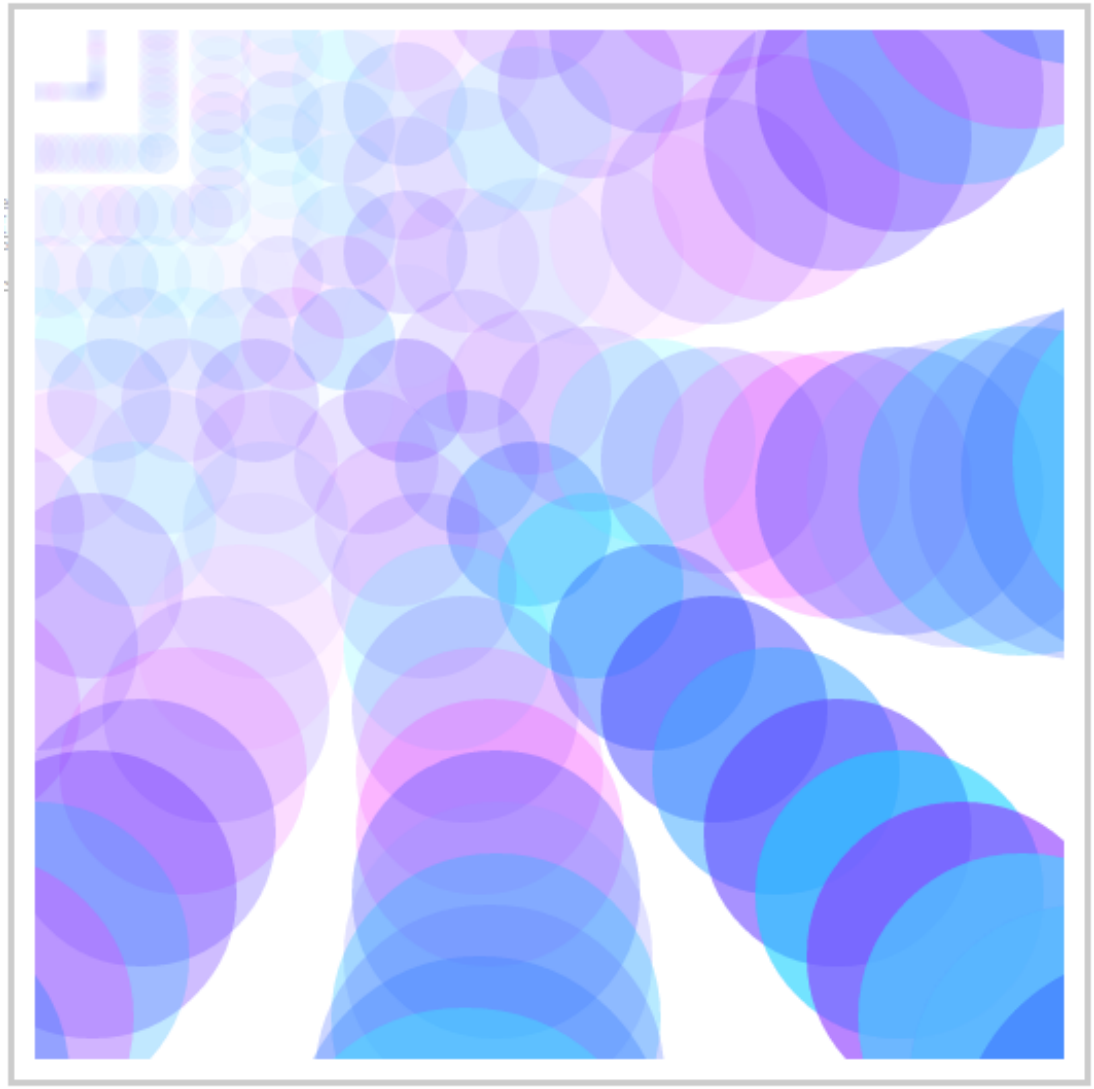
Raimond Reichert

Einführung in die Verwendung von Processing innerhalb von Eclipse

Grafikprogrammierung mit Processing

Beispiel:
«Blue Purple Circle pattern»



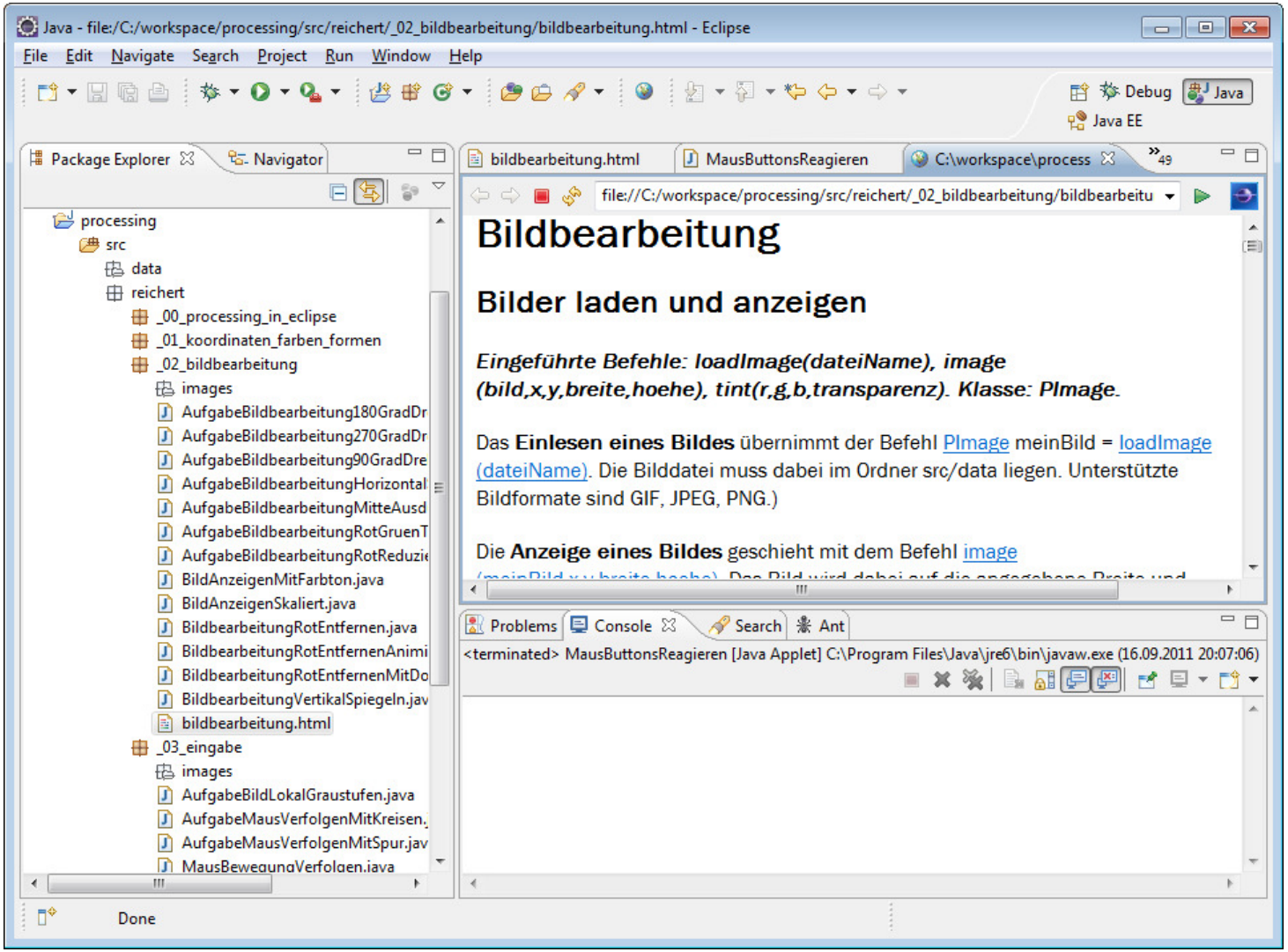
openprocessing.org/
visuals/?visualID=37337

