GRAFICA in Java AWT si Java 2D
Implementarea transformarilor grafice

Afisarea intr-un applet

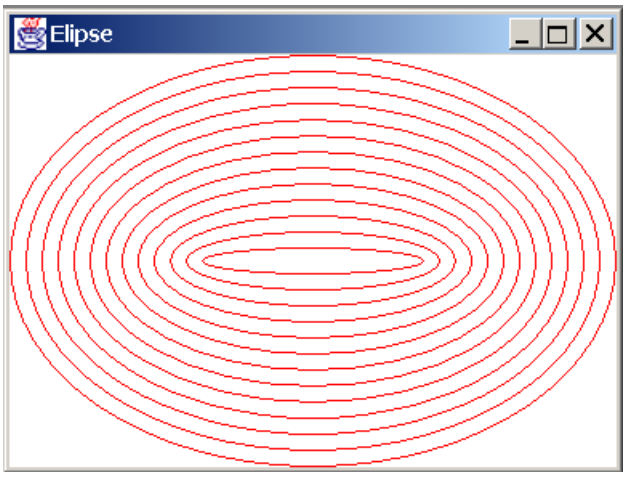
Applet → aplicatie bazata pe evenimente:

- orice prelucrare programata intr-un applet este executata ca raspuns la un eveniment
- Orice applet trebuie sa defineasca metodele prin care aplicatia raspunde la evenimente:
  - init() – apelata la momentul incarcarii applet-ului
  - paint() – apelata pentru afisarea / reafisarea continutului ferestrei applet-ului
  - destroy() – apelata la terminarea excutiei applet-ului
  - keyPressed(), keyReleased(), keyTyped()
  - mouseCliked(), mouseReleased(), si altele

Structura unui applet

```
import java.awt.*;
import java.applet.*;

public class myApplet extends Applet
{
  //  Variabile specifice aplicatiei
  int mouseX =0, mouseY =0;
  String click = ”Click”;

  public void init()
  {
     // Initializari la incarcarea applet-ului
  }

  public void paint(Graphics g)
  {
    // afisarea in fereastra aplicatiei
   // Apelata: imediat dupa init(),pentru afisarea initiala
      //  ori de cate ori continutul fereastrei applet-ului
      //  trebuie refacut
     g.drawString(click, mouseX, moseY);
  }
  public void mouseCliked(MouseEvent me)
  {
  // apelata atunci cand utilizatorul apasa si elibereaza
  // un buton al mouse-ului in aceeasi pozitie
  // me contine: pozitia cursorului
  //         informatii despre starea butoanelor mouse-ului
   mouseX = me.GetX();
   mouseY = me.GetY();
  }
  public void destroy()
  {
   // eliberare resurse
  }

}
```

```
import java.applet.*;
import java.awt.*;

public class Elipse extends Applet
{
public void paint(Graphics g)
{
g.setColor(Color.red);

Dimension d = getSize();
int xmax = d.width - 1, ymax = d.height - 1;

for (int x1=0, y1=0; x1<xmax/2-10; x1+=10, y1+=10)
  g.drawOval(x1, y1, xmax-2*x1, ymax-2*y1);

}
}
```



Atribute grafice

Atributele primitivelor grafice sunt definite in clasa **Graphics**. Ele au valori implicite dar pot fi modificate prin apelul unor functii specifice ale clasei Graphics.

Atribut	Tip	Descriere
`Culoarea`	`Color`	Culoarea folosita la afisarea primitivelor grafice.Poate fi definita folosind metoda `Setcolor()`
`Font-ul`	`Font`	Font-ul folosit la afisarea textelor
`Regiunea de decupare`	`Rectangle`	Zona la marginile careia sunt decupate primitivele grafice. Definita cu ajutorul functiilor `setClip()` si `clipRect()`
`Originea`	`Point`	Originea sistemului de coordonate folosit la desenare. Poate fi modificata folosind metoda `translate()`.
`Modul raster`	`Boolean`	Defineste modul de scriere in memoria ecran. Modul implicit este „prin suprascrierea pixelilor”. Poate fi selectat modul”XOR” folosind metoda `setXORMode()`. Pentru revenirea la modul implicit se poate folosi functia `setPaintMode()`.
`Culoarea de fond`	`Color`	Acest atribut este folosit numai de metoda `clearRect()` si nu poate fi modificat. In cazul desenarii pe suprafata unei componente, culoarea de fond este definita prin atributul `background` al componentei. Nu este

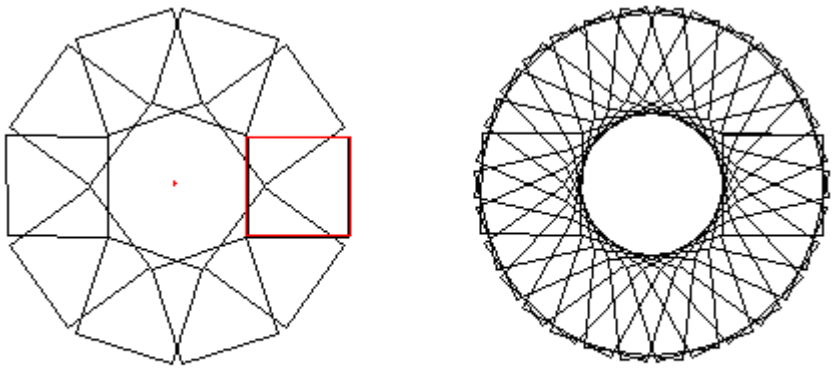
		folosit atunci cand desenarea are loc intr-o imagine in memorie.

