УДК 539.4

Ю.В. Корниенко

КОМПЬЮТЕРНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ ЧИСЛЕННО-АНАЛИТИЧЕСКОГО МЕТОДА ГРАНИЧНЫХ ЭЛЕМЕНТОВ

Представлена новая версия программы расчета конструкций численно-аналитическим методом граничных элементов – JUKOR2. Значительно расширены возможности программы, впервые разработан графический интерфейс. Использование Java делает программу доступной для использования в разных операционных системах.

Ключевые слова: *численно-аналитический метод, граничные элементы, графический интерфейс, операционные системы.*

Рис. 6. Табл. 2. Форм. 2. Літ. 7.

Ю.В. Корнієнко

КОМП'ЮТЕРНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ЧИСЕЛЬНО-АНАЛІТИЧНОГО МЕТОДУ ГРАНИЧНИХ ЕЛЕМЕНТІВ

Представлена нова версія програми розрахунку конструкцій чисельно-аналітичним методом граничних елементів — JUKOR2. Значно розширені можливості програми, вперше розроблений графічний інтерфейс. Використання Java робить програму доступною для використання в різних операційних системах.

Ключові слова: *Чисельно-аналітичний метод, граничні елементи, графічний інтерфейс, операційні системи.*

Ju.V. Korniyenko

COMPUTER REALIZATION OF ALGORITHMS OF A NUMERICAL- ANALYTICAL METHOD OF BOUNDARY ELEMENTS

The new version of the program of calculation of designs by a numerical- analytical method of boundary elements — JUKOR2 is introduced. Possibilities of the program are considerably expanded; the graphic interface for the first time is developed. Use of Java makes the program available for use in different operating systems.

Keywords : *Numerical-analytic method; boundary elements; graphic interface; operating systems.*

Реализация алгоритмов численно-аналитического метода граничных элементов (ЧА МГЭ) до недавнего времени осуществлялась по программам, написанным на языке среды MATLAB, причем каждая программа носила локальный характер, то есть предназначалась для решения конкретной задачи — расчета балки, рамы, арки и т.д. Отметим также, что построение матриц ЧА МГЭ для использования этих программ выполнялось «вручную», что является достаточно трудоемким процессом.

В связи с этим была поставлена цель: создать такую новую САПР, которая по возможностям расчетов и визуализации их результатов будет способна конкурировать с конечно-элементными пакетами и охватит все имеющиеся на сегодняшний день алгоритмы ЧА МГЭ [1].

Первая версия такой программы охватывала исключительно расчеты стержневых систем [2]. Программа была написана в среде Delphi и позволяла ввести данные, которые необходимы для расчета, в виде таблицы, а графический интерфейс не был разработан.

После обработки данных составлялась квадратная матрица фундаментальных ортонормированных функций $A(x)$ дифференциального уравнения задачи, вектор начальных параметров $X(0)$, вектор параметров напряженно-деформированного состояния стержня в точке x (вектор состояния стержня в точке x) $Y(x)$ и вектор элементов внешней нагрузки $B(x)$, необходимые для составления матричного уравнения ЧА МГЭ

$$\overline{Y}(x) = \overline{A}(x) \cdot \overline{X}(0) + \overline{B}(x). \qquad (1)$$

Затем, в соответствии с алгоритмом метода, преобразовывались векторы $X(0)$ и $Y(x)$ в $X_*(0, x)$, выполнялись соответствующие изменения в матрице $A(x)$ и вычислялись неизвестные начальные и конечные граничные параметры всех стержней системы.

После завершения всех вычислительных операций программа выводила необходимые результаты в виде числовых значений, эпюр и (или) графиков, которые являлись, по сути, единственным атрибутом графического интерфейса первой версии программы.

Описанные возможности программы позволяли реализовать только алгоритмы статического расчета балок и рам на изгиб и динамического расчета на свободные колебания.

Помимо очевидного недостатка — реализована незначительная часть достаточно обширной базы полученных решений ЧА МГЭ [1] — первая версия программы, как и любая другая, написанная в среде разработки Delphi на языке Object Pascal, имеет и ряд других недостатков:

- такие программы можно использовать только под операционной средой Windows;

- возможности реализации разных задач намного меньше, чем в других объектно-ориентированных языках;
- затруднен процесс отладки программы;
- применение среды Delphi требует лицензию.

Поэтому нами была разработана вторая версия САПР на основе численно-аналитического метода граничных элементов, которая получила название JUKOR2.

Первым принципиальным отличием новой версии программы является ее перевод на более мощный язык программирования — Java [3]. Одно из главных преимуществ языка Java — его независимость от платформы, на которой выполняются программы. Таким образом, один и тот же код можно запускать под управлением разных операционных систем — Windows, Linux, FreeBSD, Solaris, Apple Mac и др. Это становится очень важным, когда программы загружаются посредством глобальной сети Интернет.

Второе важнейшее отличие JUKOR2 заключается в том, что программа получила собственный графический интерфейс. Стало возможным визуализировать исходные данные в виде расчетной схемы конструкции, отображаемой на мониторе компьютера. На рис. 1 показано окно программы с построенной моделью стержневой системы, в данном случае — балки.

