электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

 # 02, февраль 2016

УДК 0004.4

Интеграция информационных технологий в процесс обучения начертательной геометрии

Сайфуллин К.Р., студент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Радиоэлектронные системы и устройства»

Научный руководитель: Эрастова Е.С.,
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
преподаватель «Инженерная графика»
bauman@bmstu.ru

Наш век - век информационных технологий, и вполне естественно, что сейчас они повсюду: каждый день они захватывают все больше аспектов нашей жизни, поэтому и бежать от них нет смысла, необходимо принять и адаптировать под наши нужды.

Использование новых подходов к образованию помогают оптимизировать процесс обучения для достижения больших результатов за короткий промежуток времени. Последний пункт особенно важен, так как экономия времени очень необходима в нашей жизни.

В курсе начертательной геометрии для развития пространственного воображения были разработаны виртуальные модели вращения плоскости вокруг линии уровня[1;2] и модели объясняющие построение касательных плоскостей и нормалей к поверхностям вращения[3].

Исходя из вышесказанного, вытекают две основные задачи, которые необходимо решить:

1. Облегчение учебного процесса.
2. Уменьшение затрат времени, которое тратится на обучение.

Цель, которую я поставил для себя заключалась в следующем: создать фундамент для решения задач по начертательной геометрии, доказать возможность и показать пути решения базовых задач по выбранному курсу.

Конкретная задача, которую я вызвался решить, формулировалась следующим образом:

«Для произвольного треугольника необходимо построить его копию на заданном расстоянии от точки, которую выберет пользователь, и описать происходящее».

Как предмет для рассмотрения я взял начертательную геометрию, так как в ней присутствуют:

1. Интересные задачи и оригинальные подходы к их решению.
2. Большое методов подходов разнообразие задач
3. .Это первый динамический обучающий подход к программному решению задач по начертательной геометрии.

Реализовывать поставленные цели я решил с помощью HTML5 и Java Script. Это обуславливается:

1. Простотой лексики для написания программ (действительно необходимо иметь минимум знаний по программированию).
2. Простотой в использовании (чтобы открыть данную программу, необходимо иметь всего лишь браузер(единственное ограничение это старые версии Internet Explorer). Но как-либо решать данную проблему я не стал, потому что целевая аудитория либо не пользуется этим браузером, либо используют последние версии, которые приспособлены).
3. Возможностью расширения целевой аудитории.

В будущем можно создать сайт и залить на него эти программы. Так как они написаны на языке, который используется в интернет-программировании, то проблем с отображением на сайте не будет. Таким образом можно захватить большую аудиторию.

Общий алгоритм решения задачи (Каждый из этих пунктов будет рассмотрен в программной части):

1. Проводится фронталь (горизонталь).
2. Проводится нормаль.
3. Откладывается на нормали отрезок заданной длины.
4. Строится копия заданного треугольника.

Вычислительная часть оказалась достаточно простой. Так как мы будем часто иметь дело с координатами и прямыми, то нам необходимо знать уравнения прямой в координатной форме. Это уравнение будет для нас так сказать, фундаментальным. От него мы и будем отталкиваться все время. Через него соответственно можно выражать любые значения x и y:

$$\frac{(x - x_0)}{(x_1 - x_0)} = \frac{(y - y_0)}{(y_1 - y_0)}.$$

$$y = y_0 + \frac{(x - x_0)(y_1 - y_0)}{(x_1 - x_0)}.$$

Мы будем пользоваться не только прямыми, но и нормалями, ее уравнение в координатной форме выведем следующим образом.

Стандартное уравнение нормали:

$$y - y_0 = -\frac{1}{f'(x_0)} \cdot (x - x_0).$$

Из уравнения прямой мы выражаем У, берем производную, которую соответственно подставляем в данное уравнение.

Конечный вид будет следующим:

$$y = y_0 - \frac{(x_1 - x_0)(x - x_0)}{(y_1 - y_0)},$$

где x1, x0, y1, y0 - координаты прямой, к которой проводим нормаль, x и y - координаты точки на нормали.

На этом матчасть заканчивается. Встретилось еще несколько проблем, но их решение будет представлено в программной части. HTML и CSS файлы описывать нет смысла - все написано предельно просто. Так же опущены простейшие функции, такие как: рисование осей координат, проставление точек, рисование линий, оформление.