Implementarea transformarilor grafice 2D intr-un aplet Java AWT

```
package transformari2D;

import java.applet.*;
import java.awt.*;

//creaza un desen rotind un patrat in jurul centrului ferestrei applet-ului

public class Rotatie extends Applet
 {

        double xc, yc; //punctul fix al transformarii
        double[] x, y; //coordonatele colturilor patratului


        double cosu, sinu, du;
        int nrRotatii=50,
        raza = 70, // raza cercului parcurs de centrul patratului
        latura = 50;//latura patratului


        public void init()
       {

          /*
            try
            {
               nrRotatii = Integer.parseInt(getParameter("nrRotatii"));
            }

           catch (NumberFormatException e)
           }

          System.out.println("Valoare eronata/nedefinita pt parametrul 'nrRotatii'");

          nrRotatii = 10;

       }

       */
```

```
        x = new double[4];
        y = new double[4];

        du = (double) (2 * Math.PI / nrRotatii); //pasul unghiular de rotatie

        cosu = Math.cos(du);
        sinu = -Math.sin(du);

        }


public void desen(Graphics g)
{
  // deseneaza patratul prin linii care conecteaza varfurile sale

    g.drawLine((int) x[0], (int) y[0], (int) x[1], (int) y[1]);
    g.drawLine((int) x[1], (int) y[1], (int) x[2], (int) y[2]);
    g.drawLine((int) x[2], (int) y[2], (int) x[3], (int) y[3]);
    g.drawLine((int) x[3], (int) y[3], (int) x[0], (int) y[0]);


}

public void rotatie()
{
 // aplica transformarea de rotatie varfurilor patratului
double xx;
for (int i = 0; i < 4; i++)
{// rotatie in jurul punctului (xc,yc)
        xx = x[i];

        x[i] = xx * cosu - y[i] * sinu + xc - xc * cosu + yc * sinu;

        y[i] = xx * sinu + y[i] * cosu + yc - xc * sinu - yc * cosu;
}

}


public void paint(Graphics g)
{
         xc = (double)getSize().width / 2.0;
         yc = (double)getSize().height / 2.0;
// initializeaza coordonatele varfurilor patratului
```

```
            x[0] = x[3] = xc + raza - latura / 2;
            x[1] = x[2] = x[0] + latura;
            y[0] = y[1] = yc - latura / 2;
            y[2] = y[3] = yc + latura / 2;

            g.setColor(Color.red);
// executa desenul

for (double u = 0; u < 2 * Math.PI; u += du)
{
            desen(g);
            rotatie();
}
}

}
```



Rotatia patratului in jurul unui punct

Animatia patratului – rotatia in jurul centrului ferestrei applet-ului

Modul raster – atribut grafic

- Defineste modul de scriere in memoria ecran.
- Modul implicit este „prin suprascrierea pixelilor”.

- Modul XOR→ operatia binara XOR intre valorile care trebuie sa fie scrise in memoria ecran si valorile existente in aceiasi pixeli
    - Poate fi selectat apeland metoda setXORMode().

Programul anterior se modifica astfel:

Se adauga functia :

```
public static void sleep(int ms)
 {
      try
      {
         Thread.currentThread().sleep(ms); // suspenda executia thread-ului
                                           curent pentru un interval de ms milisecunde
      }
      catch (InterruptedException e)
      {
      }
  }

public void paint(Graphics g)
{

……..
// in loc de ciclul pentru executia desenului
   g.setXORMode(Color.white); // fondul este alb
   for(int i=0; i<100; i++)
   {
     desen(g); // executa desen patrat
     sleep(100); // asteapta 100 milisecunde
     desen(g); // re-executa desenul – efectul: stergerea sa
     rotatie(); // trasforma coordonatele varfurilor
   }
   }
```

Java2D

- Ofera capabilitati sporite pentru grafica bidimensionala si pentru lucrul cu texte si imagini prin extensii ale AWT.
- Interfata de programare Java 2D furnizeaza:
    - un model de redare uniform pentru dispozitive ecran si imprimante
    - o colectie bogata de primitive geometrice
    - mecanisme pentru identificarea primitivelor grafice
    - un model de combinare a culorilor pixelilor la scrierea in memoria raster
    - suport sporit pentru culori
    - suport pentru tiparirea documentelor complexe

- Java 2D extinde clasa Graphics cu clasa **Graphics2D**.