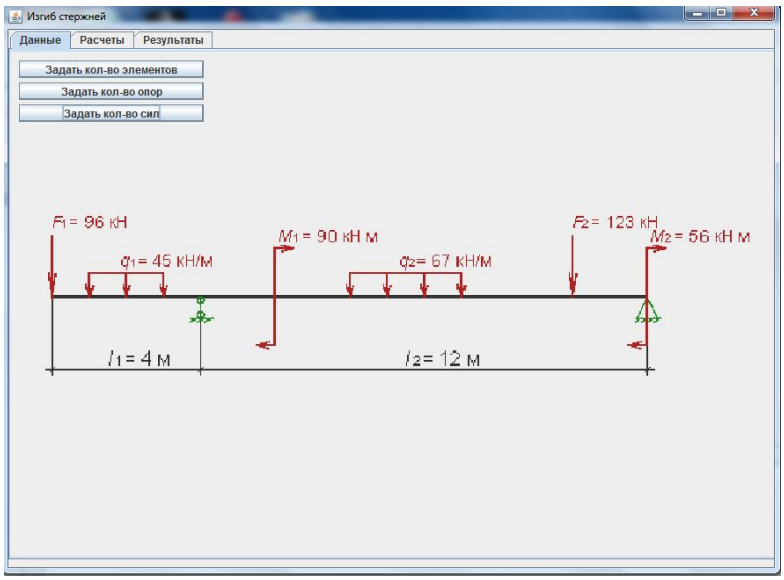

Рис. 1. **Графика в программе JUKOR2**

Поговорим о возможностях графики в Java. Последние несколько лет разработчики прилагали массу усилий, чтобы интегрировать графику и анимацию в свои апплеты и приложения Java. Однако первоначально включенные в Java графические пакеты AWT Java имели ограниченные средства для решения таких задач. Теперь же, используя интерфейсы прикладного программирования Java 2D и Java 3D, разработчики могут реализовывать гораздо более сложные графические приложения. Java позволяет визуализировать любые программы, что придает яркость и удобство для использования тех или иных программ. Позволяет создавать яркие и интересные Web-страницы, удобна при разработке баз данных, написании мобильных и компьютерных игр. Применительно к расчету пластин и оболочек разных видов (да и многих стержневых систем) необходима 3D графика. Программирование трехмерной графики на сегодняшний день является актуальной и разработчики языков программирования не могут оставить эту тему в стороне [4, 5].

В исходной версии пакета JDK 1.0(Java Development Kit) механизм для рисования фигур выглядел очень просто. Можно было только выбирать необходимый цвет и режим рисования и вызывать методы класса Graphics, вроде drawRect() или fillOval(). API-интерфейс в Java 2D поддерживает гораздо больше возможностей. В частности он позволяет делать следующее:

- легко создавать множество различных фигур;
- управлять штрихом, то есть пером, прорисовывающим границы фигур;
- заливать фигуры любым сплошным цветом, используя различные оттенки и узоры;
- использовать трансформации для перемещения, масштабирования, поворачивания и растягивания фигур;
- отсекать фигуры так, чтобы они занимали только какую-то определенную область;

• выбирать правила композиции для описания того, как пиксели новой фигуры должны комбинироваться с пикселями уже существующей фигуры;

• задавать рекомендации по визуализации для достижения компромисса между скоростью загрузки и качеством рисования.

В системе пакетов и классов Java 2D, основа которой — класс Graphics2D пакета java.awt, имеется несколько принципиально новых положений.

Кроме координатной системы, принятой в классе Graphics и названной координатным пространством пользователя (User Space), введена еще система координат устройства вывода (Device Space): экрана монитора, принтера. Методы класса Graphics2D автоматически переводят систему координат пользователя в систему координат устройства при выводе графики.

Преобразование координат пользователя в координаты устройства можно задать "вручную", причем преобразованием способно служить любое аффинное преобразование плоскости, в частности, поворот на любой угол и/или сжатие/растяжение. Оно определяется как объект класса AffineTransform. Его можно установить как преобразование по умолчанию методом setTransform. Возможно выполнять преобразование "на лету" методами transform и translate и делать композицию преобразований методом concatenate.

Поскольку аффинное преобразование вещественно, координаты задаются вещественными, а не целыми числами.

Графические примитивы: прямоугольник, овал, дуга и др., реализуют теперь новый интерфейс shape пакета java.awt. Для их вычерчивания можно использовать новый единый для всех фигур метод draw, аргументом которого способен служить любой объект, реализовавший интерфейс shape. Введен метод fill, заполняющий фигуры — объекты класса, реализовавшего интерфейс shape.

Для вычерчивания (stroke) линий введено понятие пера (реп). Свойства пера описывает интерфейс stroke. Класс Basicstroke реализует этот интерфейс.

Методы заполнения фигур описаны в интерфейсе Paint. Три класса реализуют этот интерфейс. Класс color реализует его сплошной (solid) заливкой, класс GradientPaint — градиентным (gradient) заполнением, при котором цвет плавно меняется от одной заданной точки к другой заданной точке, класс Texturepaint — заполнением по предварительно заданному образцу (pattern fill).

