Merlin の魔術: Porter-Duff のルール！

Java 2D が残りの 4 つのルールを追加

John Zukowski 2001年 9月 01日

Java 言語による二次元グラフィックス・プログラミングがちょっとした進化を遂げています。これまでディジタル・イメージ合成に関する 12 の Porter-Duff ルールのうち 8 項目しかサポートしていなかった `AlphaComposite` クラスは、12 項目すべてをサポートするようになりました。Merlin の魔術の今回の記事で、John Zukowski は 12 の全ルールに言及し、インタラクティブ・プログラムを通じてそのはたらきを紹介しています。

このシリーズの他の記事を見る

かつて 1984年に、Thomas Porter と Tom Duff は、「Compositing Digital Images」というテーマの論文を著し、2 つのイメージを合成する場合の 12 のルールについて論じました。こうした合成のルールに対するサポートは、Java 言語のバージョン 1.2 で最初に導入された `AlphaComposite` クラスにみられます。現在ベータ 2 となっているバージョン 1.4 は、12 のルールすべてをサポートします。

合成に対するサポートは、背景イメージやプレーヤー・キャラクター・イメージなどの複数のイメージ・タイプを含むゲームのようなアプリケーションで必要になります。常に背景の上にプレーヤーを描くことは簡単ですが、プレーヤーが木の向こうにジャンプするような状況では、キャラクター・イメージが木のイメージの後ろに姿を消すような効果が欲しくなることでしょう。こんな時こそ、合成が役立ちます。

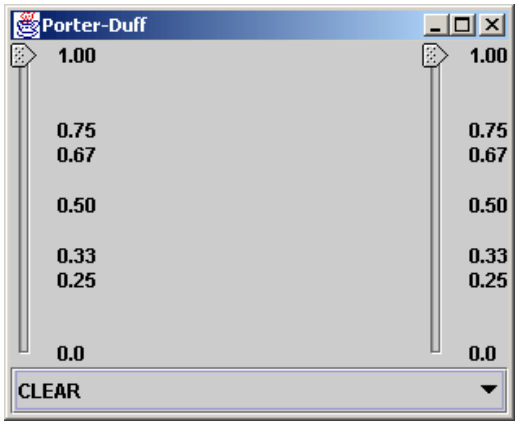
Porter と Duff の 12 のルール

`AlphaComposite` クラスには 12 の定数があり、それぞれが各ルールに対応しています。ルールがどのように適用されるのかを説明すると、込み入ったことになるおそれもあるので、稼動中の各ルールを図解して説明します。イメージが不透明でない場合には、実際の組み合わせも異なってきます。リスト 1 に示されているデモンストレーション・プログラムは、参考文献のセクションでダウンロードできるので、さまざまな透過レベルで試してみることができます。

注: Merlin リリースで新たに加えられたルールには、アスタリスク (*) を付けてあります。

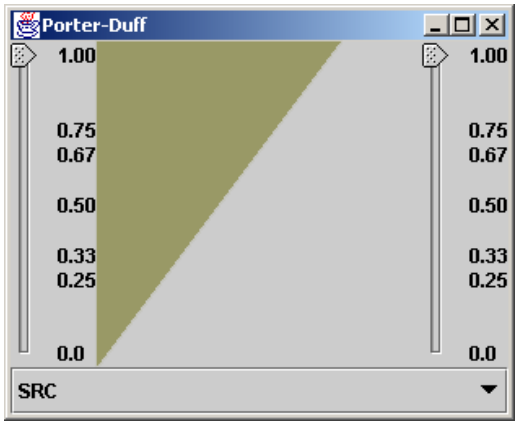
ルール 1.`AlphaComposite.CLEAR`: 結合イメージ (combined image) には何も描画されません。