Для начала скажем несколько слов про систему координат. Она переносится в правую середину рабочего экрана. Координаты ось X направлена в правую сторону. Ось Z направлена вверх. Ось Y вниз. Программа разделена на несколько функций, каждая из которых универсальна и подстраивается под любую точку, которую выбрал пользователь:

```
// Чертим горизонталь
    if(chosenPoint == 1)
    {
      drawHorizontal(pointCX,pointCY,pointCZ,pointBX,pointBY,pointBZ,pointAX,pointAY
  ,pointAZ);
    }
    …

    // Чертим фронталь
    if(chosenPoint == 1) {
    drawFrontal(pointCX,pointCY,pointCZ,pointBX,pointBY,pointBZ,pointAX,pointAY,poin
  tAZ);}
```

```
…
// Чертим нормаль
if(chosenPoint == 1) {
drawNormal(pointAX,pointAY,pointAZ,pointXHorizontalCome,pointYHorizontalCome,poin
tZHorizontalCome,pointXFrontalCome,pointZFrontalCome);}
…
// Откладываем длину
if(chosenPoint == 1) {
  otlogitDliny(pointAX,pointAY,pointAZ,pointXNormalCome1,pointXNormalCome2,poi
ntYNormalCome,pointZNormalCome,200);}
…
// Достраиваем треугольник
if(chosenPoint == 1) {
  newTriangle(pointAX,pointAY,pointBX,pointBY,pointCX,pointCY);}
…
```

Первейшая функция: построение фронтали(горизонтали):

```
function drawFrontal(CX,CY,CZ,BX,BY,BZ,AX,AY,AZ)
{
  …
  // Пересечение горизонтальной проекции фронтали и B'C"
  var F1X = ((((AY - BY)*(CX - BX))/(CY - BY))+BX);
  // Пересечение фронтальной проекции фронтали и B"C"
  var F1Z = ((((F1X - BX)*(CZ - BZ))/(CX - BX))+BZ);
  …
}
```

На вход поступают данные о трех точках (из условия задачи), по вышеприведенной формуле программа находит точку пересечения горизонтальной проекции фронтали или фронтальной проекции горизонтали со стороной треугольника, эта точка переносится во вторую плоскость, и через эту точку завершается построение фронтальной проекции фронтали или горизонтальной проекции горизонтали.

Следующей по списку идет построение нормали:

```
function drawNormal(X1,Y1,Z1,X2,Y2,Z2,X3,Z3)
{
…
      Y = canvas.height/2;
      var X = X1 - ((Y2-Y1)/(X2-X1))*(Y-Y1);
      Z = -canvas.height/2;
      var X3 = X1 - ((Z3-Z1)/(X3-X1)*(Z-Z1));
      var Z3 = Z1 + (((X3-X1)*(Z-Z1))/(X2-X1));
…
 }
```

Пользуясь выведенной формулой, программа находит координаты точек, принадлежащих нормали, проведенной к прямой, Z3 - координата по оси Z, не лежащая на одной линии связи с Y, так получается из-за того, что нам необходимо построить нормаль, которая будет лежать целиком в одной плоскости, не заходя на другую, поэтому нормали «смотрят» в разные стороны.

Третью функцию я распишу подробно, так как эта функция является своего рода отдельной задачей, программное решение логически не отличается от обычного, проблемы таились в реализации:

```
function otlogitDliny(X1,Y1,Z1,X2,X3,Y2,Z2,lenght)
{
// X1,Y1,Z1 - первая точка, через которую проходит прямая
// X2,Y2,Z2 - соответственно вторая точка
// lenght - длина, которую надо отложить

// Для горизонтальной плоскости.

// 1) Берем произвольную точку на прямой. На интервале от Y1 до Y2

if(Y1 < Y2) { var YRand = randomNumber(Y1,Y2);}
if(Y1 > Y2) { var YRand = randomNumber (Y2,Y1);}
if(Y1 == Y2) { var YRand = Math.floor(Y1*1.1);}

var XRand = X1 + (((YRand-Y1)*(X2-X1))/(Y2-Y1));

// 2) Проводим перпендикуляр из точки (XRand,YRand)
Y = 1000;

var X = XRand - ((Y2-YRand)/(X2-XRand))*(Y-YRand);

    // 3) Откладываем на координате Z случайную точку и находим разность координат на оси
    // Это необходимо, чтобы отложить отрезок равный по модулю разности на нормали

    ZRand = Z1 + (((XRand-X1)*(pointZNormalCome-Z1))/(pointXNormalCome1-X1));

    var deltaZ = ZRand - Z1;

    // 3) Откладываем отрезок длиной deltaZ
    // Найдем угол наклона к оси x:
    // Угол наклона прямой, на которой откладываем
    angle = Math.atan(Math.abs(YRand-Y)/ Math.abs(XRand-X));

    // Находим длину, откладываемую на оси x
    var katet = deltaZ*Math.cos(angle);
    var XNatur = XRand-katet;
```

```
// Вторая координата
var YNatur = Y + ((XNatur-X)*((YRand-Y)/(XRand-X)));

// 4) Строим линию натуральной величины:
YLineNatur = 10000;
var XLineNatur = XNatur + (((YLineNatur-YNatur)*(X1-XNatur))/(Y1-YNatur));

// 5) Откладываем на линии натуральной величины отрезок (по уже имеющейся схеме)

// Угол наклона прямой, на которой откладываем
angleR = Math.atan(Math.abs(YNatur-Y1)/ Math.abs(XNatur-X1));

// Находим длину, откладываемую на оси x
var katetR = lenght*Math.cos(angleR);
XReal = X1-katetR;

// Вторая координата
YReal = YLineNatur + (((XReal-XLineNatur)*(Y1-YLineNatur))/(X1-XLineNatur));

// 6) Возвращаем точку на исходную линию. В данной части для решения надо использовать простейшую геометрию. Как видно из приведенного фрагмента выполненной программы, треугольники E'A'R' и RA'A'R1' подобны, из этого исходит последующее решение:

// Найдем длину от вершины треугольника до случайной точки
var TopRand = Math.sqrt(Math.pow(X1-XNatur,2)+Math.pow(Y1-YNatur,2));
// Тоже самое, только до реальной точки
var TopReal = Math.sqrt(Math.pow(X1-XReal,2)+Math.pow(Y1-YReal,2));
// Расстояние основания
var osnova = Math.sqrt(Math.pow(XNatur-XRand,2)+Math.pow(YNatur-YRand,2));
var osnovaIsk = (TopReal*osnova)/TopRand;
var proeksiaX = osnovaIsk*Math.cos(angle);
XREAL = XReal + proeksiaX;
YREAL = Y1 + (((XREAL - X1)*(Y2 - Y1))/(X2-X1));
}
```