Beispiel:
«Blue Purple Circle pattern»

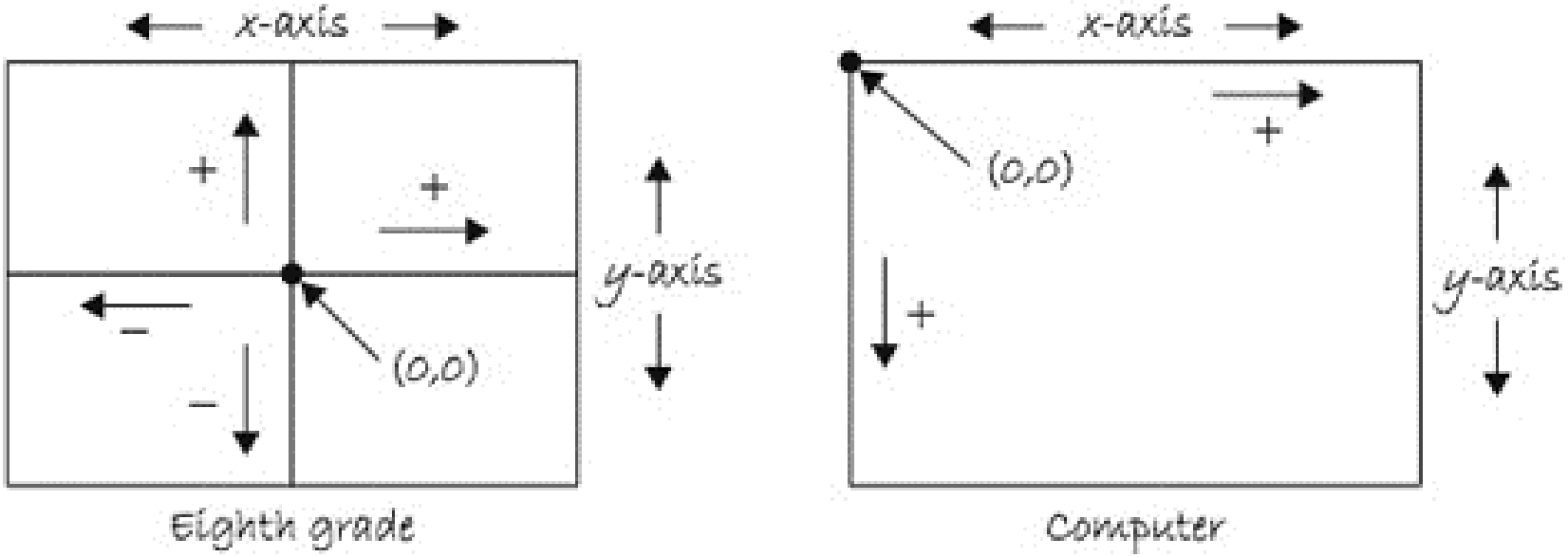
```
int distance = 30;
size(500, 500);
background(255, 255, 255);

for (int i=1; i<=width; i++) {
  noStroke();
  ellipse(distance*i, i*distance, 10*i, 10*i);
  for (int j=0; j<=width; j++) {
    fill(random(256), random(256), 255, 5*i);
    ellipse((distance*i)-(i*i*j), i*distance, i*10, i*10);
    ellipse(i*distance, (distance*i)-(i*i*j), 10*i, 10*i);
  }
}
```

openprocessing.org/visuals/?visualID=37337

Eclipse Projekt für Processing mit Skript und Aufgaben



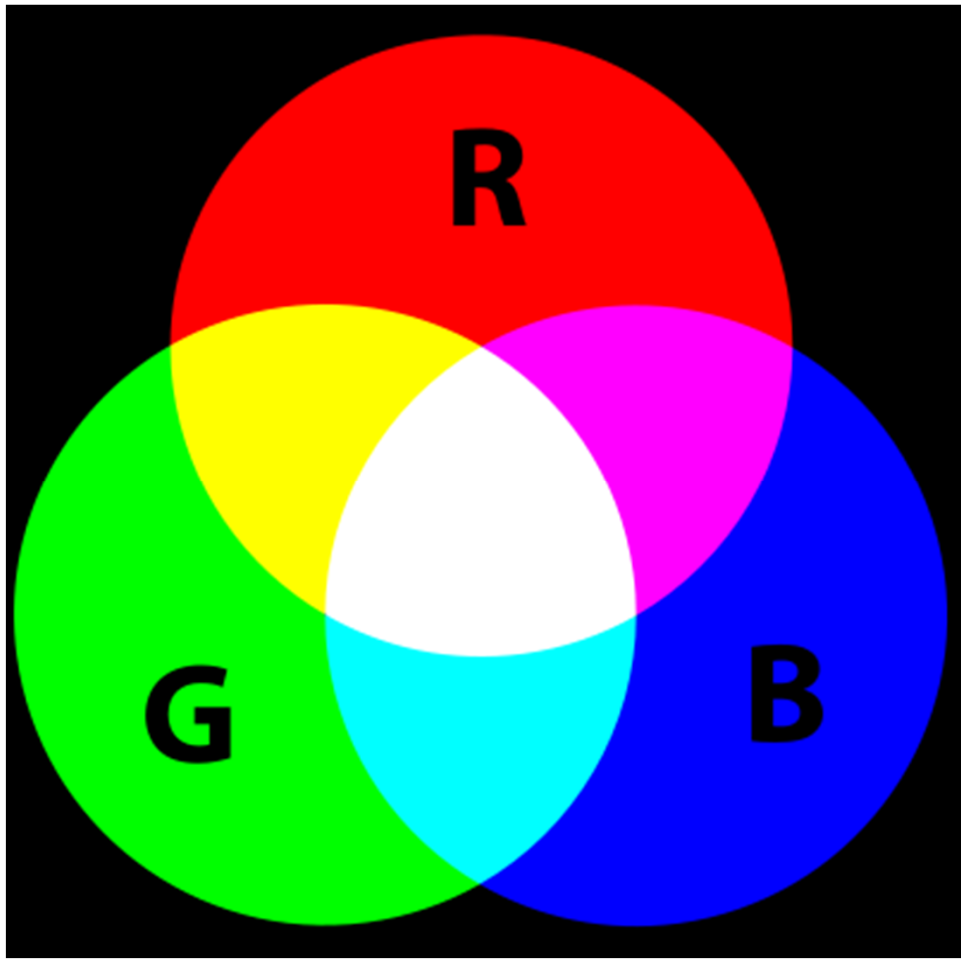
Koordinatensystem




```
size(500, 500); // Breite, Höhe definieren
line(0, 0, width, height); // Diagonale von links oben nach rechts unten
// width enthält aktuelle Breite des Programmfensters,
// height aktuell Höhe
```

Bild von processing.org

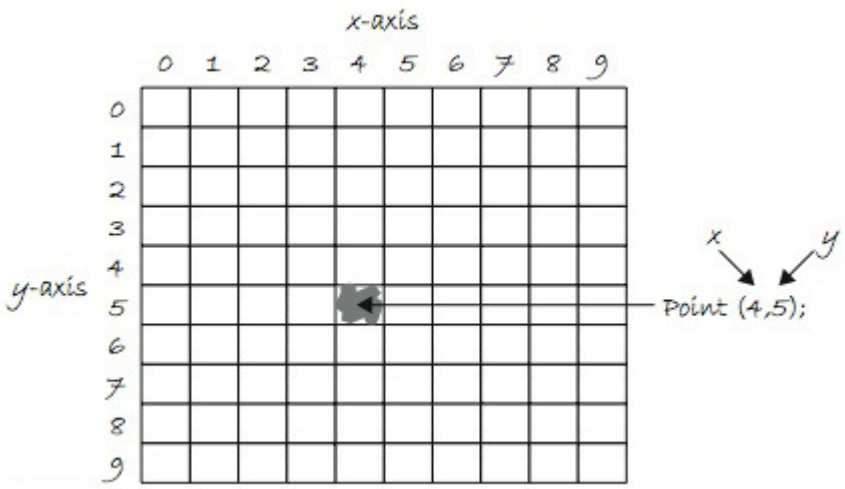
Farben: Mischung aus Rot-, Grün-, Blauanteilen