AlphaComposite.CLEAR



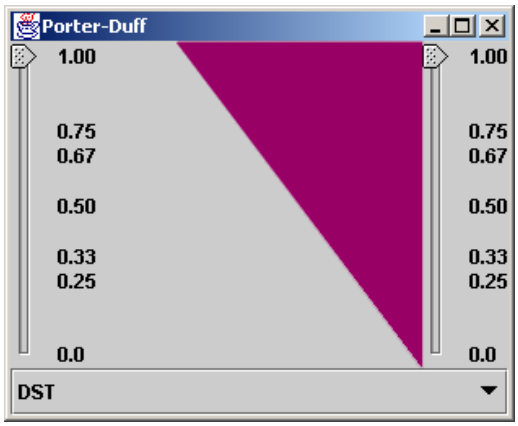
ルール 2.`AlphaComposite.SRC`: SRC はソースを意味します。ソース・イメージ (source image) だけが結合イメージに含まれます。

AlphaComposite.SRC



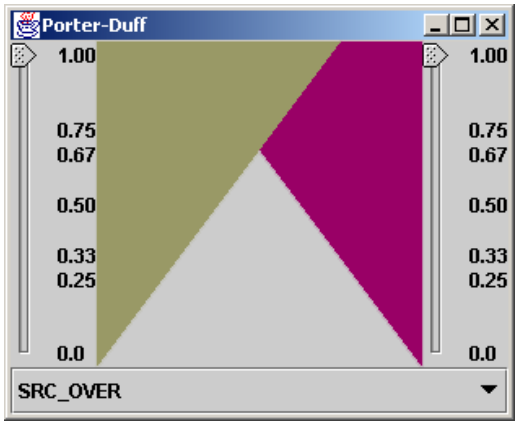
ルール 3.`AlphaComposite.DST`*: DST は宛先を意味します。宛先イメージ (destination image) だけが、結合イメージに含まれます。

AlphaComposite.DST



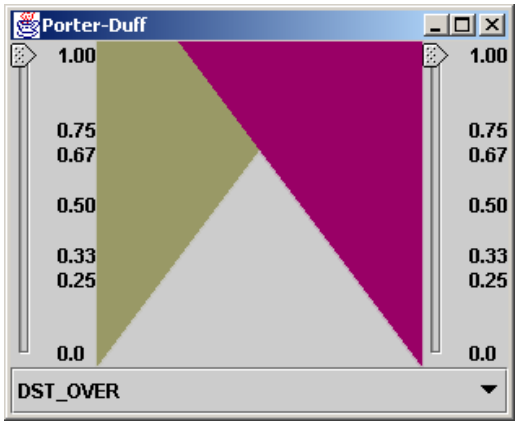
ルール 4.`AlphaComposite.SRC_OVER`: ソース・イメージが宛先イメージの上に描画されます。

AlphaComposite.SRC_OVER



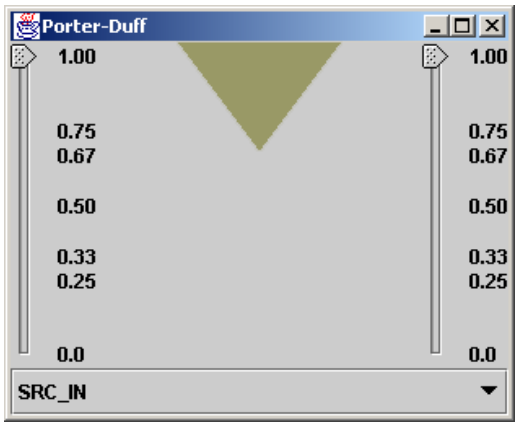
ルール 5.`AlphaComposite.DST_OVER`: 宛先イメージがソース・イメージの上に描画されます。

AlphaComposite.DST_OVER



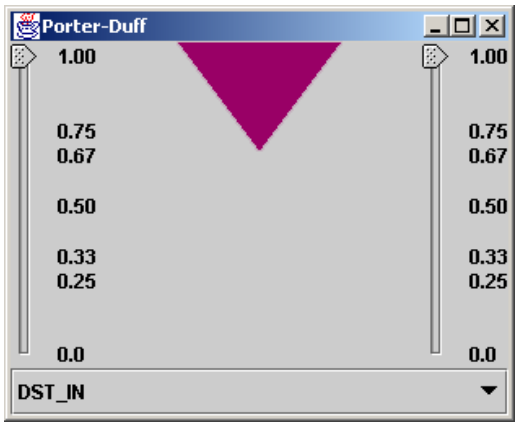
ルール 6.`AlphaComposite.SRC_IN`: ソース・イメージのうちの宛先イメージと重なる部分だけが描画されます。

