Przekształcenia 2D w bibliiotece Java 2D

February 24, 2020

1 Introduction

Celem jest użycie biblioteki Java 2D.

Do tworzenia prostych rysunków możemy wykorzystać metody klasy Graphics. Metody te sa wystarczajace w przypadku podstawowych apletów i aplikacji, ale nie nadaja sie do zastosowań wymagajacych tworzenia skomplikowanych rysunków lub potrzebujacych daleko idacej kontroli nad tworzona grafika. Do takich zastosowań przeznaczona jest biblioteka Java 2D, której możliwości przedstawimy tutaj.

2 Potokowe tworzenie grafiki

JDK 1.0 dysponowła bardzo prostym mechanizmem tworzenia grafiki. Wystarczyło wybrać jedynie kolor i tryb rysowania, a nastepnie wywołać odpowiednia metode klasy **Graphics**, na przykáad `drawRect` lub `fillOval`.

Tworzenie grafiki za pomoca Java 2D udostepnia wiele wiecej możliwości:

- tworzenie różnorodnych figur,
- kontrole nad sposobem tworzenia obrysu figur,
- wypełnianie figur różnymi kolorami i ich odcieniami, a także wzorami wypełnień,
- metody przekształceń figur — przesuniecia, skalowania, obrotu i zmian proporcji,
- przycinanie figur do dowolnie wybranych obszarów,
- wybór zasad składania obrazów opisujacych sposób tworzenia kombinacji pikseli nowej figury z istniejacymi już pikselami tła,

- wskazówki tworzenia grafiki umożliwiajace uzyskanie kompromisu pomiedzy szybkościa tworzenia grafiki a jej jakościa.

Aby narysować figure, należy kolejno wykonać nastepujace kroki.

1. Musimy uzyskać obiekt klasy `Graphics2D`, która jest klasa pochodna klasy `Graphics`. Poczawszy od Java SE 1.2 metody, takie jak `paint` czy `paintComponent`, automatycznie otrzymuja jako parametr obiekty klasy `Graphics2D`. Wystarczy wiec wykonać nastepujace rzutowanie:

   ```
   public void paintComponent(Graphics g)
   {
    Graphics2D g2 = (Graphics2D)g;
    . . .
   }
   ```

2. Korzystamy z metody `setRenderingHints` w celu ustalenia kompromisu miedzy szybkościa tworzenia grafiki a jej jakościa.

   ```
   RenderingHints hints = . . . ;
   g2.setRenderingHints(hints);
   ```

3. Wywołujemy metode `setStroke` w celu określenia rodzaju śladu pedzla, który zostanie użyty do narysowania obrysu figury. Musimy wybrać odpowiednia grubość śladu pedzla oraz sposób jego pozostawiania (ciagły, przerywany).

   ```
   Stroke stroke = . . . ;
   g2.setStroke(stroke);
   ```

4. Wywołujemy metode `setPaint` w celu określenia sposobu wypełnienia obszaru ograniczonego obrysem. Należy wybrać wypełnienie stałym kolorem, kolorem o zmieniajacych sie odcieniach lub wzorem wypełnienia.

   ```
   Paint paint = . . . ;
   g2.setPaint(paint);
   ```

5. Korzystamy z metody `clip` w celu określenia obszaru przyciecia.

   ```
   Shape clip = . . . ;
   g2.clip(clip);
   ```

6. Wywołujemy metode `transform`, aby przekształcić współrzedne użytkownika na współrzedne urzadzenia. Przekształcenie takie stosujemy, jeśli w programie łatwiej nam posługiwać sie własnym układem współrzednych niż współrzednymi pikseli.

```
AffineTransform transform = . . . ;
g2.transform(transform);
```

7. Wywołujemy metode `setComposite` dla określenia zasady składania obrazów opisujacej tworzenie kombinacji nowych pikseli z już istniejacymi.