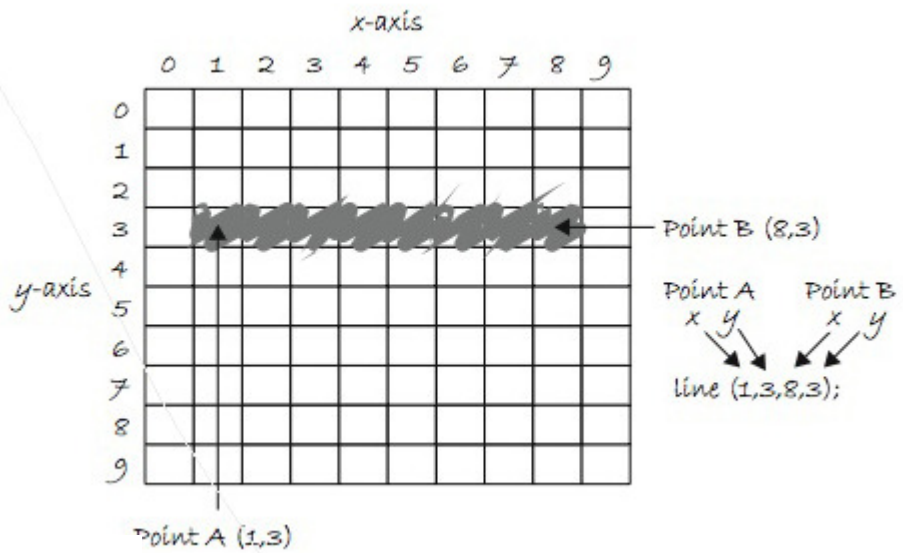


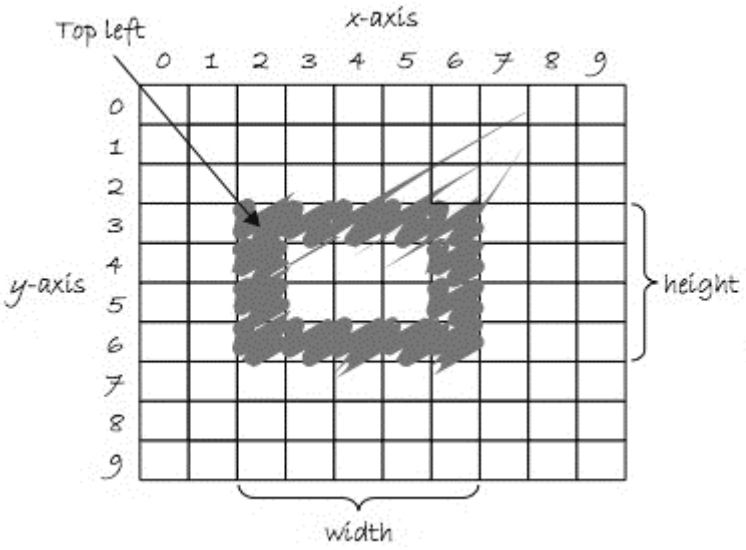

```
// Hintergrundsfarbe
background(255,0,0); // 100% rot
background(0,255,0); // 100% grün
background(0,0,255); // 100% blau

// Rahmenfarbe
stroke(255,255,255); // weiss
noStroke(); // keine Rahmenfarbe

// Füllfarbe
fill(255,255,0); // gelb
fill(255,255,0,128);
// gelb, halbtransparent
noFill(); // keine Füllfarbe
```

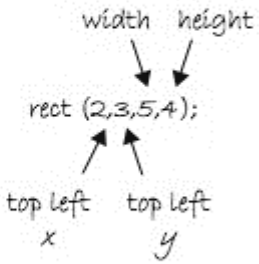
Bild von processing.org

Formen: Punkte, Linien, Rechtecke





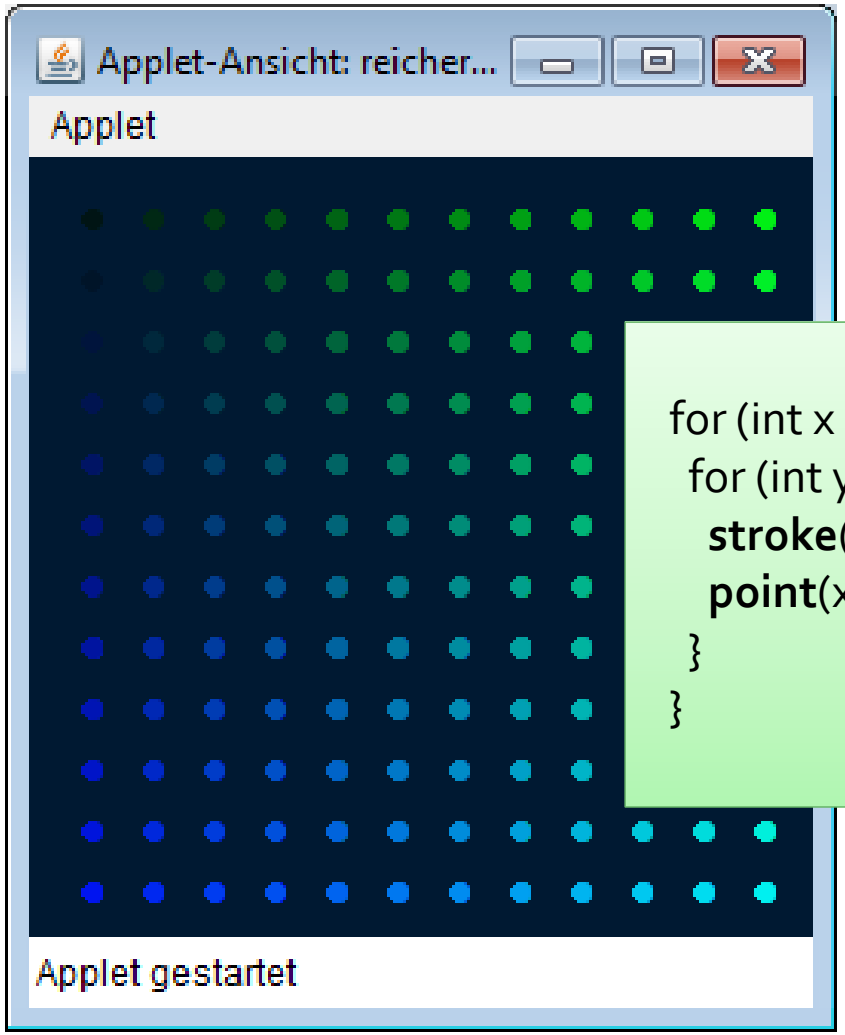





```
point(4,5);
line(1,3,8,3);
rect(2,3,5,4);
```

Bild von processing.org

Beispiel: Punkte im Gitter setzen




```
for (int x = ABSTAND; x < BREITE; x = x + ABSTAND) {
  for (int y = ABSTAND; y < HOEHE; y = y + ABSTAND) {
    stroke(o, x, y); // Grün-Blau-Mischung
    point(x, y);
  }
}
```

