; //координата положения плота
var x_hr = 0; //координата положения человека относительно плота
var x_c; //координата положения центра масс системы
var t = 0; //время
var k = 20; //переменная предназначенная для перехода от системы
//координат модели в координаты графического контекста
var xg0_h = root.human._x; /'/начальная координата положения человека
var xg0_r = _root.raft._x; //начальная координата положения плота
var xg0_c = root.center,_x; // координата положения центра масс
var playing = false; //определяет состояние движения
}
onClipEvent (enterFrame) {//метод вызываемый при переходе и кадр
//видеоролика
if(playing) {
x_hr = v*t;
x_h = x_hr/(l+mass_h/mass_r);
x_r = -x_hr/(l+mass_r/mass_h);
if(x_h>x_r+L_r) {//остановка по достижению человеком края плота play-
ing = false;
_root.human.gotoAndStop( 1);
} else {//перемещения в системе человек плот
setProperty( root.human, x, xg0_h+Math.round(x_h*k));
setProperty(_root.raft, _x, xg0_r+Math,round(x_r*k));
t = t+dt; }
} else {//изменение параметров системы
x_c = mass_r*L_r/(mass_r+mass_h)/2;
setProperty(_root.center, x, xg0_c+Math.round(x c*k));
sctProperty( root.raft.g raft, _width, L_r*k);
setProperty(_rool.raft.raft, x, Math.round(k*L r/2)); }
}
```

События назначенные кнопкам управления моделью:

```
on (release) {//назначается кнопке "Play"
playing = true; root.human.play();}
on (release) {//назначается кнопке "Stop"
playing = false; _root.human.gotoAndStop(l);}
on (release, keyPress "<Home>") {//назначается кнопке "Refresh"
root,human.gotoAndStop( 1);
setProperty(_root.human, _x, xgO_h);
sctProperty(_root.raft, _x, xgO_r);
t = 0; playing - false;}
```

На рисунке 2 изображен вид разработанной ИКМ, позволяющей манипулировать массами объектов и длиной плота.

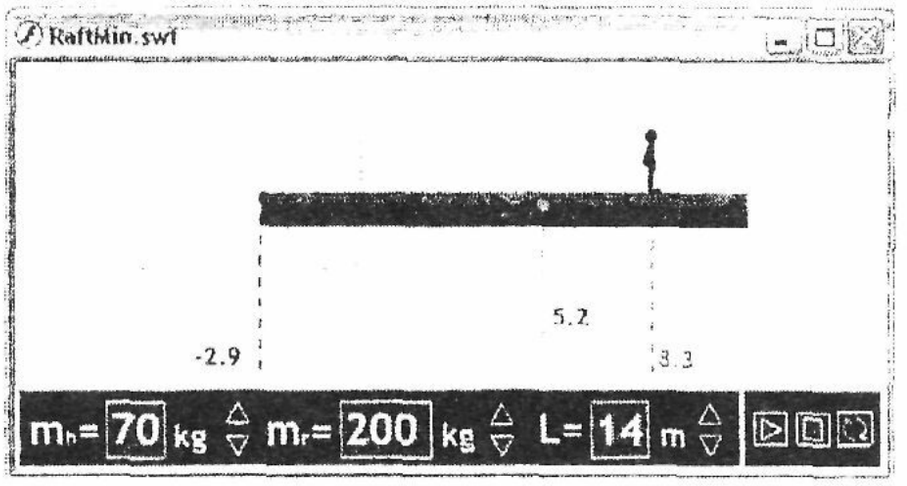

Рис. 2.

Работая с данной моделью, учащиеся могут определить величину смещения плота и убедиться в неизменности положения центра масс системы тел, в ходе перемещения человека по плоту.

5. Итоги

Рассмотренные модели применялись в качестве дополнительного материала для очных семинарских занятий (на нервом курсе физико-математического факультета КГПУ), однако изначальная сетевая направленность позволяет применение их в системе ДО.

Разработка ИКМ с помощью языка Java, в сравнении с Flash-анимацией, является трудоёмким процессом. Однако его использование имеет преимущества, в случае необходимости построения функциональных зависимостей, кроме того, он обладает широкими сетевыми возможностями. Рабочие версии ИКМ доступны по адресу http://aks.dem.ru.

СПИСОК ЛИТЕРАТУРЫ

1. North West Educational magazine // http://www.nwrel.org/nwedu/2002sp/outdoors.htral.
2. Ахаян А.А. //http://www.herzen.dax.ru/top/top2.hlm.
3. Брюс Эккель Философия lava СПб.: Питер, 2001.
4. М. Холл Сервлеты и JavaServer Pages. - СПб.: Питер, 2001.
5. Задачник-практикум по курсу общей физики. Механика. Под редакцией проф. А.Я.Яшкина. М.: Просвещение, 1977.