

# Zeichnen und animieren

- Grundlage bildet wieder die Klasse Graphics
- Ein Objekt der Klasse Graphics wird immer der paint Methode übergeben

```
public void paint (Graphics g)
```

- Auch außerhalb von paint kann Graphics verwendet werden, über die Methode getGraphics() von Component erhält man das Graphicsobjekt. Voraussetzung ist aber, dass die Komponente bereits sichtbar ist.
- Die Sichtbarkeit einer Komponente lässt sich nur mit einem componentShown -event sicherstellen.

# Zeichenfunktionen der Klasse Graphics (nicht vollständig)

## zeichnen

- drawLine
- drawArc
- drawString
- drawOval
- drawRect
- drawRoundRect
- drawPoygon
- DrawImage
- DrawString

## Flächen füllen

- fillOval
- fillArc
- fillRect
- fillRoundRect
- fillPoygon

# Einstellungen vornehmen

- **setFont**

- `setFont(new Font("Courier", Font.PLAIN, 20));`

- **SetColor**

- `setColor(Color.red);`

- `setColor(new Color(0x5050FF)); // hellblau`

- `setColor(new Color(0x50, 0x50, 255));`

- **setClip(x,y,w,h);**

- `setClip(x, y, w, h);`

# Arbeit mit Schrift

- drawString
  - `drawString("Sommer 2015", xpos, ypos);`
- getFontMetrics, getFontMetrics(Font f)
  - getAscent, getDescent, getLeading, getHeight
  - `stringWidth("Javaspass 2015");`

# Graphics, erstes Beispiel

```
import java.awt.*;
import java.awt.event.*;

import java.util.*;
class TestGr extends Panel
{
    . . .

    public static void main(String args[])
    {
        Frame f=new Frame();
        f.add(new TestGr());
        f.setVisible(true);
        f.setSize(500,500);
        f.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e){System.exit (0);}
        });
    }
}
```

```
Vector<MyLine> V=new Vector<MyLine>();  
int x,y;  
  
class MyLine  
{  
    int x1,y1,x2,y2;  
    MyLine(int x1, int y1, int x2, int y2)  
    {  
        this.x1=x1;  
        this.y1=y1;  
        this.x2=x2;  
        this.y2=y2;  
    }  
}  
  
public void paint(Graphics g)  
{  
    for(MyLine l:V){ g.drawLine(l.x1,l.y1,l.x2,l.y2);}  
}
```

```
public TestGr()  
{  
    addMouseListener(new MouseAdapter()  
    {  
        MyLine l;  
        public void mousePressed(MouseEvent e)  
        {  
            x=e.getX(); y=e.getY();  
        }  
        public void mouseReleased(MouseEvent e)  
        {  
            l=new MyLine(x, y, e.getX(), e.getY());  
            V.add(l);  
            repaint();  
        }  
    });  
}
```

# Hubschrauber

- An Hand der Spielerei eines animierten Hubschraubers werden eine ganze Reihe von draw- und fillmethoden und Animation demonstriert.
- Die Aufrufe der verschiedenen drwafunktionen erfolgt in der paint-Methode
- Es gibt zwei Methoden drawBackground , die wahlweise implementiert werden müssen.



# Ein zweites Beispiel: Hubschrauber

```
import java.awt.*;
import java.awt.event.*;

class Helicopt extends Panel
{
    int x=50, y=30;
    . . .

    public static void main(String args[])
    {
        Frame f=new Frame();
        f.setSize(500,300);
        Helicopt p=new Helicopt();
        f.add(p);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter()
            {public void windowClosing(WindowEvent e)
                {System.exit(0);}});
    }
}
```

```
public void paint(Graphics g)
{
    Polygon P=new Polygon();
    P.addPoint(x+4,y+15);
    P.addPoint(x+35,y+15);
    P.addPoint(x+35,y+25);
    drawBackground(g, c1, c2, 128);
    g.setColor(Color.black);
    Color C=g.getColor();
    g.setColor(Color.red);
    g.fillPolygon(P);
    g.fillOval(x,y+10,10,10);
    g.fillRect(x+41,y+3,5,7);
    g.fillRect(x+20,y+1,50,3);
    g.fillOval(x+30,y+10,26,26);
    g.fillOval(x+24,y+31,8,8);
    g.fillOval(x+49,y+31,8,8);
    g.setColor(Color.blue);
    g.fillArc(x+31,y+11,24,24,0,90);
    g.fillArc(x+31,y+11,24,24,0,-75);
    g.setColor(C);
}
```

```
Color c1 = new Color(0x5050ff); // Blau
Color c2 = new Color(0xffffffff); // Weiss

    public Dimension getPreferredSize()
    {
        return new Dimension(500,300);
    }

void drawBackground(Graphics gr,
                    Color c1,
                    Color c2,
                    int numsteps)
{
    gr.setColor(Color.blue);
    Dimension size = this.getSize();
    gr.fillRect(0,0,size.width,size.height);
}
```