Bildbearbeitung: Pixel für Pixel

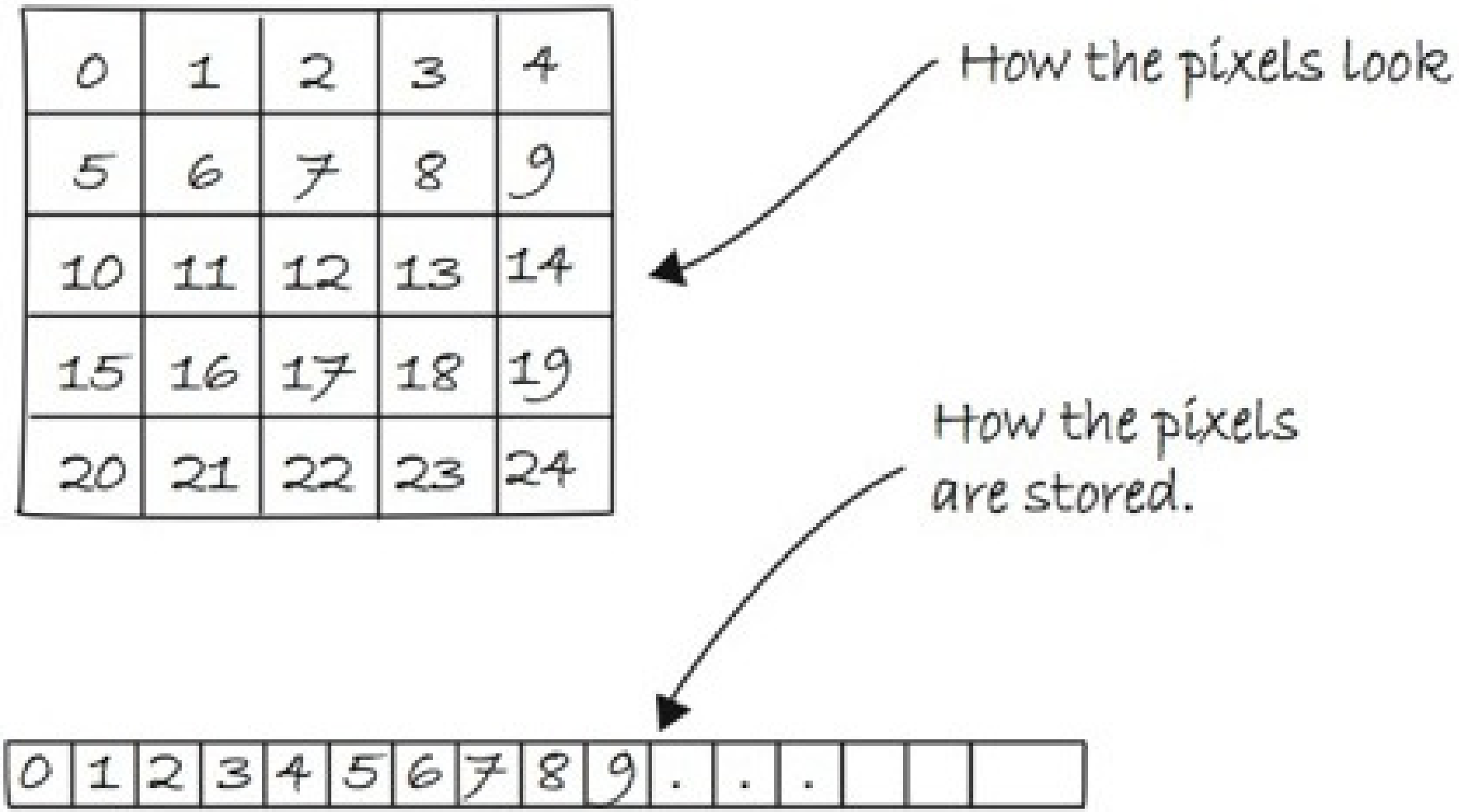
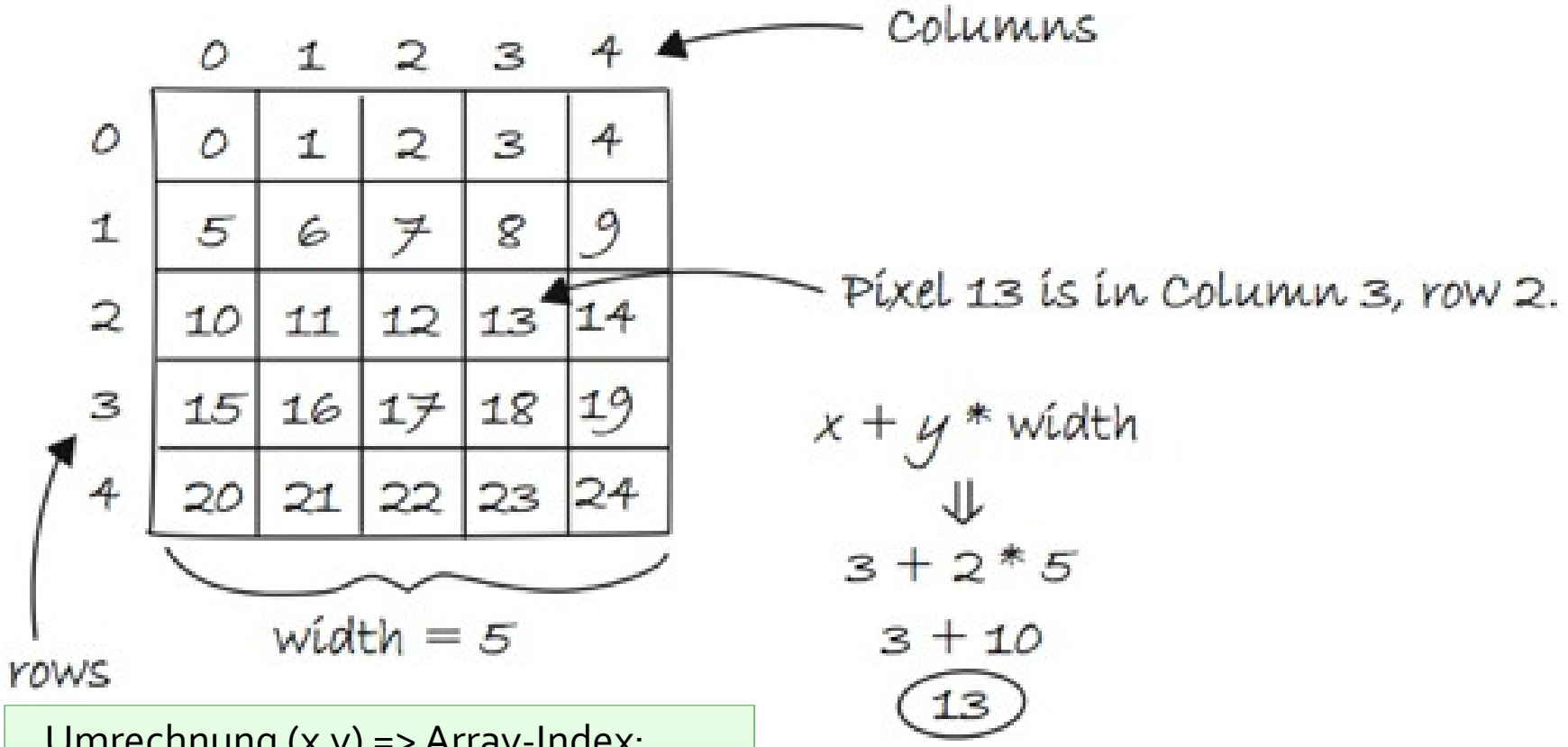


