УДК 681.5:624.072.2

***А.Г. Сорокина***

# ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ РЕЗУЛЬТАТОВ В АВТОМАТИЗИРОВАННОЙ СИСТЕМЕ РАСЧЕТА ПЛОСКИХ СТЕРЖНЕВЫХ СИСТЕМ

Приведено описание разработанной автором обучающей программы для расчета плоских рам методом конечных элементов.

The paper presents the description of the training program for the analysis of plane frames using the method of finite elements, which was developed by the author.

Применение вычислительной техники и программного обеспечения инженерами конструкторами для расчета плоских стержневых систем в современную эпоху стало привычным. Появилось множество программных продуктов, удовлетворяющих потребностям инженера с момента проектирования конструкции до ввода ее в эксплуатацию. Это коммерческие продукты и потребность в них переоценить невозможно. Такие программы призваны качественно и быстро решить поставленную задачу, предоставив результат пользователю. Однако работают они по принципу «черного ящика». Инженер вводит входные данные и получает результат, при этом ему известен только метод, применяемый для решения поставленной задачи, он не может изменить алгоритм, используемый в программе, посмотреть промежуточный результат. Поэтому возникла потребность создания программы, которая будет не только выполнять расчет строительной конструкции, но и поможет инженеру или студенту строительной специальности изучить метод расчета, получить и применить навыки программирования для изменения программы. Эта программа предназначена для расчета плоских рам методом конечных элементов (в дальнейшем «РПР МКЭ») и написана на языке высокого уровня Java, с применением объектно-ориентированного подхода.

Программный продукт, создаваемый как помощник в учебном процессе, должен обладать следующими свойствами: приближенность графического представления к виду в технической учебной литературе, возможность вывода всех промежуточных результатов, просмотр графических результатов, открытые коды программ. Такая программа должна обладать удобным и простым интерфейсом. В большинстве программ для расчета стержневых систем применяются сходные решатели, использующие метод конечных элементов. Это подробно рассмотрено в [1]. Такой способ решения дает точные и результаты за достаточно малый промежуток времени и их можно посмотреть в выходном файле программы. Однако студенту, изучающему принципы расчета строительной конструкции, привычнее оценивать результат в графическом представлении в виде эпюр и схем деформированного состояния. Поэтому не менее интересен процесс создания интерфейса программы, основу которого составляет графический редактор. Имеющие графические редакторы являются закрытыми, поэтому были специально разработаны графические компоненты, которые отображать и редактировать конструкцию на экране, заданные нагружения и закрепления, представлять эпюры результатов и деформированную схему конструкции. Рассмотрим эти пункты подробнее.

Плоская стержневая конструкция это совокупность узлов и элементов. К узлам и элементам прикладываются нагрузки, в узлах задаются закрепления. После расчета для каждого элемента строятся эпюры и деформированная схема. Все эти компоненты отображаются на области рисования панели DrawBord (рис. 1).

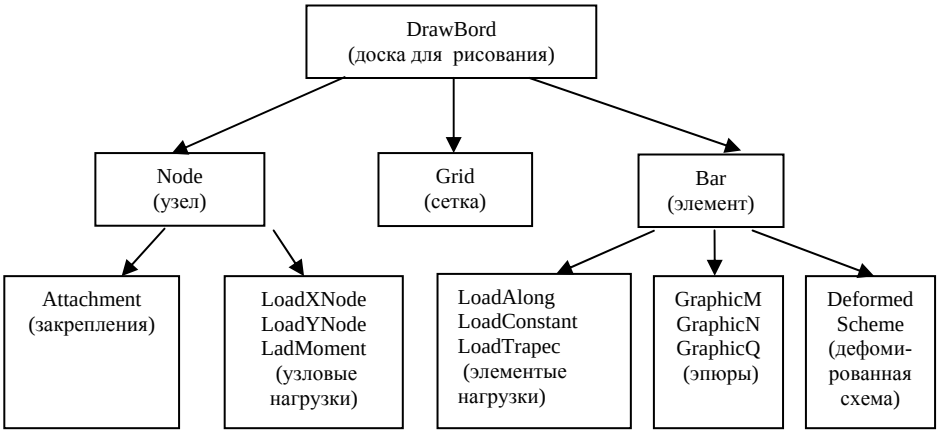


Рис. 1

Область для рисования получают посредством создания экземпляра класса Graphics (стандартная библиотека Java) и присвоения ему области рисования панели DrawBord:

```
Graphics g = DrawBord.getGraphics();
```

Полученная область для рисования передается всем графическим объектам, отображаемым на панели DrawBord, поэтому отрисовка происходит в единой системе координат.

Узел на графической панели представляется как квадрат постоянного размера ($4 \times 4$ пикселей) с номером внизу (рис. 2). Каждый узел имеет два вида координат: реальные, которые задает пользователь в метрах, и координаты на области для рисования в пикселях. Закрепления изображаются точками рядом с узлом: по оси $X$ — справа, по оси $Y$ снизу, закрепления поворота вокруг оси $Z$ — в правом нижнем углу (рис. 2). Графическая панель DrawBord содержит поле Scale — величину, отражающую масштаб конструкции. При изменении параметра Scale координаты узлов на области для рисования пересчитываются и происходит их перерисовка.