AlphaComposite.SRC_IN



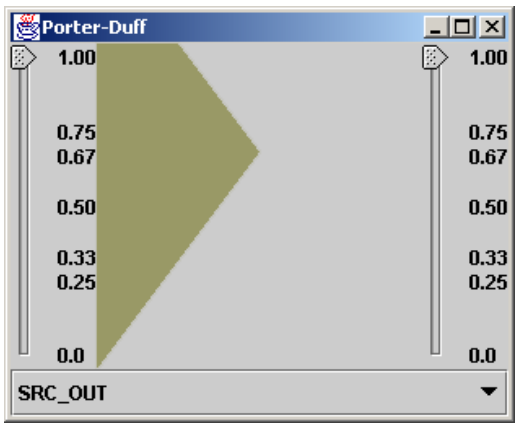
ルール 7.`AlphaComposite.DST_IN`: 宛先イメージのうちのソース・イメージと重なる部分だけが描画されます。

AlphaComposite.DST_IN



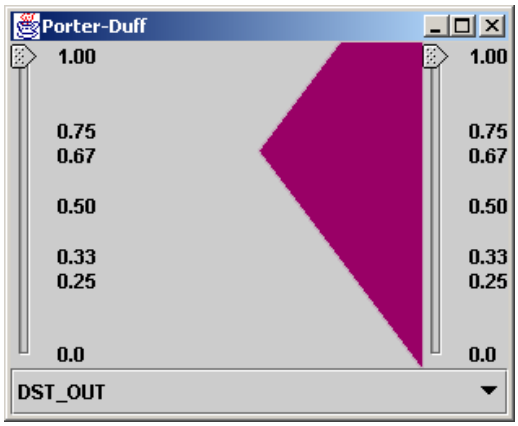
ルール 8.`AlphaComposite.SRC_OUT`: ソース・イメージのうちの宛先イメージと重ならない部分だけが描画されます。

AlphaComposite.SRC_OUT



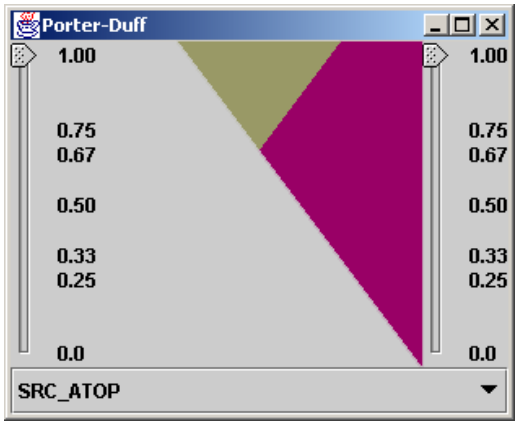
ルール 9.`AlphaComposite.DST_OUT`: 宛先イメージのうちのソース・イメージと重ならない部分だけが描画されます。

AlphaComposite.DST_OUT



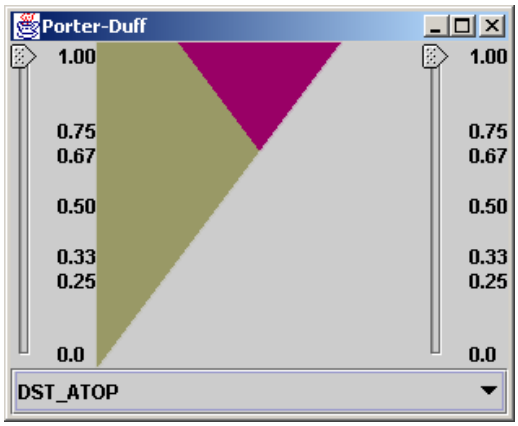
ルール 10.`AlphaComposite.SRC_ATOP`*: ソース・イメージのうちのソース・イメージと重なる部分が、宛先イメージのうちのソース・イメージと重ならない部分とともに描画されます。