Bild von processing.org

Bildbearbeitung: Pixel für Pixel

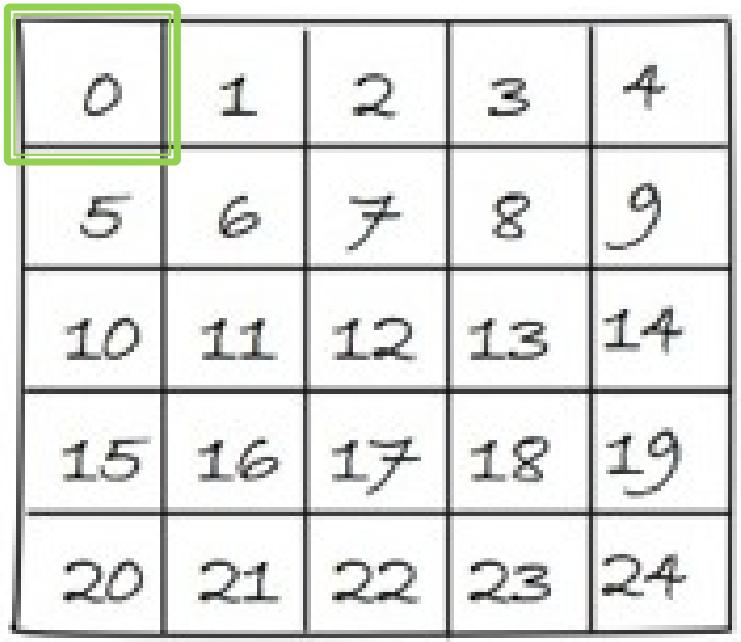


Umrechnung (x,y) => Array-Index:
index = y * **width** + x;

Umrechnung Array-Index => (x,y):
x = index % **width**; y = index / **height**;