Буквы текста понимаются как фигуры, т. е. объекты, реализующие интерфейс shape, и могут вычерчиваться методом draw с использованием всех возможностей этого метода. При их вычерчивании применяется перо, все методы заполнения и преобразования.

Кроме имени, стиля и размера, шрифт получил много дополнительных атрибутов, например, преобразование координат, подчеркивание или перечеркивание текста, вывод текста справа налево. Цвет текста и его фона являются теперь атрибутами самого текста, а не графического контекста. Можно задать разную ширину символов шрифта, надстрочные и подстрочные индексы. Атрибуты устанавливаются константами класса TextAttribute.

Процесс визуализации (rendering) регулируется правилами (hints), определенными Константами класса RenderingHints.

С такими возможностями Java 2D в новой версии нашей программы JUKOR2 стала полноценной системой рисования, вывода текста и изображений.

Java 2D способна отображать три типа встроенных графических объектов — они называются графическими примитивами — изображения, текст и геометрические фигуры. Имеется семь атрибутов состояния Graphics 2D, которые определяют, как воспроизводятся графические примитивы, — clipping (отсечение), compositing (наложение изображений), font (шрифт), paint (раскрашивание), rendering hints (правила отображения), stroke (отображение линий и контуров) и transforms (преобразования).

Графический контекст Java дает возможность рисовать на экране. Объект Graphics управляет графическим контекстом, задавая способ рисования. Объекты Graphics содержат методы для рисования, работы со шрифтами, цветом и т.п. Каждое приложение, которое выполняет рисование на экране, использует объект Graphics для управления графическим контекстом приложения.

Рисование выполняется различным образом для каждой платформы, поддерживающей Java, поэтому не может быть одного класса, который реализует возможности рисования для всех систем. Например, графические возможности, которые позволяют компьютеру, работающему под Microsoft Windows, рисовать прямоугольники, отличны от графических возможностей, которые

позволяют рисовать прямоугольники под управлением UWIX1, и отличны от графических возможностей, позволяющих рисовать прямоугольники на Macintosh. Для каждой из платформ подкласс Graphics реализует все функциональные возможности рисования. Эта реализация скрыта классом Graphics, который предоставляет интерфейс, позволяющий писать программы, которые используют графику, без учета различий платформ.

Трехмерная графика требует графических алгоритмов, использующих сложный математический аппарат. Java 3D предоставляет разработчикам надежные и развитые возможности для работы с трехмерной графикой, в то же время оставляя за сценой математику, необходимую для реализации графических алгоритмов (рис. 2). Java 3D — это высокоуровневый API программирования трехмерной графики. Java 3D управляет всеми необходимыми низкоуровневыми операциями для работы с графикой, поэтому существует возможность создавать сложные трехмерные сцены, не задумываясь об используемом аппаратном обеспечении. Подобно Java, код Java 3D, будучи написанным, однажды, работает повсеместно. Приложения Java 3D будут работать аналогичным образом на различных графических платформах.

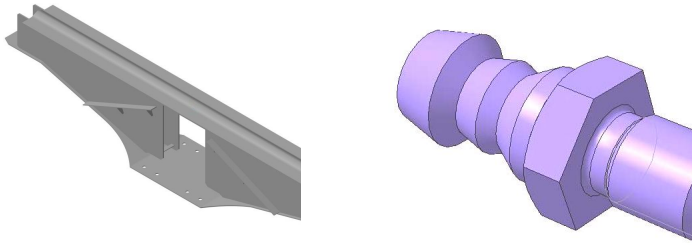

Рис. 2. **Трехмерные модели конструкций на основе Java 3D**

Таким образом, разработанная нами новая версия программы существенно расширяет возможности реализации алгоритмов численно-аналитического метода граничных элементов, визуализации расчетных схем и результатов расчета, а использование Java делает программу доступной для использования в разных операционных системах.

1. Дащенко А.Ф. Численно-аналитический метод граничных элементов / А.Ф. Дащенко, Л.В. Коломиец, В.Ф. Оробей, Н.Г. Сурьянинов — Одесса: ВМВ, 2010. — В 2–х томах. — Т.1. — 416 с. — Т.2. — 512 с.
2. Сур'янінов М.Г., Корнієнко Ю.В. Програмна реалізація....
3. Арнолд К. Язык программирования Java / К. Арнолд, Дж. Гослинг, Д. Холмс — М., СПб., Киев: Вильямс, 2001. — 3-е изд. — 624 с.
4. Дейтел Х. М. Технологии программирования на Java (графика, JAVABEANS, интерфейс пользователя) / Х. М. Дейтел, П.Дж. Дейтел, С.И Сантри — М.: Бином-Пресс, 2003. —В 3-х томах. — Т.1. — 560 с.
5. Бишоп Д. Эффективная работа Java — СПб.: Питер; К.: Издательская группа BHV, 2002. — 592 с.

Стаття надійшла до редакції 24.09.2013.