Теоретический материал для индивидуального задания №3

1) Java 2D API (продолжение)

1.1 Пользовательские классы Stroke

Класс `Stroke` преобразует операцию рисования линий в операцию заливки области, принимая объект `Shape`, очертания которого нужно прорисовать, и возвращая штриховую фигуру, представляющую собой контур исходной фигуры.

Поскольку интерфейс `Stroke` прост, то относительно нетрудно определять собственные классы, реализующие интерфейс `Stroke` и создающие интересные графические эффекты.

`Пример 1` включает в себя четыре пользовательские реализации `Stroke`, которые используются вместе с простым объектом `BasicStroke`.

При изучении примера, особое внимание следует уделить реализациям `ControlPointsStroke` и `SloppyStroke`. Интересны эти классы тем, что они используют объект `PathIterator` для разбиения фигур на составляющие их отрезки прямых и кривых (в противоположность тому, что делалось в классе `Spiral`). Эти два пользовательских класса `Stroke` также используют класс `GeneralPath` пакета `java.awt.geom` для построения пользовательских фигур из произвольных сегментов прямых и кривых (что показывает тесную связь класса `GeneralPath` и интерфейса `PathIterator`).

1.2 Пользовательские фигуры и анимация

Ранее мы уже рассматривали спираль, которая рисовалась при помощи класса `Spiral`, пользовательской реализации интерфейса `Shape`. Интерфейс `Shape` определяет **три важных метода** (часть из них имеют несколько перегруженных (overload) версий), которые должны реализовать все фигуры.

Методы `contains()` определяют, содержит ли фигура точку или прямоугольник; объект `Shape` должен уметь отличать свою внутреннюю область от внешней. Поскольку наша спираль является незамкнутой кривой, у нее нет внутренней области, и эти методы всегда возвращают `false`.

Методы `intersects()` определяют, пересекается ли какая-то часть фигуры с заданным прямоугольником. Поскольку для спирали это трудно вычислить точно, для этих методов программа аппроксимирует спираль окружностью.

Методы `getPathIterator`() составляют ядро любой реализации `Shape`. Каждый метод возвращает объект `PathIterator`, описывающий контур фигуры в терминах сегментов линий и кривых. `Java 2D` использует объекты `PathIterator` при рисовании и заполнении фигур. Ключевыми методами в реализации `SpiralIterator` являются метод `currentSegment`(), возвращающий один прямой отрезок спирали, и метод `next`(), переводящий итератор к следующему отрезку.

Пример 2 - это программа, выполняющая насыщенную графикой анимацию, которая, к тому же, не мерцает, поскольку в ней применяется прием, известный как двойная буферизация: каждый кадр анимации рисуется «вне экрана», а затем целиком копируется на экран. И производительность этого примера выше, поскольку ему приходится перерисовывать относительно небольшую часть экрана - лишь то, что нужно изменить.

Другая интересная особенность этого примера состоит в использовании класса `javax.swing.Timer` для вызова с заданными интервалами метода `actionPerformed`() заданного объекта `ActionListener`. Класс `Timer` применяется здесь для того, чтобы исключить необходимость создания потока `Thread`. (Обратите внимание, что `Java` включает в себя класс `java.util.Timer`, который похож на `javax.swing.Timer`, но все же отличается от него).

Демонстрационные примеры (в папке Примеры)

Для запуска примера используйте команду go.cmd в каталоге примера (требуется java.exe, возможно нужно скорректировать переменную PATH)

Дополнительный материал в главе 7 книги _Books\ corejava2_2.djvu

Литература

1. Хабибуллин И. Ш. Java 7. — СПб.: БХВ-Питербург, 2012
2. Г. Шилдт. Java . Полное руководство, 8-е издание, М.: ООО «И.Д. Вильямс»,
3. 2012
4. Кей С. Хорстман. Java2 Основы. Том 1. С.-Питербург. 2007 (_Books\ corejava2_1.djvu)
5. Кей С. Хорстман. Java2 Тонкости программирования. Том 2. С.-Питербург. 2007 (_Books\ corejava2_2.djvu)
6. http://docs.oracle.com/javase/8/docs/