Bild von processing.org

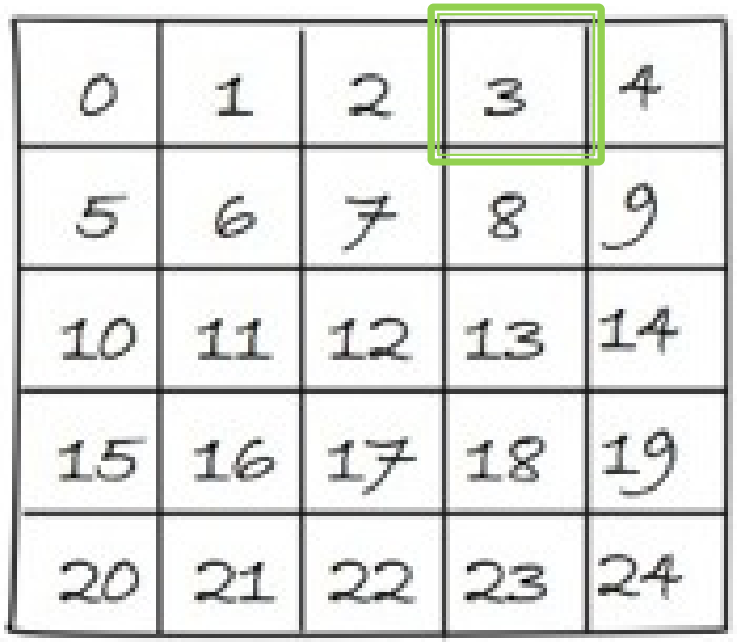
Bildbearbeitung: Pixel für Pixel



x = 0, y = 0
=> index = 0

Bild von processing.org

Bildbearbeitung: Pixel für Pixel



x = 3, y = 0
=> index = 3

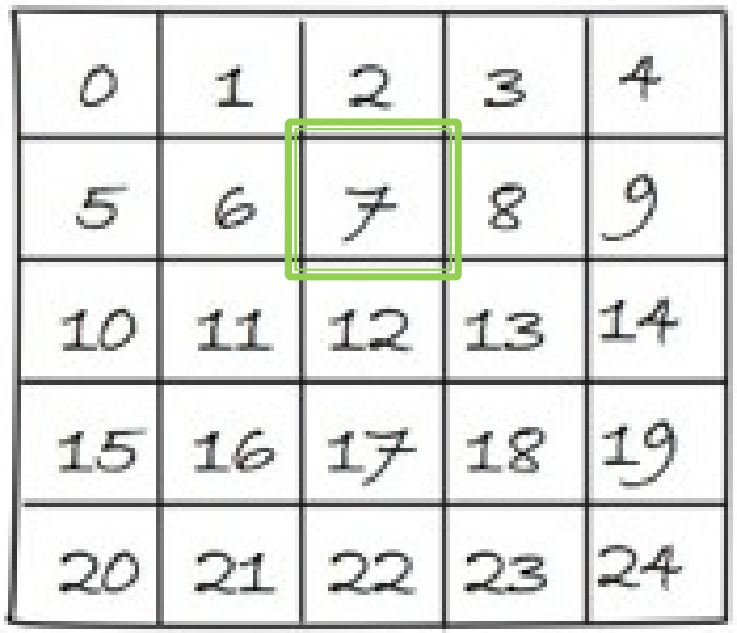
Bild von processing.org

Bildbearbeitung: Pixel für Pixel

x = 0, y = 1
=> index = 5 = width + 0

Bild von processing.org

Bildbearbeitung: Pixel für Pixel



x = 2, y = 1
=> index = 7 = width + 2

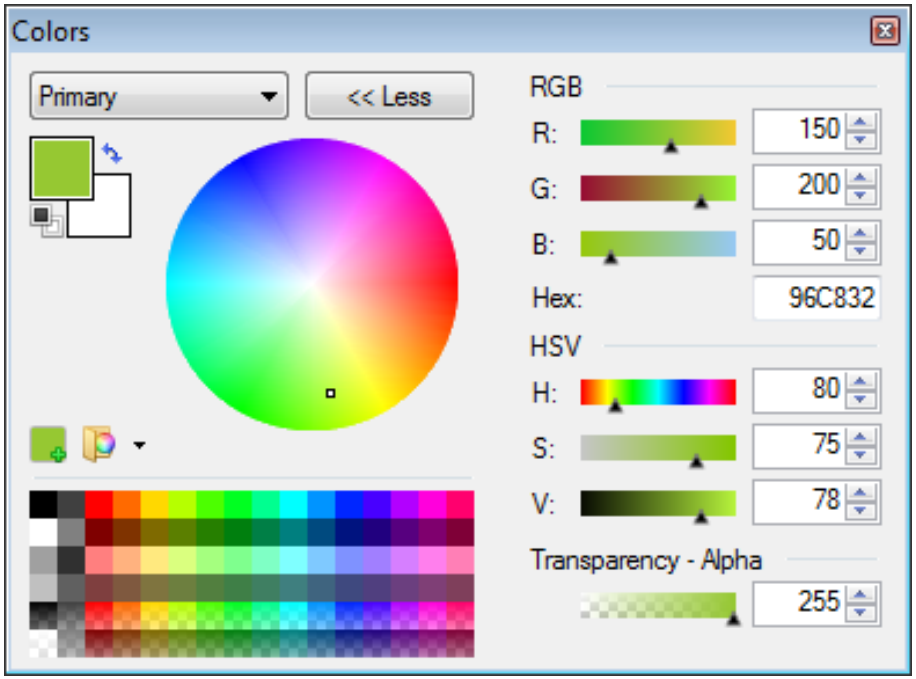
Bild von processing.org

Bildbearbeitung: Pixel für Pixel

x = 0, y = 2
=> index = 10 = width * 2 + 0

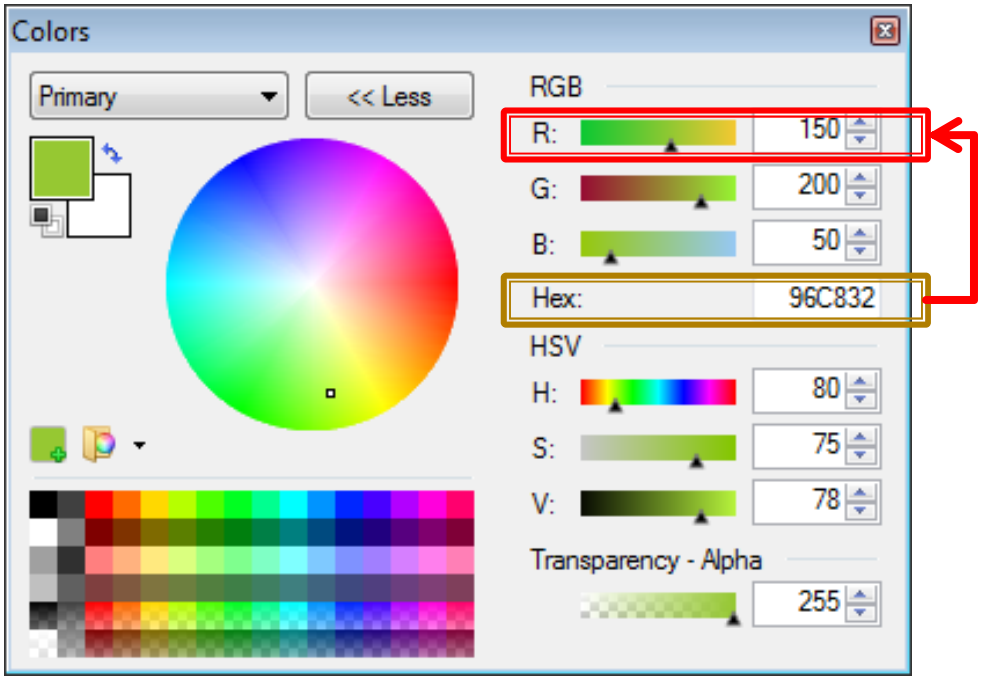
Bild von processing.org

Bildbearbeitung: Farbe eines Pixels



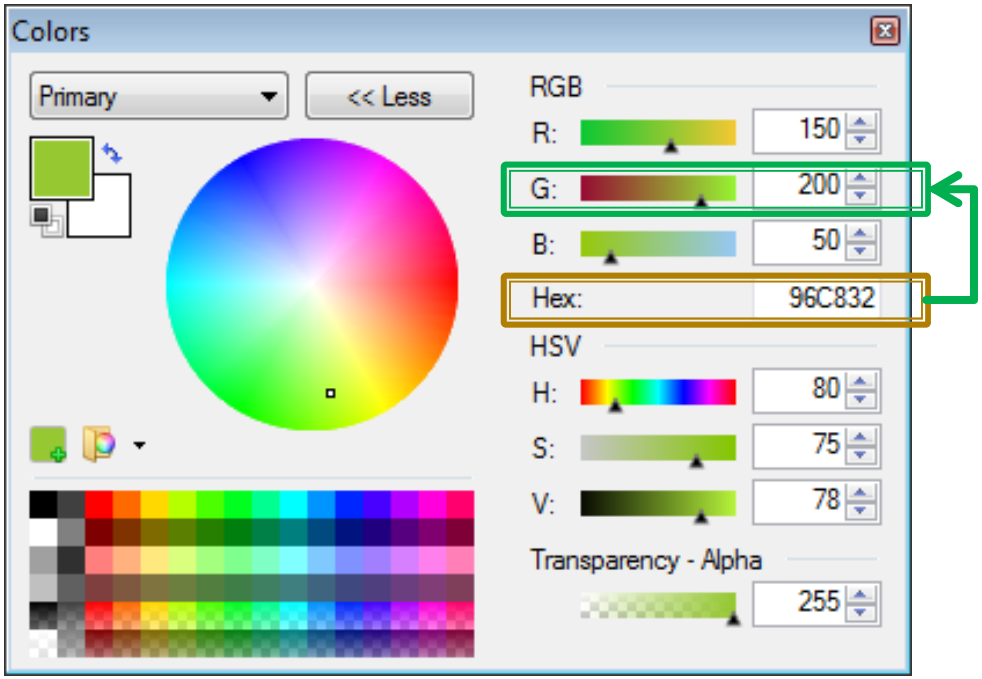

```
int farbe = bild.pixels[index];          // hex. 96 C8 32
int rot = red(farbe);                    // hex 96 = dec. 150
int gruen = green(farbe);                // hex C8 = dec. 200
int blau = blue(farbe);                  // hex 32 = dec. 50

int invertierteFarbe = color(255-rot, 255-gruen, 255-blau);
```

Bildbearbeitung: Farbe eines Pixels



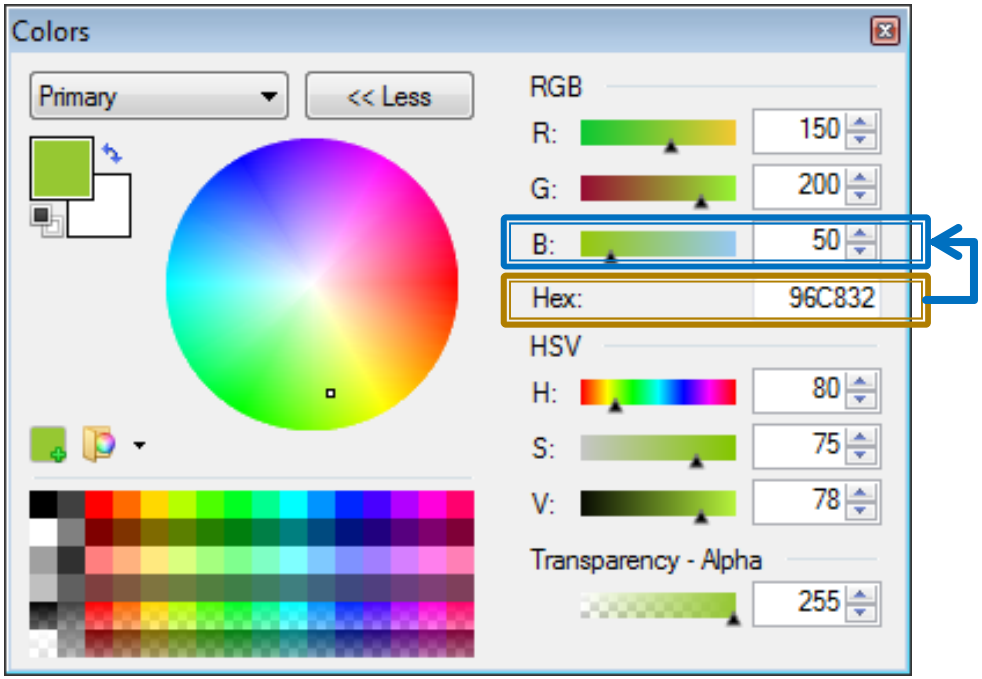

```
int farbe = bild.pixels[index];          // hex. 96 C8 32
int rot = red(farbe);                    // hex 96 = dec. 150
int gruen = green(farbe);                // hex C8 = dec. 200
int blau = blue(farbe);                  // hex 32 = dec. 50

int invertierteFarbe = color(255-rot, 255-gruen, 255-blau);
```