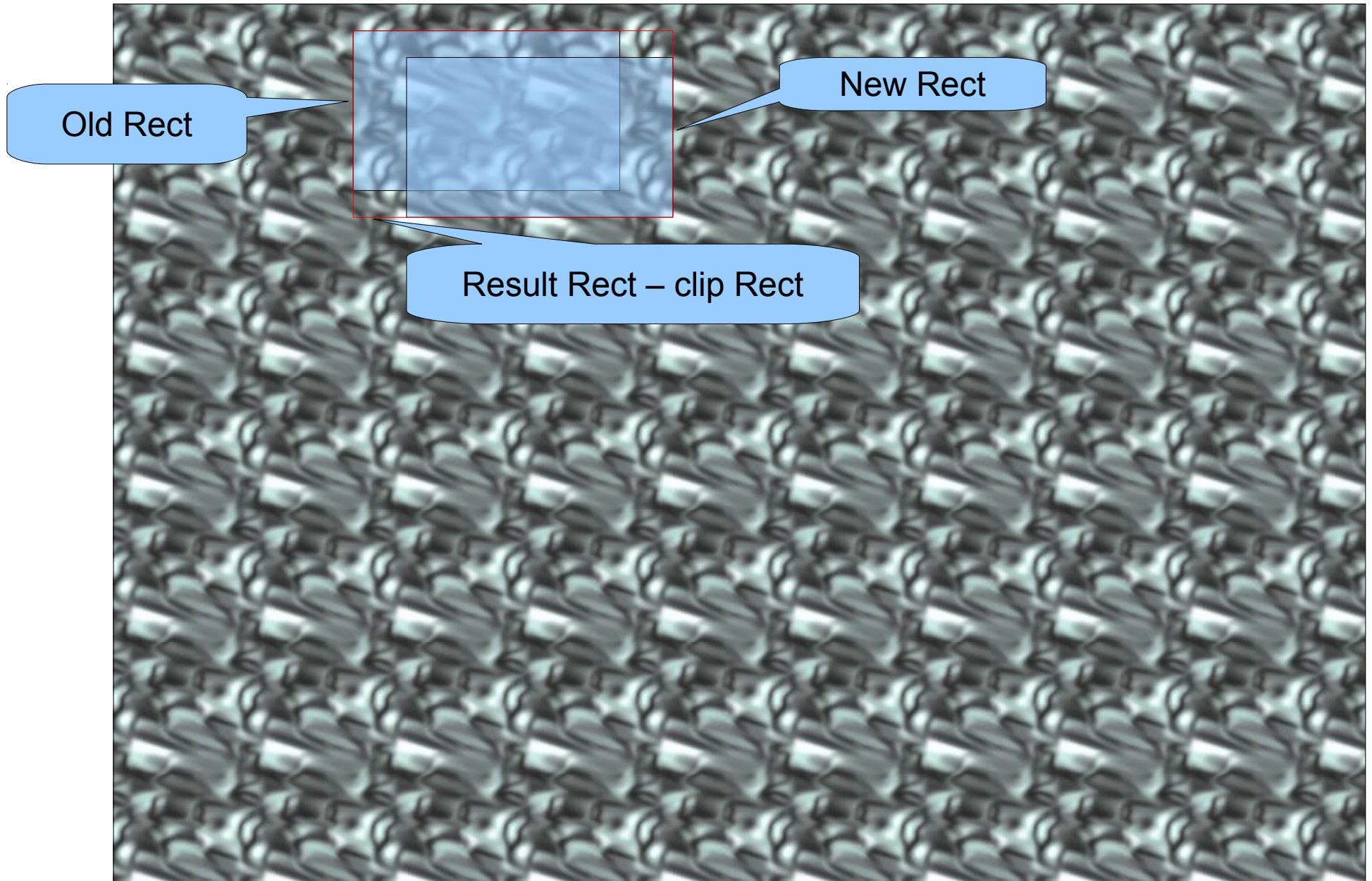
```
void drawBackground(Graphics gr,
                    Color c1,
                    Color c2,
                    int numsteps)
{
    int r, g, b;
    int dr = (c2.getRed()    - c1.getRed())    /numsteps;
    int dg = (c2.getGreen()  - c1.getGreen())  /numsteps;
    int db = (c2.getBlue()   - c1.getBlue())   /numsteps;
    Dimension size = this.getSize();
    int w = size.width, h = size.height;
    int dw = size.width/numsteps;
    int dh = size.height/numsteps;
    gr.setColor(c1);
    gr.fillRect(0, 0, w, h);
    for(r=c1.getRed(),g=c1.getGreen(),b=c1.getBlue();
        h > 0;
        h -= dh, w -= dw, r += dr, g += dg, b += db)
    {
        gr.setColor(new Color(r, g, b));
        gr.fillOval(0,0, w, h);
    }
}
```