   ```
   Composite composite = . . . ;
   g2.setComposite(composite);
   ```

8. Tworzymy figure. Java 2D udostepnia w tym celu wiele obiektów i metod umożliwiajacych tworzenie kombinacji figur.

   ```
   Shape shape = . . . ;
   ```

9. Rysujemy i (lub) wypełniamy figure. Jeśli narysujemy figure, to powstanie jedynie jej obrys, który nastepnie możemy wypełnić.

   ```
   g2.draw(shape);
   g2.fill(shape);
   ```

   Oczywiście nie zawsze musimy realizować wszystkie wymienione etapy. Parametry kontekstu graficznego posiadaja wartoĞci domyĞlne wystarczajace w wielu przypadkach.

Dalej przedstawimy sposób tworzenia figur, definiowania śladów pedzla i wypełnień, przekształcenia i zasady składania obrazów. Metody `set` nie wywołuja żadnych operacji graficznych, a powoduja jedynie zmiany stanu kontekstu graficznego.

Podobnie utworzenie obiektu reprezentujacego figure nie powoduje narysowania jej reprezentacji. Operacje graficzne wykonywane sa jedynie na skutek wywołania metod `draw` lub `fill`.

W ich rezultacie reprezentacja graficzna figury tworzona jest w `potoku rysowania`.

3 Figury

Klasa `Graphics` udostepnia szereg metod umożliwiajacych rysowanie figur:

```
drawLine
drawRectangle
drawRoundRect
draw3DRect
drawPolygon
drawPolyline
drawOval
drawArc
```

Posiada także odpowiadajace im metody `fill` tworzenia wypełnień figur.

Wszystkie te metody dostepne sa w klasie `Graphics` już od wersji JDK 1.0. Natomiast `Java 2D` proponuje inne, bardziej obiektowe podejście. Zamiast metod udostepnia odpowiednie klasy:

```
Line2D
Rectangle2D
RoundRectangle2D
Ellipse2D
Arc2D
QuadCurve2D
CubicCurve2D
GeneralPath
```

Wszystkie wymienione klasy implementuja interfejs `Shape`. Zdefiniowana jest także klasa `Point2D`, która choć reprezentuje punkt o współrzednych x i y, a nie figure, to używana jest podczas definiowania figur.

4 Pola

`Java 2D` udostepnia cztery operacje geometrii pól, z których każda tworzy nowe pole jako kombinacje dwu innych pól:

- `add` — utworzone pole zawiera wszystkie punkty, które należały do jednego z pól wyjściowych,
- `subtract` — utworzone pole zawiera wszystkie punkty, które należały do pierwszego pola, ale nie do drugiego,
- `intersect` — utworzone pole zawiera wszystkie punkty, które należały jednocześnie do obu pól,
- `exclusiveOr` — utworzone pole zawiera wszystkie punkty, które należały do jednego z pól wyjściowych, ale nie do obu naraz.

5 Ślad pedzla

6 Przekształcenia układu współrzednych

Literatura

Uwaga! Pełny opis potoku rysowania `Graphics 2D` przedstawoiny w ksiażce https://pdf.helion.pl/javtz9/javtz9.pdf (patrz str.13-47)

Teoretyczne podstawy przekształceń geometrycznych umieszczone w prezentacji wykładu https://drive.google.com/drive/folders/0B0-jI5UeRsjcVExBTExOcXNlcGc
W jezyku angielskim

- ksiażka interakcyjna http://math.hws.edu/graphicsbook/contents-with-subsections.html (rozdziały 2.1-2.5)

- podstawy jezyka Java http://math.hws.edu/graphicsbook/a1/s1.html

7 Zadania

1. Program `Transform2D.java` rysuje obraz `shuttle.jpg` w panelu. Narysować zamiast obrazu wielokat według wariantu (liczba n) w panelu wyświetlania. Panel ma wymiary 600 na 600 pikseli, a wielokat ma promień 150 pikseli. Okno zawiera również wyskakujace menu z etykieta "Transform:". Opcje w menu to "None" i cyfry od 1 do 9. W tym programie menu wyskakujace nie działa. Zadanie polega na dodaniu kodu do metody `paintComponent ()`. (Miejsce jest oznaczone TODO.) Kiedy wybór ma wartość 0, strona powinna wyświetlać obraz nietransformowany. W przypadku innych możliwych wartości musisz zastosować przekształcenie (lub bedziesz potrzebował kombinacji przekształceń) dla każdej z wartości od 1 do 9 (patrz Fig. 1).