Bildbearbeitung: Farbe eines Pixels



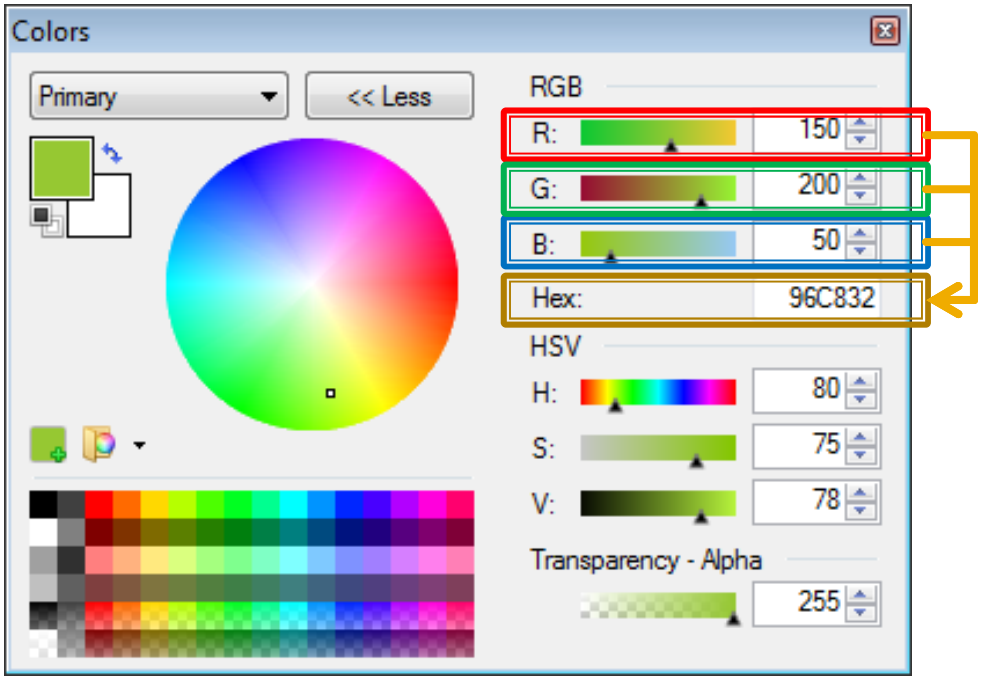

```
int farbe = bild.pixels[index];          // hex. 96 C8 32
int rot = red(farbe);                    // hex 96 = dec. 150
int gruen = green(farbe);                // hex C8 = dec. 200
int blau = blue(farbe);                  // hex 32 = dec. 50

int invertierteFarbe = color(255-rot, 255-gruen, 255-blau);
```

Bildbearbeitung: Farbe eines Pixels



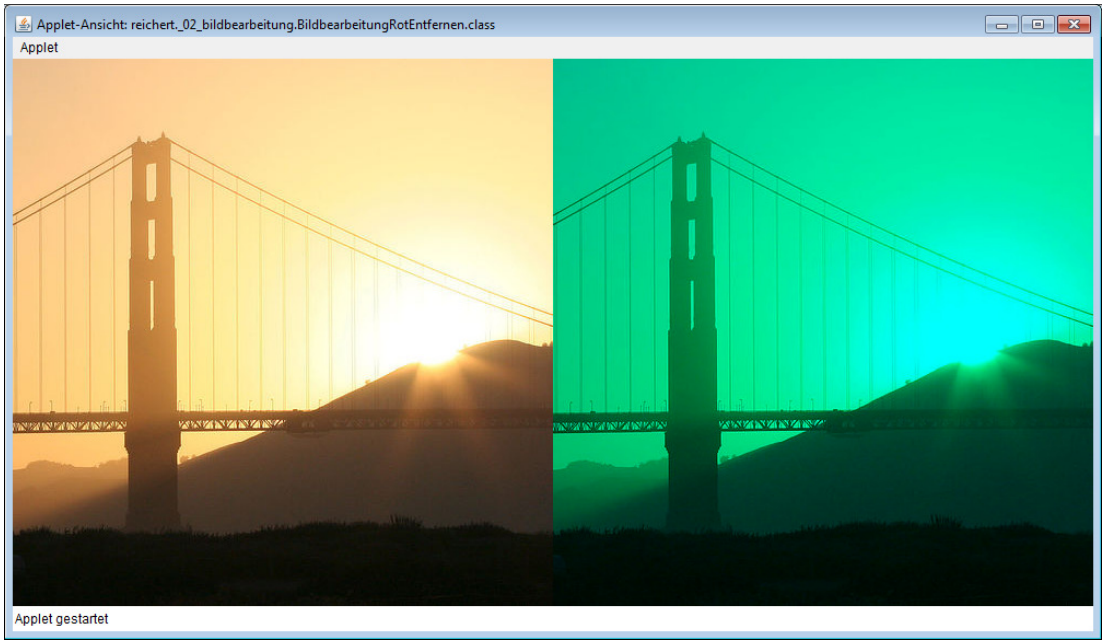

```
int farbe = bild.pixels[index];          // hex. 96 C8 32
int rot = red(farbe);                    // hex 96 = dec. 150
int gruen = green(farbe);                // hex C8 = dec. 200
int blau = blue(farbe);                  // hex 32 = dec. 50

int invertierteFarbe = color(255-rot, 255-gruen, 255-blau);
```

Bildbearbeitung: Farbe eines Pixels



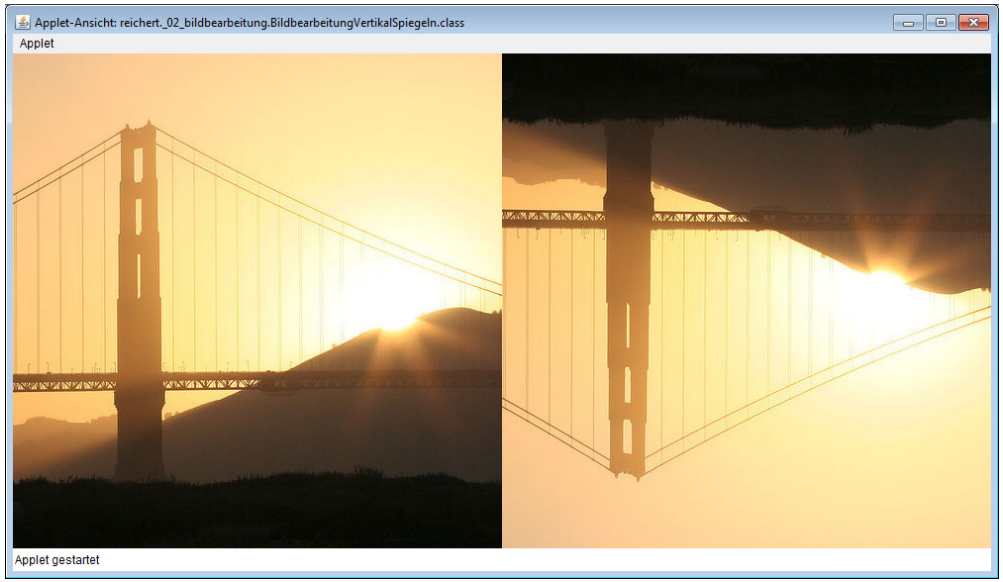

```
int farbe = bild.pixels[index];          // hex. 96 C8 32
int rot = red(farbe);                    // hex 96 = dec. 150
int gruen = green(farbe);                // hex C8 = dec. 200
int blau = blue(farbe);                  // hex 32 = dec. 50

int invertierteFarbe = color(255-rot, 255-gruen, 255-blau);
```

Bildbearbeitung: Farbveränderung



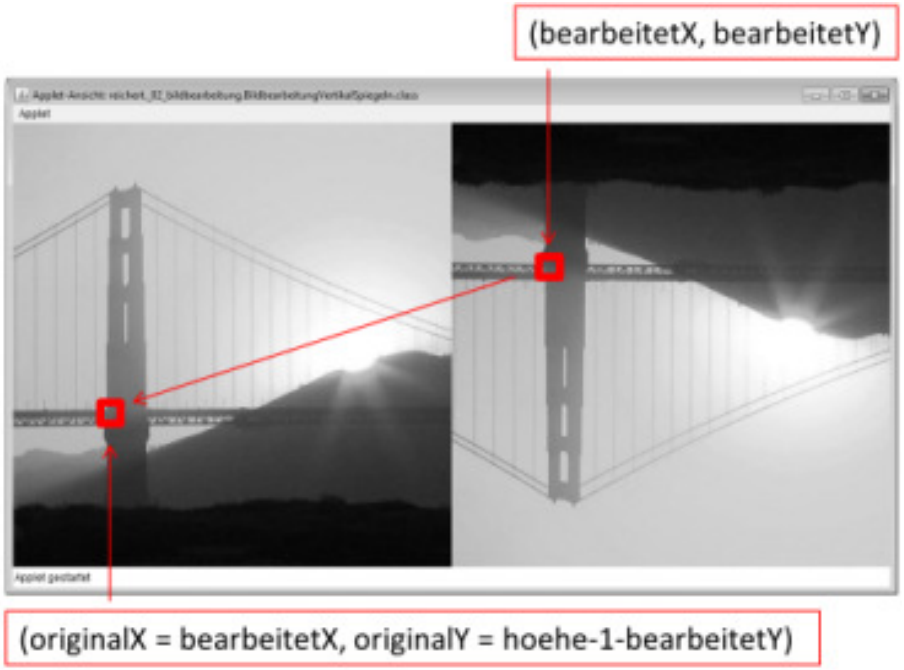

```
for (int i = 0; i < original.pixels.length; i++) {
  float gruen = green(original.pixels[i]);
  float blau = blue(original.pixels[i]);

  bearbeitet.pixels[i] = color(0, gruen, blau);
}
```