# Er soll fliegen

- Animationen laufen in der Regel in einem gesonderten Thread.
- In jedem Animationsschritt muss der Hubschrauber an seiner neuen Position gezeichnet werden. Der Hintergrund an der Stelle, wo er vorher war, muss rekonstruiert werden.
- Um eine möglichst ruhige Animation zu erhalten, sollte das Malen des Hintergrundes und des animierten Objektes im Verborgenen erfolgen.
- Man bedient sich eines sog. Offscreens, einem Image, in das man malt, statt auf den Bildschirm auszugeben.

- Nach Fertigstellung wird das gesamte Bild dann zur Ausgabe gebracht
- Zur Minimierung des Datentransfers kann der Datentransport auf das Rechteck, in dem sich Bilddaten geändert haben eingeschränkt werden.
- Dazu Berechnung des Rechtecks, in dem etwas animiert wurde aus der Vereinigung des Rechtecks, das das alte Bild und des Rechtecks, das das neue Bild umschließt.
- Einrichtung einer Clipregion für das resultierende Rechteck.

# Ein Animationsschritt:



- Benötigt wird also:
  - Thread
  - Offscreenimage
  - Clipregion



```

class HelicoptAnim extends Panel implements Runnable
{
    int x=50, y=30;           // Anfangsposition
    Color c1 = new Color(0x5050ff); // hellblau
    Color c2 = new Color(0xffffffff); // weiss
    Image offscreen;
    Thread animator = null;
    boolean please_stop = false;
    static final int deltax = 2; // Animationsschritt
    static final int deltay = 1; // x += 2, y += 1
    . . . (nächste Folie)

    public static void main(String args[])
    {
        Frame f=new Frame();
        f.setSize(500,300);
        HelicoptAnim p=new HelicoptAnim();
        p.setComponentListener(f);
        f.add(p);
        f.setVisible(true);
        f.addWindowListener(new WindowAdapter(). . .);
    }
}

```

```
public HelicoptAnim()
{
    animator=new Thread(this);
    animator.start();
}

void setComponentListener(Frame f)
{
    f.addComponentListener(new ComponentAdapter()
    {
        @Override
        public void componentShown(ComponentEvent e)
        {System.out.println("notify");
        synchronized(animator){animator.notify();}
        }
    });
}
```

```
public void run() {
    int imageWidth=0,
        ImageHeight=0; // Bildgroesse
    int animWidth=70,
        AnimHeight=40; // Groesse des animierten Objektes

    // Wait, bis Panel sichtbar ist,
    // erst dann funktioniert getGraphics
    try
    {
        synchronized(animator){animator.wait();};
    }catch(Exception e){System.out.println(e);}

    . . . → next Page

    animator = null; // Das ist das Ende !!
}
```

```
while(!please_stop)
{
    Dimension d = this.getPreferredSize();

    if ((offscreen == null) ||
        ((imageWidth != d.width) || (imageHeight != d.height)))
    {
        offscreen    = this.createImage(d.width, d.height);
        imageWidth   = d.width;
        imageHeight  = d.height;
        System.out.println("offscreen created");
    }

    . . . → next page Animationsschritt ausführen

    // warten ...
    try {Thread.sleep(25);} catch (InterruptedException e){}
}
```

```
// altes Rechteck
Rectangle oldrect = new Rectangle(x, y, animWidth, animHeight);
// Update the coordinates for animation.
x = ((x + deltax)%d.width);
y = ((y + deltay)%d.height);

// neues Rechteck
Rectangle newrect = new Rectangle(x, y, animWidth, animHeight);

// Union Rechteck
Rectangle r = newrect.union(oldrect);

// Clip region einstellen
Graphics g = offscreen.getGraphics();
g.clipRect(r.x, r.y, r.width, r.height);

// in das off-screen image zeichnen
paint(g);

// Ausgabe der Clipregion
g = this.getGraphics();
g.clipRect(r.x, r.y, r.width, r.height);
g.drawImage(offscreen, 0, 0, this);
```