Под конец идет простая функция копирования и построения заданного треугольника из вычисленной точки:

```
function newTriangle(AX,AY,BX,BY,CX,CY)
{
    // Достраиваем второй треугольник
    // Берем разницу координат между А и Б
    // И просто по этой разнице достраиваем РБ и так далее

    var differenceX1 = AX - BX;
    var differenceX2 = AX - CX;
```

```
var differenceY1 = AY - BY;
var differenceY2 = AY - CY;

var pointR1X = XREAL-differenceX1;
var pointR1Y = YREAL-differenceY1;
var pointR2X = XREAL-differenceX2;
var pointR2Y = YREAL-differenceY2;

// Третья координата
var differenceX3 = pointR2X - pointR1X;
var differenceY3 = pointR2Y - pointR1Y;
var pointR3X = pointR2X-differenceX3;
var pointR3Y = pointR2Y-differenceY3;

…
}
```

Как видно из программного кода, реализация решения очень проста и требует минимальных знаний программирования и простейшей математики. Таким образом я показал то, что для реализации программного подхода к обучению начертательной геометрии не требуется больших ресурсов. Что дает возможность быстрого развития данного направления. А развиваться можно с различными целями:

1. Создание обучающей программы, презентация решения по очереди. Например, по нажатию на определенную клавишу происходит какое либо, действие и сбоку появляется текстовое объяснение. Преимущества над уже существующими такого рода презентациями огромны: данная программа подстраивается под конкретные введенные координаты учащегося, она занимает гораздо меньше места в памяти, реализуется простыми средствами и она может быть очень доступна.
2. Предоставление возможности творчества учащемуся. Например студент получает домашнее задание, заходит в программу, забивает исходные данные, а дальше сам выбирает последовательность решения, т.е. сам выбирает где, как, в какой последовательности, когда проводить различные операции. Таким образом, ученик сможет графически решить программу, а затем чисто и без лишней грязи перенести в чистовик. Как уже упоминалось раньше, экономится время, и в тоже время даются большие знания за более короткий промежуток.

3. Создание библиотек функций для решения, которые можно будет достаточно густо намешать и сделать этакий винегрет, который будет способен вычислять достаточно сложные задачи.

Конкретно я предлагаю преподавателям начертательной геометрии взамен на некоторые так сказать, поблажки выдавать задания по написанию такого рода программ студентам, и за небольшой срок собрать библиотеку функции для решения задач. Из нее можно будет взять все то, что необходимо и решить любую из вышеприведенных целей.

Список литературы

[1]. Корягина О.М. Создание моделей метода преобразования ортогональных проекций в системе объемного моделирования Autodesk Inventor // Известия высших учебных заведений. Проблемы полиграфии и издательского дела. 2014. № 6. С.35-39

[2]. Корягина О.М. Графическое описание трехмерных объектов в Autodesk Inventor // Главный механик. 2015.№ 5–6. С.72-75

[3]. Корягина О.М. Построения касательных плоскостей и нормалейк поверхностям вращения в системе трехмерного моделирования Autodesk Inventor // Электронный журнал Cloud of Science. 2015. Т. 2. № 1. Режим доступа: http://cloudofscience.ru (дата обращения 30.06.2015).

[4]. Алямовский А.А. SolidWorks. Компьютерное моделирование в инженерной практике (+ CD-ROM). СПб.: БХВ-Петербург, 2005. 800 с.

[5]. Фролов С.А. Начертательная геометрия. М.: Машиностроение, 1983. 240 с.