Atributele grafice Java2D

Atributele care afecteaza afisarea primitivelor grafice sunt definite in clasa Graphics2D. Valorilele lor implicite pot fi modificate apeland functii ale aceleiasi clase.

Atribut	Tip	Descriere
`Culoarea`	`Color`	Atribut mostenit de la clasa `Graphics`. Reprezinta culoarea folosita la afisarea liniilor si a contururilor. Poate fi definita folosind metoda (mostenita) `Setcolor()`.
`Font-ul`	`Font`	Atribut mostenit de la clasa `Graphics`. Sunt realizate toate fonturile sistem.
`Regiunea de decupare`	`Shape`	Atribut mostenit de la clasa `Graphics`. Regiunea de decupare poate avea o forma arbitrara. Este

		definita cu ajutorul functiei `clip()`.
`Culoarea de fond`	`Color`	Atribut mostenit de la clasa **Graphics**. Folosit numai de metoda **clearRect()**. Poate fi setat/interogat folosind `setBackground()` / `getBackground()`.
`Stilul liniei`	`Stroke`	Specifica modul de desenare a liniilor si contururilor: latimea, sablonul, si altele. Poate fi specificat prin `setStroke()`.
`Stilul interiorului`	`Paint`	Specifica modul de generare a pixelilor interiori unei suprafete. Se poate specifica prin `setPaint()`.
`Regula de combinare a culorilor`	`Composite`	Specifica modul de combinare a culorii care trebuie inscrisa intr-un pixel cu culoarea curenta a pixelului. Tipul `Composite` este implementat de clasa `AlphaComposite`.
`Transformarea`	`Java.awt.geom.AffineTransform`	Defineste transformarea care se aplica coordonatelor transmise functiilor de afisare. Se defineste cu `setTransform()`, `translate()`, `scale()`, `rotate()`, `shear()`.
`Calitatea redarii`	`RenderingHints`	Specifica preferinta privind modul de generare a imaginilor. De exemplu, se poate cere efectuarea operatiei de antialiasing la generarea primitivelor grafice. Se defineste cu `setRenderingHints()`, `setRenderingHint()` sau `addRenderingHints()`.

Implementarea transformarile geometrice intr-un applet Java 2D

Atributul Transformare reprezinta o transformare afina (care conserva liniile drepte si paralelismul liniilor). Ea este o transformare liniara care se reprezinta matricial in coordonate omogene.

Matricea unei transformari afine 2D are forma generala:

$$M = \begin{bmatrix} sx & shx & tx \\ shy & sy & ty \\ 0 & 0 & 1 \end{bmatrix}$$

unde, sx si sy sunt factorii de scalare, shx si shy sunt factorii de forfecare (shearing) iar (tx, ty) exprima translatia.

Transformarea coordonatelor primitivelor grafice inainte de afisare este exprimata astfel:

$$\begin{bmatrix} xe \\ ye \\ 1 \end{bmatrix} = M^* \begin{bmatrix} xp \\ yp \\ 1 \end{bmatrix}$$

unde

- (xp, yp) reprezinta coordonata unei primitive, specificata in programul de aplicatie
- (xe, ye) reprezinta un punct in coordonate ecran

Functiile rotate(), scale(), shear() si translate() modifica matricea curenta a transformarii, prin acumularea transformarii:

- MC - matricea de transformare curenta (valoarea curenta a atributului transformare)
- MT – matricea corespunzatoare transformarii specificate prin parametrii de apel ai functiei de transformare
- M=MC*MT – valoarea atributului transformare dupa executia functiei de transformare

Exemplu:

```
public class Rotatie extends Applet
{
 Point2D.Float C;
 Point2D.Float[] patrat;
```

```
float cosu, sinu, du;
int nrRotatii = 10, latura = 50, raza = 70;

public void init()
 {
      super.init();

   // du este pasul unghiular de rotatie
      du = (float) (2 * Math.PI / nrRotatii);
      cosu = (float) Math.cos(du);
      sinu = (float) Math.sin(du);
   }


   public void paint(Graphics g)
 {
      Point2D.Float C;

      Graphics2D g2D = (Graphics2D) g;


      C = new Point2D.Float((float)getSize().width / 2.0f, getSize().height / 2.0f);
      patrat[0] = new Point2D.Float(C.x + raza - latura / 2, C.y - latura / 2);
      patrat[1] = new Point2D.Float(patrat[0].x + latura, patrat[0].y);
      patrat[2] = new Point2D.Float(patrat[1].x, C.y + latura / 2);
      patrat[3] = new Point2D.Float(patrat[0].x, patrat[2].y);


       g2D.setColor(Color.red);

// Aici, transformea curenta este cea identica

       for (double u = 0; u < 2 * Math.PI; u += du)
       {
          desen(g2D);
// acumuleaza la transformarea curenta o rotatie de unghi du in jurul punctului C
          g2D.rotate(du, C.x, C.y);
       }
   }
```