Теоретический материал для индивидуального задания №2

1) Java 2D API (продолжение)

1.1 Заливка фигур при помощи классов Paint

Java 2D API обобщает понятие цвета и допускает заливку области при помощи любой реализации интерфейса `Paint`. Реализация `Paint` отвечает за задание цветов, используемых при операции заливки области (или рисования линий). Начиная с Java 1.2, класс `Color` реализует метод интерфейса `Paint`, который позволяет заливать область сплошным цветом. Java 2D определяет также две другие реализации `Paint`: `GradientPaint` закрашивает фигуру цветовым градиентом, а `TexturePaint` заливает фигуру копиями изображения-«плитки» (`tile`). Эти классы можно использовать для получения различных эффектов заливки.

1.2 Сглаживание

При рисовании текста и графики можно потребовать, чтобы Java 2D выполнял сглаживание. Сглаживанию подвергаются линии и границы фигур (таких как символы шрифтов), в результате чего уменьшаются видимая ступенчатость. Сглаживание при рисовании необходимо из-за того, что контур фигуры на экране монитора не может быть проведен совершенно ровно; идеальная математическая фигура не может быть абсолютно точно перенесена на дискретную пиксельную решетку. При рисовании фигуры пиксели внутри нее заливаются, а пиксели снаружи нее - нет. Контур фигуры, однако, редко точно попадает на границу между пикселями, поэтому края фигуры приблизительны. В результате получаются ступенчатые линии, приблизительно похожие на абстрактную фигуру, которую мы хотим представить. Сглаживание представляет собой просто технику улучшения вида этого приближения при помощи прозрачных цветов. Если, например, пиксель на краю фигуры покрыт фигурой наполовину, пиксель заполняется полупрозрачным цветом. Если покрыта только одна пятая часть пикселя, пиксель делается прозрачным на одну пятую. Эта техника вполне подходит для уменьшения видимых ступенек.

1.3 Комбинирование цветов при помощи AlphaComposite

Сглаживание осуществляется при помощи рисования краев фигур прозрачными цветами. При рисовании прозрачным цветом нижележащий цвет «просвечивает» сквозь него. Пусть, например, фоновые серые цвета просвечивают сквозь чистые, прозрачные красный и синий, - в результате мы получим синеватый и красноватый оттенки серого.

На аппаратном уровне, разумеется, нет никаких прозрачных цветов; рисование прозрачным цветом имитируется сочетанием цвета рисунка и цвета подложки. Такого рода сочетание цветов называется их композицией и осуществляется интерфейсом `Composite`. Можно передать объект `Composite` методу `setComposite()` объекта `Graphics2D`, чтобы сообщить ему, как следует комбинировать цвет рисунка (цвет источника) с цветами, уже находящимися на поверхности рисунка (цветами приемника). Java 2D определяет одну реализацию интерфейса `Composite`, `AlphaComposite`, которая комбинирует цвета, основываясь на их уровнях прозрачности. Принимаемый объектом `Graphics2D` по умолчанию объект `AlphaComposite` подходит для

большинства рисунков, поэтому создавать собственные объекты `AlphaComposite` приходится редко.

1.4 Обработка изображений

Для фильтрования изображений Java 2D определяет API, основанный на интерфейсе `BufferedImageOp` из пакета `java.awt.image`. Этот пакет также содержит несколько универсальных реализаций этого интерфейса, с помощью которых можно получать эффекты обработки изображений.

Самостоятельно рассмотреть описание классов/интерфейсов в документации JDK: `Paint, GradientPaint, TexturePaint, Composite, AlphaComposite, BufferedImage, Area, BufferedImageOp.`

Дополнительный материал в главе 7 книги _Books\ corejava2_2.djvu

2) Демонстрационные примеры (в папке Примеры)

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

1 Paints.java

Пример демонстрирует использование классов `GradientPaint`, `TexturePaint` и другие возможности Java 2D: прозрачные цвета, представление букв как объектов `Shape` для создания «художественного текста», сочетание трансформаций класса `AffineTransform` и прозрачных цветов для создания «теней», а также применение класса `BufferedImage` для внеэкранного рисования. Пример иллюстрирует также применение класса `GenericPaint`, собственной (custom) реализации интерфейса `Paint`.

2 AntiAlias.java

Пример иллюстрирует процедуру сглаживания: он показывает сглаженный рисунок в искусственно увеличенном виде, чтобы стали видны прозрачные цвета, использованные на краях фигур и элементов шрифта. Рисунок создан с помощью простого кода из примера.

3 CompositeEffects.java

Демонстрирует эффекты, созданные при помощи `AlphaComposite`. В этом примере большая часть рисования производится на внеэкранном изображении, после чего содержимое изображения копируется на экран. Причина в том, что многие эффекты композиции могут быть получены только при работе с такой поверхностью рисования, которая обладает альфа-каналом («alpha channel») и поддерживает прозрачные цвета (например, внеэкранное изображение). Убедитесь, что разобрались, как это внеэкранное изображение создано при помощи `BufferedImage`.

Пример иллюстрирует также тип эффектов, получаемых при задании *области отсечения*. Java 2D позволяет использовать любой объект `Shape` для задания *области отсечения*; графика отображается, только если она попадает в эту область. В примере используется класс `java.awt.geom.Area` для определения сложной фигуры путем комбинации двух эллипсов и прямоугольника, затем эта фигура используется как *область отсечения*.

4 ImageOps.java

В примере показан код, использованный для создания эффектов обработки изображения. Код легко понять: для обработки, изображения `BufferedImage` просто передаются методу `filter()` интерфейса `BufferedImageOp`.

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/