AlphaComposite.SRC_ATOP



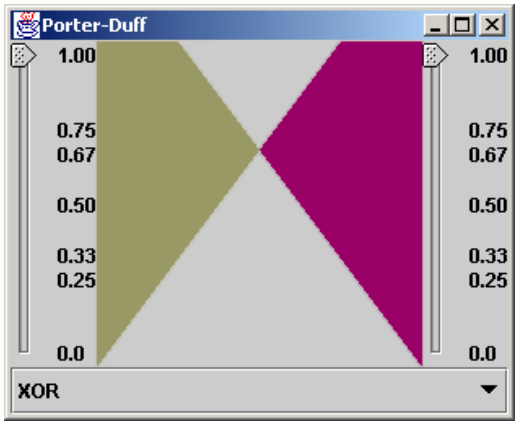
ルール 11.`AlphaComposite.DST_ATOP`*: 宛先イメージのうちのソース・イメージと重なる部分が、ソース・イメージのうちの宛先イメージと重ならない部分とともに描画されます。

AlphaComposite.DST_ATOP



ルール 12.`AlphaComposite.XOR`*: ソース・イメージのうちの宛先イメージと重ならない部分が、宛先イメージのうちのソース・イメージと重ならない部分とともに描画されます。

AlphaComposite.XOR



完結した例

次に示すプログラムは、アルファ合成ルールのインタラクティブ・デモンストレーションです。各トライアングルの不透過度を変更し、イメージの合成による効果を確かめるために使うルールを選んでみてください。

