### 23.2.4.10.1 Image-Matrix und Bildmanipulationen

Zuerst wird beschrieben, was ein Pixelbild ist. Dann wird gezeigt, wie Sie in Gambas die Farben aller Bildpunkte eines Pixelbildes auslesen. Abschließend werden Ihnen Algorithmen vorgestellt, bei denen Image-Matrizen verändert werden, um bestimmte Bild-Effekte wie zum Beispiel einen Negativ-Effekt oder eine Spiegelung zu erzielen.

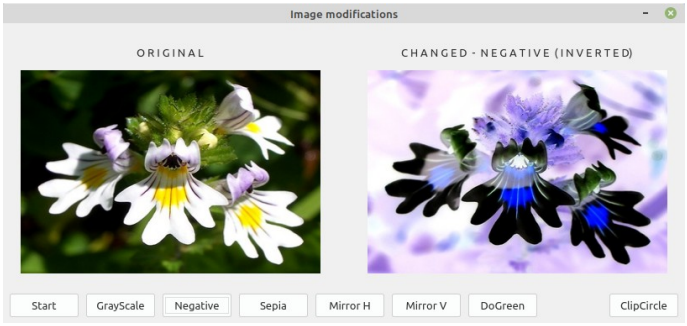


Abbildung 23.2.4.10.1.1: Negativ-Effekt

#### 23.2.4.10.1.1 Pixel-Bild

- Ein Pixel-Bild (Pixelgrafik, Rastergrafik) besteht aus einem xy-Raster von meist quadratischen Bildpunkten oder Pixeln.
- Die Position eines einzelnen Pixels im Bild wird seine xy-Koordinaten P(x,y) festgelegt.
- Ein Pixel repräsentiert die Bildpunkt-Farbe an der Position P(x,y).

#### 23.2.4.10.1.2 Farben aus einem Pixelbild auslesen oder setzen

Um Farbwerte eines Pixels in einem Image zu erhalten oder zu setzen, können Sie die Klasse Image als zweidimensionales Array auffassen: Image[x,y] würde sich auf das Pixel an der Position P(x, y) innerhalb des Bildes beziehen.

So können Sie die Farbe eines bestimmten Pixels an der Position P(x,y) auslesen:

```
Dim hImage As Image
Dim iColor As Integer
iColor = hImage [ X As Integer, Y As Integer ]
```

und so setzen Sie die Farbe eines bestimmten Pixels an der Position P(x,y):

```
Dim hImage As Image
Dim iColor As Integer
hImage [ X As Integer, Y As Integer ] = iColor
```

#### 23.2.4.10.1.3 Beispiel Image-Matrix

Mit dem folgenden Quelltext können Sie aus einem Pixelbild die Farben aller Bildpunkte P(x,y) auslesen und in einer Gambas-Konsole anzeigen:

```
[1]  Public Sub ShowImageMatrix(hImage As Image)
[2]
[3]      Dim iX, iY As Integer
[4]      Dim sColor As String
[5]
[6]  '-- Iteration line by line
[7]      For iY = 0 To hImage.H - 1
[8]      '-- Iteration column by column
[9]          For iX = 0 To hImage.W - 1
[10]             Select Case hImage[iX, iY]
[11]               Case 16711680
[12]                 sColor = "R"
[13]               Case 65280
[14]                 sColor = "G"
[15]               Case 255
[16]                 sColor = "B"
[17]               Case Else
[18]                 sColor = "Y"
```

```
[19]             End Select
[20]
[21]             Print "(" & Str(iY + 1) & "|" & Str(iX + 1) & "): " & sColor
[22]         '-- Print "(" & Str(iY + 1) & "|" & Str(iX + 1) & ") -> " & Str(hImage[iX, iY])
[23]         Next
[24]     Next
[25]
[26] End
[27]
[28] Public Sub btnShowImageMatrix_Click()
[29]
[30]     ShowImageMatrix(Image.Load("images/8x8.png"))
[31]
[32] End
```

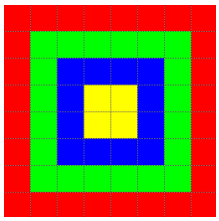

Abbildung 23.2.4.10.1.2: 8x8-Pixel-Bild

Image-Matrix mit den Koordinaten und Farben:

```
(1|1):R (1|2):R (1|3):R (1|4):R (1|5):R (1|6):R (1|7):R (1|8):R
(2|1):R (2|2):G (2|3):G (2|4):G (2|5):G (2|6):G (2|7):G (2|8):R
(3|1):R (3|2):G (3|3):B (3|4):B (3|5):B (3|6):B (3|7):G (3|8):R
(4|1):R (4|2):G (4|3):B (4|4):Y (4|5):Y (4|6):B (4|7):G (4|8):R
(5|1):R (5|2):G (5|3):B (5|4):Y (5|5):Y (5|6):B (5|7):G (5|8):R
(6|1):R (6|2):G (6|3):B (6|4):B (6|5):B (6|6):B (6|7):G (6|8):R
(7|1):R (7|2):G (7|3):G (7|4):G (7|5):G (7|6):G (7|7):G (7|8):R
(8|1):R (8|2):R (8|3):R (8|4):R (8|5):R (8|6):R (8|7):R (8|8):R
```

Sie können die Zeilen 9 bis 19 sowie die Zeile 21 – die so nur für das o.a. Bild 8x8.png gelten – für andere Bilder als Kommentar setzen und durch die Zeile 22 ersetzen.

### 23.2.4.10.1.4 Image-Matrix und Bildmanipulation

Im folgenden Abschnitt wird Ihnen anhand von zwei Beispielen gezeigt, wie Sie die Farben eines Original-Pixelbildes an den Positionen P(x,y) auslesen und die Image-Matrix eines neuen Bildes schreiben. Typisch dafür sind diese beiden Konzepte:

| | |
|---|---|
| Konzept_1 | Entweder werden die Farbwerte der Image-Matrix des Originals geändert und auf die gleichen Pixel P'(x,y) eines neuen Bildes abgebildet. |
| Konzept_2 | Die Farbwerte der Image-Matrix des Originals werden unverändert auf andere Bildpunkte eines neuen Bildes abgebildet. |

Die folgenden Beispiele sind zum Teil mit nativen Methoden der Image-Klasse schneller und einfacher realisierbar. Sie wurden trotzdem umgesetzt, um die Verwendbarkeit der Pixel-Matrix auf einfache Weise zu veranschaulichen.

Im Beispiel 1 wird das Konzept_1 umgesetzt, um einen Sepia-Effekt zu erzeugen.

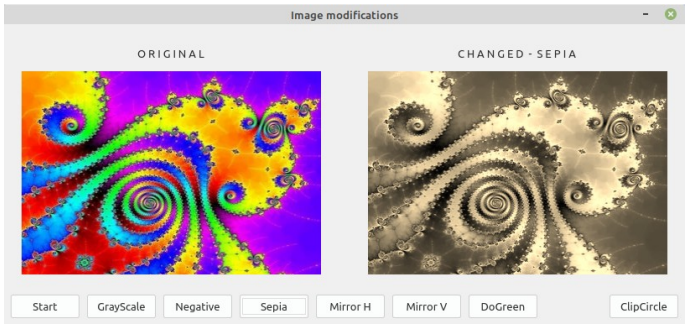


Abbildung 23.2.4.10.1.3: Sepia-Effekt

Folgender Algorithmus beschreibt die notwendige Bild-Manipulation:

(S1) Ermitteln Sie den RGB-Wert eines Original-Pixels P(x,y).
(S2) Berechnen Sie die neuen RGB-Werte für einen Sepia-Effekt mit den folgenden Wichtungen. Verwenden Sie den ganzzahligen Wert.
- newRed = Ganzzahl( 0,393 * oldRed + 0,769 * oldGreen + 0,189 * oldBlue)
- newGreen = Ganzzahl( 0,349 * oldRed + 0,686 * oldGreen + 0,168 * oldBlue)
- newBlue = Ganzzahl( 0,272 * oldRed + 0,534 * oldGreen + 0,131 * oldBlue)

(S3) Legen Sie den neuen RGB-Wert des Pixels P'(x,y) im neuen Bild gemäß der folgenden Bedingung fest:
- Wenn newRed > 255 dann newRed = 255
- Wenn newGreen > 255, dann newGreen = 255
- Wenn newBlue > 255, dann newBlue = 255

(S4) Ersetzen Sie die RGB-Werte des Original-Pixels P(x,y) durch die in den Schritten S2 und S3 berechneten Werte im neuen Bild in P'(x,y).
(S5) Wiederholen Sie die Schritte S1 bis S4 für jedes Pixel P(x,y) des Original-Bildes.

Quelltext:

```
Public Function Image2Sepia(hImage As Image) As Image
    Dim iX, iY As Integer
    Dim iOldRed, iOldGreen, iOldBlue, iNewRed, iNewGreen, iNewBlue As Integer
    Dim tmpImage As Image

    tmpImage = New Image(hImage.W, hImage.H, Color.White)

'-- Iteration line by line
    For iY = 0 To hImage.H - 1
    '-- Iteration column by column
        For iX = 0 To hImage.W - 1
            iOldRed = Color[hImage[iX, iY]].Red
            iOldGreen = Color[hImage[iX, iY]].Green
            iOldBlue = Color[hImage[iX, iY]].Blue
            iNewRed = Int(0.393 * iOldRed + 0.769 * iOldGreen + 0.189 * iOldBlue)
            If iNewRed > 255 Then iNewRed = 255
            iNewGreen = Int(0.349 * iOldRed + 0.686 * iOldGreen + 0.168 * iOldBlue)
            If iNewGreen > 255 Then iNewGreen = 255
            iNewBlue = Int(0.272 * iOldRed + 0.534 * iOldGreen + 0.131 * iOldBlue)
            If iNewBlue > 255 Then iNewBlue = 255
            tmpImage[iX, iY] = Color.RGB(iNewRed, iNewGreen, iNewBlue)
        Next
    Next
    Return tmpImage
End

Public Sub btnDoSepia_Click()
    lblChanged.Text = "C H A N G E D  -  S E P I A"
    imgChanged = Image2Sepia(imgOriginal)
    pboxChanged.Picture = imgChanged.Picture
End
```

Um zum Beispiel ein Bild horizontal zu spiegeln, müssen Sie das Konzept_2 mit einem speziellen Algorithmus kombinieren:

(S1) Setzen Sie im neuen Bild den RGB-Wert an der Stelle P'(x,y) durch den RGB-Wert des Originals an der Stelle P(Bildweite – x,y).
(S2) Wiederholen Sie den Schritt S1 für jedes Pixel P(x,y) des Original-Bildes.

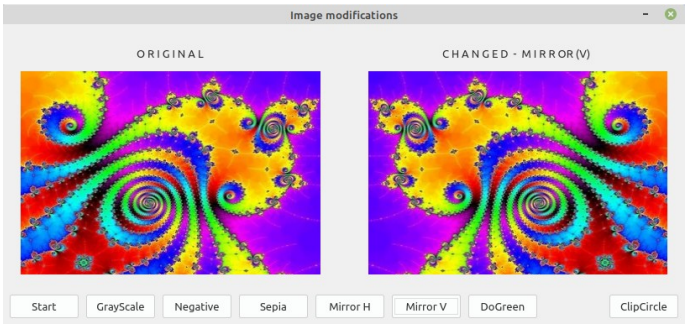


Abbildung 23.2.4.10.1.4: Horizontale Spiegelung

Dieser Quelltext wird verwendet:

```
Public Function ImageMirroring(hImage As Image, sMode As String) As Image

    Dim iX, iY As Integer
    Dim tmpImage As Image

'-- Spiegelungsrichtung entspricht einer horizontalen Linie

    tmpImage = New Image(hImage.W, hImage.H)

'-- Iteration line by line
    For iY = 0 To hImage.H - 1
    '-- Iteration column by column
        For iX = 0 To hImage.W - 1
            If Upper(sMode) = "H" Then
               tmpImage[iX, iY] = hImage[hImage.W - iX - 1, iY]
            Else
               tmpImage[iX, iY] = hImage[iX, hImage.H - iY - 1]
            Endif
        Next
    Next

    Return tmpImage

End

Public Sub btnDoMirrorV_Click()

    lblChanged.Text = "C H A N G E D  -  M I R R OR (V)"

    imgChanged = ImageMirroring(imgOriginal, "V")

    pboxChanged.Picture = imgChanged.Picture

End
```

Im Download-Bereich finden Sie ein Quelltext-Archiv für die Beispiele 1 bis 7 für eigene Erkundungen.

Beispiel 3

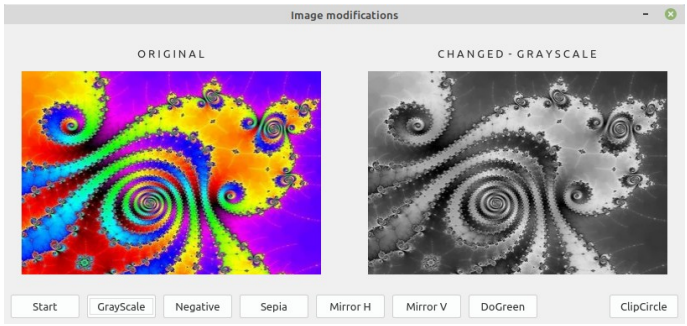


Abbildung 23.2.4.10.1.5: Graustufenbild

Beispiel 4

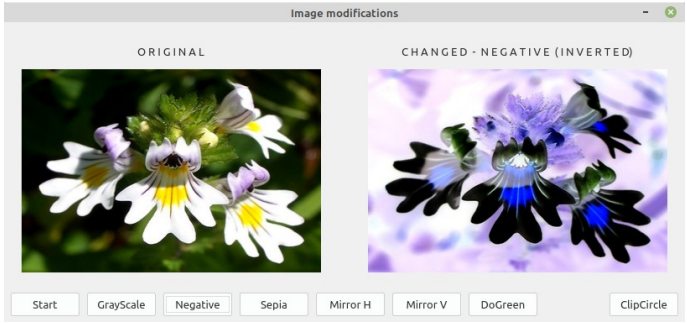


Abbildung 23.2.4.10.1.6: Negativ-Effekt

## Beispiel 5

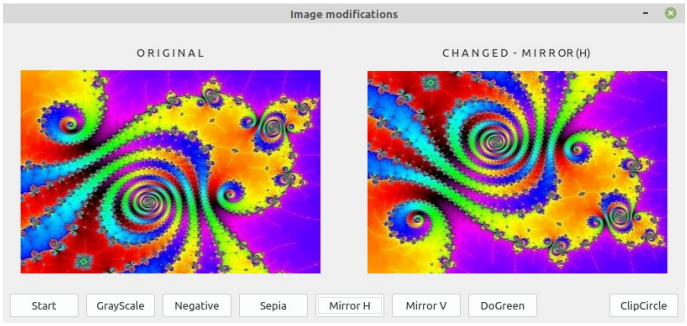


Abbildung 23.2.4.10.1.7: Vertikale Spiegelung

## Beispiel 6

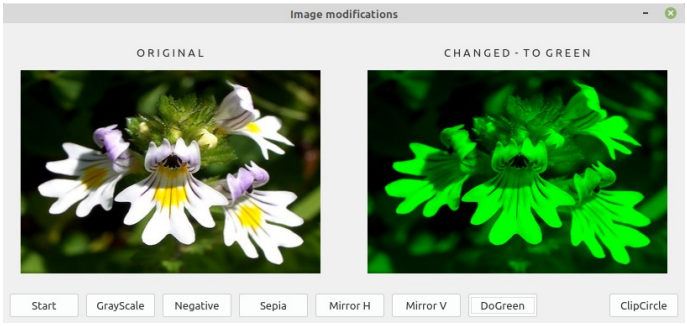


Abbildung 23.2.4.10.1.8: Mono-Color-Effekt

## Beispiel 7

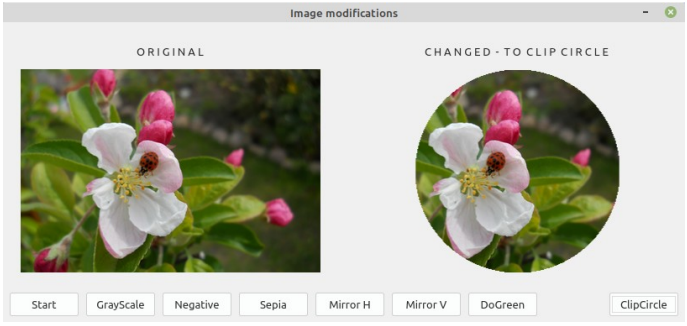


Abbildung 23.2.4.10.1.9: Bild mit kreisförmigem Ausschnitt

Das Beispiel 7 basiert auf einem Quelltext des Gambas-Buch-Autors Claus Dietrich.

## Quelltext:

```
Public Function Image2Circle(hImage As Image) As Image

    Dim iX, iY As Integer
    Dim tmpImage As Image
    Dim fAngle As Float
    Dim fRadius As Float
    Dim fD As Float
    Dim fYY As Float

    tmpImage = New Image(hImage.W, hImage.H)
    tmpImage = hImage

    fRadius = tmpImage.H / 2

    For iX = 0 To tmpImage.W / 2 - fRadius
        For iY = 0 To tmpImage.H - 1
      '-- Make sure that the alpha channel of the colour is used so that
      '-- the colour becomes transparent. So &HFFFFFFFF instead of &HFFFFFF!
          tmpImage[iX, iY] = &HFFFFFFFF
```

```
          tmpImage[tmpImage.W / 2 + fRadius + iX, iY] = &HFFFFFFFF
        Next
    Next
    For iX = (tmpImage.W - 1) / 2 To (tmpImage.W - 1) / 2 - fRadius Step -1
        fD = ((tmpImage.W - 1) / 2 - iX) / fRadius
        fAngle = ACos(fD)
        fYY = 1 - Sin(fAngle)
        For iY = 0 To fYY * fRadius
            tmpImage[iX, iY] = &HFFFFFFFF
            tmpImage[iX, tmpImage.H - iY] = &HFFFFFFFF
        Next
    Next
    For iX = (tmpImage.W - 1) / 2 To (tmpImage.W - 1) / 2 + fRadius
        fD = ((tmpImage.W - 1) / 2 - iX) / fRadius
        fAngle = ACos(fD)
        fYY = 1 - Sin(fAngle)
        For iY = 0 To fYY * fRadius
            tmpImage[iX, iY] = &HFFFFFFFF
            tmpImage[iX, tmpImage.H - iY] = &HFFFFFFFF
        Next
    Next

    Return tmpImage

End

Public Sub btnDoCircle_Click()

    lblChanged.Text = "C H A N G E D  -  T O  C L I P  C I R C L E"
    imgChanged = Image2Circle(imgOriginal.Copy())
    pboxChanged.Picture = imgChanged.Picture

End
```