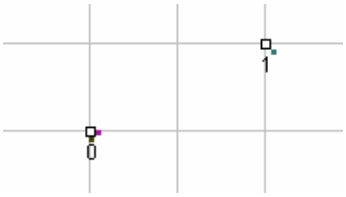


Рис. 2

В узле могут быть приложены три вида узловых нагрузок: вдоль оси *X*, вдоль оси *Y* и моменты вокруг оси *Z*. Для отображения нагрузки в узле применяется следующая последовательность действий:

1) преобразование области рисования из Graphics в Graphics2D для получения новых возможностей работы с графическими примитивами:

2) создание фигуры нагрузки, приложенной в начало системы координат области для рисования;

3) сохранение состояния нетрансформированной области для рисования;

4) выполнение трансформаций системы координат области для рисования с помощью класса AffineTransform;

5) отображение нагрузки на панели DrawBord;

6) восстановление исходного состояния области для рисования.

Далее в Листинге 1 (рис. 3) приводится фрагмент кода для отображения узловой нагрузки вдоль оси *Y*.

Листинг 1

```
public void paintLoad ()//Метод для отрисовки узловой нагрузки вдоль оси Y
{
Graphics2D g =  (Graphics2D) this.g;    //Преобразум область для рисования
                                        //для получения новых свойств

GeneralPath path =  new GeneralPath();  //Создаем путь фигуры
{
    path.append(new Line2D.Double(0,   0 , 0, -40),false);//1-я линия
    path.append(new Line2D.Double(-5, -7,  0,   0),false);//2-я линия
    path.append(new Line2D.Double( 5, -7,  0,   0),false);//3-я линия
}
Shape  shape   =   path;                  //Запоминаем фигуру

AffineTransform save = g.getTransform();  //Сохраняем нетрансформированное
                                          //области для рисования

transforms    =   new AffineTransform();//Создаем трансформацию
transforms.translate(x,y);              //Перемещение системы координат
transforms.rotate   (dir*Math.PI, 0, 0);//Поворот системы координат
//В зависимости от знака нагрузки параметр dir задается как 0 или 1
g.setColor      (Color.RED) ;           //Установка цвета нагрузки
g.transform     (transforms);           //Применение трансформации к области
                                        //для рисования
g.draw          (shape)     ;           //Отрисовка нагрузки
g.setTransform  (save)      ;           //Восстановление исходного состояния
                                        //Области для рисования
}
```

Рис. 3

Аналогично выглядят методы для отрисовки нагрузки вдоль оси *X* и моментов вокруг оси *Z*. На рис. 4 приведены возможные варианты этих графических объектов.

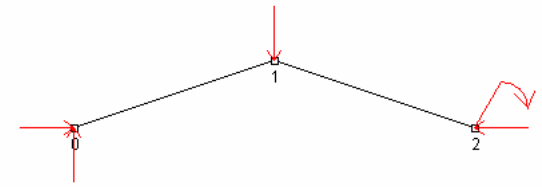


Рис. 4

Элемент отображается как линия между двумя узлами (рис. 5). При отрисовке элемента используются координаты узлов на области для рисования DrawBord в пикселях, поэтому нет необходимости масштабировать элемент. Если изменяется параметр Scale, узлы получают новые координаты, а значит и элемент изменяет свое положение по этим координатам.


Рис. 5

Графические компоненты нагрузки на элемент создаются в той же последовательности, что и узловые. Отличаются лишь по сложности самой фигуры нагрузки и дополнительной трансформацией системы координат области для рисования. В Листинге 2 (рис. 5) приводится фрагмент кода для отображения продольной нагрузки на элемент. На рис. 6 представлены графические компоненты элементных нагрузок.

Листинг 2

```
public void paintLoad () //Метод для отрисовки продольной нагрузки на элемент
{
Graphics2D graphic2D = (Graphics2D) this.g; //Преобразум область для рисования
                                            //для получения новых свойств

GeneralPath path = new GeneralPath();//Создаем путь фигуры
   for(int i=0; i<kol; i++)
{
    path.append(new Line2D.Double((i+1)*s-2, 0, (i+1)*s-2-6,  4), false);
    path.append(new Line2D.Double((i+1)*s-2, 0, (i+1)*s-2-6, -4), false);
}
this.shape        =    path;                    //Запоминаем фигуру

AffineTransform save = graphic2D.getTransform();//Сохраняем
                                                //нетрансформированное
                                                //состояние области для рисования

this.transforms   =    new AffineTransform();//Создаем трансформацию
if (!flag)
    transforms.translate     (x1, y1)       ;//Перемещение системы координат
else
    transforms.translate     (x2, y2)       ;
transforms.rotate (angl, 0, 0)              ;//Поворот на угол наклона элемента
transforms.rotate (dir* Math.PI, (length+3)/2, 0); //Поворот на 180 градусов
                                                   //для отрицательной нагрузки

graphic2D.setColor(color)           ;   //Установка цвета
graphic2D.transform     (transforms);   //Применение трансформаций для системы
                                        //координат
graphic2D.draw          (shape)     ;   //Отрисовка нагрузки
graphic2D.setTransform  (save)      ;   //Восстановление состояния
                                        //области для рисования
}
```

Рис. 5

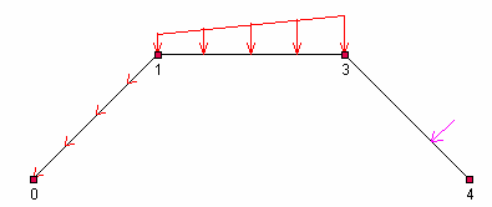


Рис. 6

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Аткин А.В*. Проектирование и реализация автоматизированной системы для расчета плоских стержневых систем на основе объектно-ориентированного подхода // Интернет-вестник ВолгГАСУ. Сер.: Строительная информатика. 2007. Вып. 2 (4). www.vestnik.vgasu.ru.