Bildbearbeitung: Strukturveränderung




```
for (int bearbeitetY = 0; bearbeitetY < bearbeitet.height; bearbeitetY++) {
  for (int bearbeitetX = 0; bearbeitetX < bearbeitet.width; bearbeitetX++) {
    int originalX = bearbeitetX;
    int originalY = bearbeitet.height - 1 - bearbeitetY;
    int originalFarbe = getColor(original, originalX, originalY);
    setColor(bearbeitet, bearbeitetX, bearbeitetY, originalFarbe);
  }
}
```

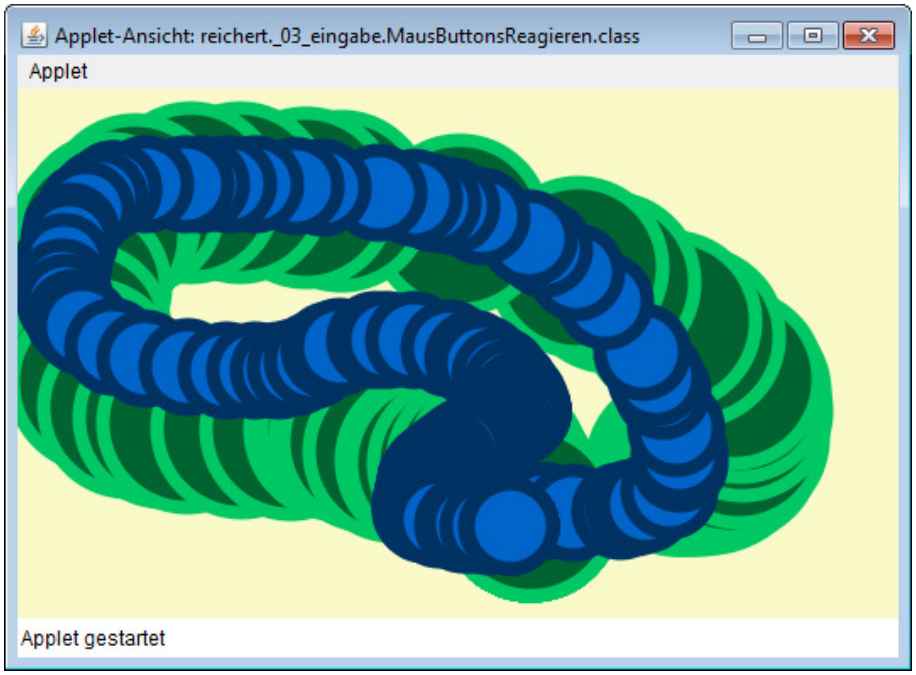
Bildbearbeitung: Strukturveränderung



Für jeden Pixel an den Koordinaten (bearbeitetX, bearbeitetY) im neuen Bild:

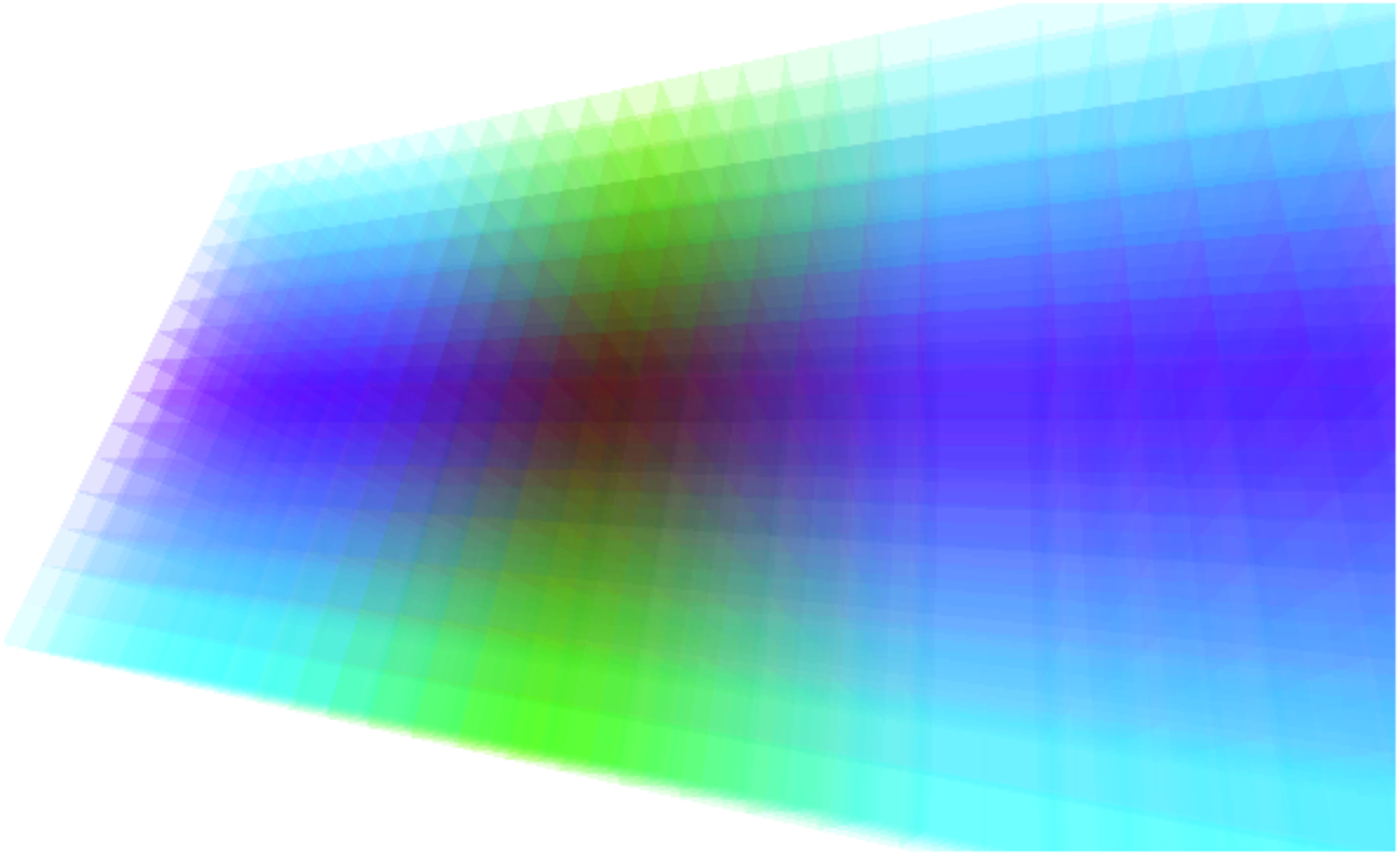
Berechne, von welchen Koordinaten (originalX, originalY) im Originalbild die Farben für diesen Pixel gelesen verwendet werden sollen.

Interaktionen mit der Maus




```
if (mousePressed) {
  if (mouseButton == LEFT) {
    stroke(0, 200, 100);
    fill(0, 100, 50);
    ellipse(mouseX, mouseY, 100, 100);
  } else if (mouseButton == RIGHT) {
    stroke(0, 50, 100);
    fill(0, 100, 200);
    ellipse(mouseX, mouseY, 50, 50);
  }
}
```

Und vieles, vieles mehr: processing.org/learning



processing.org/learning/3d/cubicgrid.html