リスト 1. インタラクティブな例

```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import java.awt.image.*;
import java.lang.reflect.*;
import javax.swing.*;
import javax.swing.event.*;
import java.util.*;

public class CompositeIt extends JFrame {
  JSlider sourcePercentage = new JSlider();
  JSlider destinationPercentage = new JSlider();
  JComboBox alphaComposites = new JComboBox();
  DrawingPanel drawingPanel = new DrawingPanel();

  public CompositeIt() {
    super("Porter-Duff");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel contentPane = (JPanel) this.getContentPane();
    Dictionary labels = new Hashtable();
    labels.put(new Integer(0),   new JLabel("0.0"));
    labels.put(new Integer(25),  new JLabel("0.25"));
    labels.put(new Integer(33),  new JLabel("0.33"));
    labels.put(new Integer(50),  new JLabel("0.50"));
    labels.put(new Integer(67),  new JLabel("0.67"));
    labels.put(new Integer(75),  new JLabel("0.75"));
    labels.put(new Integer(100), new JLabel("1.00"));
    sourcePercentage.setOrientation(JSlider.VERTICAL);
    sourcePercentage.setLabelTable(labels);
    sourcePercentage.setPaintTicks(true);
    sourcePercentage.setPaintLabels(true);
    sourcePercentage.setValue(100);
    sourcePercentage.addChangeListener(new ChangeListener() {
      public void stateChanged(ChangeEvent e) {
        int sourceValue = sourcePercentage.getValue();
        drawingPanel.setSourcePercentage(sourceValue/100.0f);
      }
    });
    destinationPercentage.setOrientation(JSlider.VERTICAL);
    destinationPercentage.setLabelTable(labels);
    destinationPercentage.setPaintTicks(true);
    destinationPercentage.setPaintLabels(true);
    destinationPercentage.setValue(100);
    destinationPercentage.addChangeListener(new ChangeListener() {
      public void stateChanged(ChangeEvent e) {
        int destinationValue = destinationPercentage.getValue();
        drawingPanel.setDestinationPercentage(destinationValue/100.0f);
      }
    });
    String rules[] = {
      "CLEAR",    "DST",
      "DST_ATOP", "DST_IN",
      "DST_OUT",  "DST_OVER",
      "SRC",      "SRC_ATOP",
      "SRC_IN",   "SRC_OUT",
      "SRC_OVER", "XOR"};
    ComboBoxModel model = new DefaultComboBoxModel(rules);
    alphaComposites.setModel(model);
```

```
    alphaComposites.setSelectedItem("XOR");
    alphaComposites.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        String alphaValue = alphaComposites.getSelectedItem().toString();
        Class alphaClass = AlphaComposite.class;
        try {
          Field field = alphaClass.getDeclaredField(alphaValue);
          int rule = ((Integer)field.get(AlphaComposite.Clear)).intValue();
          drawingPanel.setCompositeRule(rule);
        } catch (Exception exception) {
          System.err.println("Unable to find field");
        }
      }
    });
    contentPane.add(sourcePercentage, BorderLayout.WEST);
    contentPane.add(destinationPercentage, BorderLayout.EAST);
    contentPane.add(alphaComposites, BorderLayout.SOUTH);
    contentPane.add(drawingPanel, BorderLayout.CENTER);
    pack();
  }

  public static void main(String args[]) {
    new CompositeIt().show();
  }

  class DrawingPanel extends JPanel {
    GeneralPath sourcePath, destPath;
    BufferedImage source, dest;
    float sourcePercentage = 1, destinationPercentage = 1;
    int compositeRule = AlphaComposite.XOR;
    Dimension dimension = new Dimension(200, 200);

    public DrawingPanel() {
      sourcePath = new GeneralPath();
      sourcePath.moveTo(0,   0);   sourcePath.lineTo(150, 0);
      sourcePath.lineTo(0, 200);   sourcePath.closePath();
      source = new BufferedImage(200, 200, BufferedImage.TYPE_INT_ARGB);
      destPath = new GeneralPath();
      destPath.moveTo(200,  0);    destPath.lineTo(50, 0);
      destPath.lineTo(200, 200);    destPath.closePath();
      dest = new BufferedImage(200, 200, BufferedImage.TYPE_INT_ARGB);
    }

    public void setSourcePercentage(float value) {
      sourcePercentage = value;
       repaint();
    }

    public void setDestinationPercentage(float value) {
      destinationPercentage = value;
       repaint();
    }

    public void setCompositeRule(int value) {
      compositeRule = value;
       repaint();
    }

    public void paintComponent(Graphics g) {
      super.paintComponent(g);
      Graphics2D g2d = (Graphics2D)g;
      Graphics2D sourceG = source.createGraphics();
      Graphics2D destG = dest.createGraphics();

      destG.setComposite(AlphaComposite.Clear);
      destG.fillRect(0, 0, 200, 200);
      destG.setComposite(AlphaComposite.getInstance(
```

```
          AlphaComposite.XOR, destinationPercentage));
      destG.setPaint(Color.magenta);
      destG.fill(destPath);

      sourceG.setComposite(AlphaComposite.Clear);
      sourceG.fillRect(0, 0, 200, 200);
      sourceG.setComposite(AlphaComposite.getInstance(
        AlphaComposite.XOR, sourcePercentage));
      sourceG.setPaint(Color.green);
      sourceG.fill(sourcePath);

      destG.setComposite(AlphaComposite.getInstance(compositeRule));
      destG.drawImage(source, 0, 0, null);
      g2d.drawImage(dest, 0, 0, this);
    }

    public Dimension getPreferredSize() {
      return dimension;
    }
  }
}
```

ダウンロード可能なリソース

内容	ファイル名	サイズ
j-mer0918.zip	j-mer0918.zip	

関連トピック

- さまざまな Porter-Duff ルールについては、原典の SigGraph paper をお読みください。
- Java 2 SDK, 1.4 の新たな Java 2D features について理解を深めてください。
- 「The Java 2 user interface」(developerWorks、2001年 7月) で Java UI の発展過程についてお読みください。
- IBM ではグラフィックスとビジュアリゼーションの分野で幅広い研究を行っています。現在進行中のプロジェクトをご覧ください。
- John Zukowski による、以下の Merlin のヒント集もご一読ください。
  - Swing の新たな Spinner コンポーネント: JSpinner を使用して、ユーザーがピック・リストから素早く日付、数値および選択肢を選べるようにする
  - もう 1 つのシンプルなフレーム: AWT Frame を最大化し、その装飾を取り外す
  - 長期の永続性: JavaBean コンポーネントの状態を XML にシリアライズする
  - 挿入順序を保持する: 新たにリンクされた HashSet と HashMap の実装を使ってみる
  - タブ付きペインのスクロール: 大き過ぎて収まりきらない場合にどうするか
- developerWorks の Java ゾーンで他の Java 参考文献をご覧ください。