2. Narysować figure określona wariantem (patrz Fig. 2). Dostepne sa trzy podstawowe kształty: circle (), square () i triangle (). Zacznij od programu TransformedShapes.java. TODO. Możesz użyć poleceń do rysowania, takich jak g.fillRect () itp.

Figure 2:

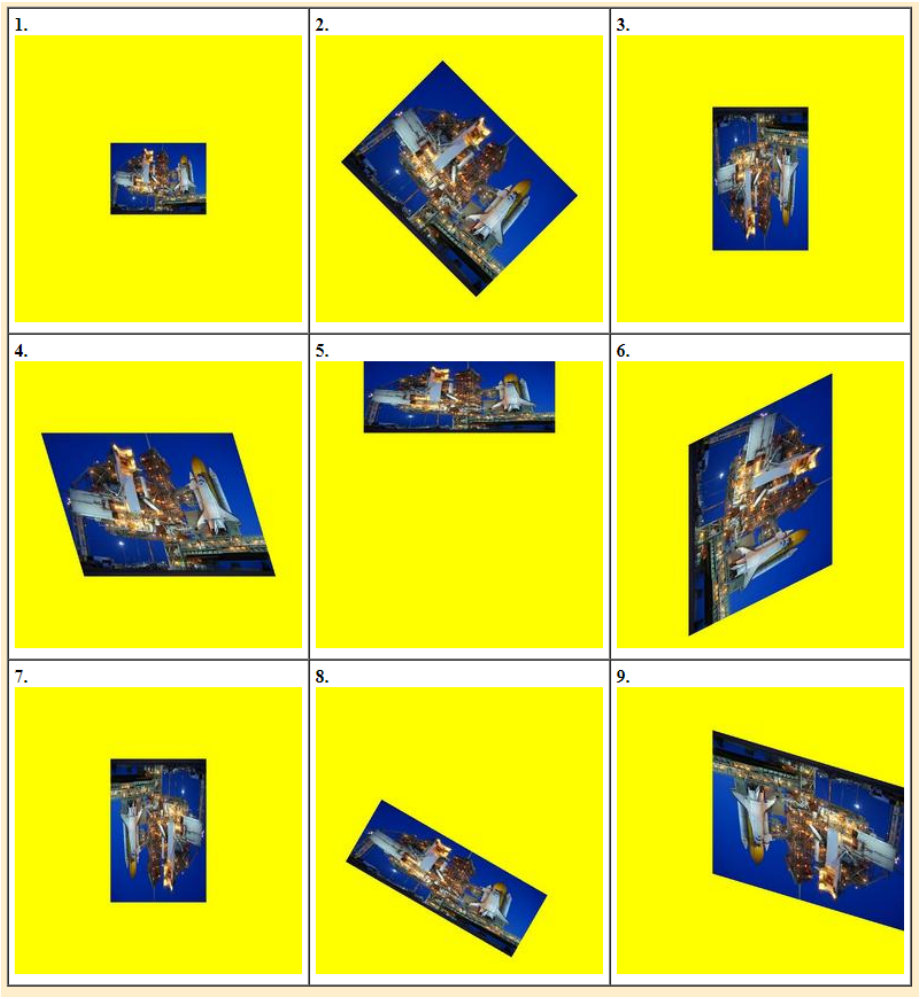

Figure 1: