24.8.4 Alternativen

Zum Einsatz der Komponente Map gibt es mindestens drei Alternativen:

1. Aufruf des Standard-Browsers aus einem Gambas-Programm heraus, dem Sie als URL die Seite von *openstreetmap.org* mit der gewünschten Karte (geografische Breite und Länge, Zoom) übergeben:

```
Public Sub btnShowMap_Click()
  Desktop.Open("http://www.openstreetmap.org/?lat=52.78979&amp;lon=11.75280&amp;zoom=15&amp")
End ' btnShowMap_Click()
```

2. Sie nutzen in einem eigenen Gambas-Formular die Komponente *WebView*, der Sie als URL die Seite von *openstreetmap.org* mit der gewünschten Karte (Breite, Länge, Zoom) übergeben:

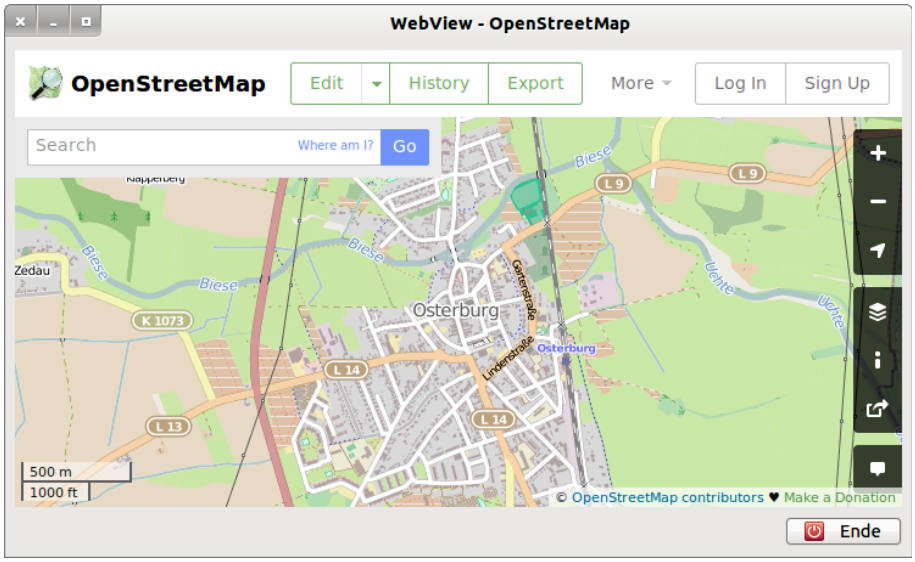

Abbildung 24.8.4.1: OSM-Karte (Hanse-Stadt Osterburg in Sachsen/Anhalt)

Ja – das ist der vollständige Quelltext:

```
' Gambas class file

Public Sub Form_Open()
  FMain.Center
  WebView1.Url = "http://www.openstreetmap.org/?lat=52.78979&amp;lon=11.75280&amp;zoom=14&amp"
End ' Form_Open()

Public Sub btnClose_Click()
  FMain.Close
End ' btnClose_Click()
```

3. Benötigen Sie nur Karten-Kacheln, um sie zum Beispiel zu einer Karte zusammenzusetzen und weiter zu verwenden, dann können Sie auf der Konsole mit dem Programm *wget* arbeiten. Vergessen Sie aber nicht, das importierte Bild im png-Format mit einem entsprechenden Copyright-Vermerk in einem Textfeld zu versehen!

Abbildung 24.8.4.2: Karten-Kachel als 256x256-Pixel-Grafik mit eingefügtem Copyright-Text

```
hans@linux:~$ wget http://85.30.190.241/13/4363/2676.png -P $HOME/OSM-Karten
```

```
--2013-05-01 09:40:41--  http://85.30.190.241/13/4363/2676.png
Verbindungsaufbau zu 85.30.190.241:80... verbunden.
HTTP-Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 26533 (26K) [image/png]
In »»/home/hans/OSM-Karten/2676.png«« speichern.

100%[======================================>] 26.533      --.-K/s   in 0,1s

2013-05-01 09:40:42 (267 KB/s) - »»/home/hans/OSM-Karten/2676.png«« gespeichert [26533/26533]

hans@linux:~$
```

Die Syntax für die URL im Kommando *wget*:

```
http://a.tile.openstreetmap.org/ZOOM/xTILE/yTILE.png
http://85.30.190.241/ZOOM/X-TILE/Y-TILE.png
```

erfordert notwendigerweise die Berechnung der Karten-Kachel-Koordinaten *xTile* und *yTile* aus den Werten der geografischen Breite, der geografischen Länge und die Angabe des gewünschten Zoom-Faktors. Der folgende Quelltext kann für die Berechnung genutzt werden:

```
Public Function LatLonZoom2xyTile(fLatitudeDeg As Float,fLongitudeDeg As Float,iZoom As Integer) As Inte-
ger[]
  Dim n As Integer
  Dim fXTile, fYTile, fLatitudeRad, fSecans As Float
  Dim aMatrix As New String[]

  fLatitudeRad = Rad(fLatitudeDeg)
  fSecans = 1 / Cos(fLatitudeRad)

  n = 2 ^ iZoom
  aMatrix.Add(Str(iZoom))
  fXTile = n * ((fLongitudeDeg + 180) / 360)
  aMatrix.Add(Int(fXTile))
  fYTile = n * (1 - (Log(Tan(fLatitudeRad) + fSecans) / Pi)) / 2 ' Log() in Gambas -> ln() mit Basis = e
  aMatrix.Add(Int(fYTile))

  Return aMatrix

End ' LatLonZoom2xyTile(...) As Integer[]

Public Sub btnBerechnungKachelKoordinaten_Click()
  Dim iXTile, iYTile, iZoom As Integer
  Dim fLatitude, fLongitude As Float
  Dim sTileServer As String
  Dim aMatrix As New String[]
  Dim aTileServer As String[] = ["a", "b", "c"]

  Randomize
  sTileServer = aTileServer[CInt(Rnd(0, 3))]
' Realität: Server-Auswahl im Content Delivery Network (CDN). Wird eine Anfrage an das CDN gesendet,
' dann wählt das Request-Routing-System einen geeigneten Replica-Server aus.

  iZoom = 13
  fLatitude = 52.78979   ' Breite  52,78979°
  fLongitude = 11.75280  ' Länge   11,75280°

  aMatrix = LatLonZoom2xyTile(fLatitude, fLongitude, iZoom)

  Print "Ort = Osterburg"
  Print "Geografische Breite = " & fLatitude & "°"
  Print "Geografische Länge = " & fLongitude & "°"
  Print
  Print "Kachel-Server = " & sTileServer
  Print "Zoom = " & aMatrix[0]
  Print "xTile = " & aMatrix[1]
  Print "yTile = " & aMatrix[2]
  Print "-----------------------------------------------------------------'"
  Print "Zoom = " & izoom
  Print "xTile = " & Geo.MapPointToTile(MapPoint(fLatitude, fLongitude), iZoom).X ' 4363
  Print "yTile = " & Geo.MapPointToTile(MapPoint(fLatitude, fLongitude), iZoom).Y ' 2676

End ' btnBerechnungKachelKoordinaten_Click()
```

Wesentlich flotter geht es direkt mit der Methode *Geo.MapPointToTile(..)* der Klasse Map.Geo in den letzten zwei Zeilen, welche intern die Berechnungen in der Funktion *LatLonZoom2xyTile(..)* einsetzt.

Hier folgt ein Skript, um einen Copyright-Text in ein Kachel-Bild einzufügen:

```
#!/bin/bash
#
# Quelle1: http://wiki.ubuntuusers.de/ImageMagick
# Quelle2: http://pecita.eu/police-en.php
# Wasserzeichentext in alle Bilder aus diesen Verzeichnis einfuegen
# Der Wasserzeichentext wird unten links ins Bild eingesetzt
# Sie koennen folgende Parameter anpassen:
Textabstandvonlinks=10
Textabstandvonunten=20
Schriftgroesse=30
Schriftart="Pecita.otf"
Schriftfarbe="red"
# Moegliche Farben koennen aufgelistet werden mit dem Befehl: convert -list color
Wasserzeichentext="© OpenStreetMap"
# Programmbeginn
echo "Textabstand von links: $Textabstandvonlinks"
echo "Textabstand von unten: $Textabstandvonunten"
echo "Schriftgoesse: $Schriftgroesse"
echo "Schriftart: $Schriftart"
echo "Schriftfarbe: $Schriftfarbe"
echo "Wasserzeichentext: $Wasserzeichentext"
echo " "
ls -1 *png | while read file;
do {
horizontal=`identify -verbose $file | grep Geometry: | awk {'print $2'} |cut -d"x" -f 1`
vertikal=`identify -verbose $file | grep Geometry: | awk {'print $2'} |cut -d"x" -f 2`
X=$Textabstandvonlinks
Y=$(($vertikal - $Textabstandvonunten))
convert -font $Schriftart -pointsize $Schriftgroesse -fill $Schriftfarbe -draw "text $X, \
        $Y '$Wasserzeichentext'" "$file" "`basename c_"$file"`";
echo "Bearbeite Datei $file"
}
done
echo "Wasserzeichen erfolgreich eingearbeitet."
echo
echo "Konsole schließen mit Enter!"
read dummy;
exit
# Programmende
```