Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

Mathematik der Kartenentwurfslehre
GIS - Software KARTO

Diplomarbeit in Technischer Mathematik

Matthias Wielach

Matrikelnummer: 9827088

Jänner 2007

Inhaltsverzeichnis

1 Einführung **4**
1.1 Geschichte 4

2 Grundlagen **6**
2.1 Allgemeines 6
2.1.1 Die Erde ist keine Kugel! 6
2.1.2 Probleme der Kartenentwurfslehre 6
2.2 Mathematische Grundlagen 7
2.2.1 Parameterdarstellung von Kurven 7
2.2.2 Parameterdarstellung von Flächen 7
2.2.3 Die Einheitssphäre 8
2.2.4 Polarkoordinaten in der Ebene 8

3 Verschiedene Entwürfe **9**
3.1 Kegelentwürfe 9
3.1.1 Allgemeiner Ansatz 9
3.1.2 Abstandstreuer Kegelentwurf 10
3.1.3 Allgemeiner Azimutalentwurf 11
3.1.4 Abstandstreuer Azimutalentwurf 11
3.1.5 Allgemeiner Zylinderentwurf 12
3.1.6 Abstandstreuer Zylinderentwurf 12
3.2 Hauptverzerrungen 13
3.2.1 Erste Grundform für geografische Koordinaten 13
3.2.2 Längenverzerrung 14
3.2.3 Längentreue und Winkeltreue 14
3.2.4 Hauptverzerrungsrichtungen 14
3.2.5 TISSOTsche Indikatrix 15
3.2.6 Winkelverzerrung 16
3.2.7 Geodätische Abbildungen 17
3.2.8 Theorema egregium 17
3.3 Konforme Entwürfe 18
3.3.1 Konformer Kegelentwurf 18
3.3.2 Konformer Azimutalentwurf 18
3.3.3 Konformer Zylinderentwurf 19

4 Software zur Erstellung von Kartenentwürfen **20**
4.1 Geografische Informationssysteme - GIS 20
4.1.1 Geodaten-Server-Systeme 21
4.1.2 Online GIS Systeme 21

4.2 Datenstrukturen 23
4.2.1 XML 23
4.2.2 KML - Daten aus GOOGLE earth 24
4.2.3 Dateninterface 26
4.3 Software KARTO in BORLAND Delphi 27
4.3.1 Idee 27
4.3.2 Borland Developer Studio 2006 - Warum Delphi? 27
4.3.3 Screen-Shots der Entwicklungsumgebung BDS 2006 28
4.4 KARTO - ein kurzes Handbuch 29
4.4.1 Hauptprogramm 29
4.4.2 Tool: Konvertiere XML > KartoTXT 31
4.4.3 Tool: Kombiniere 2 KartoTXT-Dateien 32
4.4.4 Tool: Gradnetz erstellen 33
4.4.5 Beispiele von mit KARTO erstellten Kartenentwürfen 34
4.5 Quellcode von KARTO 38
4.5.1 Hauptformular 38
4.5.2 Formular Konvertiere XML > KartoTXT 66
4.5.3 Formular Erstelle Gradnetz 70
4.5.4 Formular Kombiniere 2 KartoTXT-Dateien 74

5 Anhang **77**
5.1 Special Thanks 77
5.2 Persönliche Anmerkungen 77
5.3 Literaturverzeichnis 78

1 Einführung

1.1 Geschichte

Die ältesten Karten stammen bereits aus dem Neolithikum.
Wandmalereien zeigen zentralanatolische Siedlungen um 6200 v. Chr. mit Häusern und dem Doppelgipfel des Vulkans Hasan Dagi (3.268 m - Türkei).
Auch aus der babylonischen Zeit sind geritzte Tontafeln erhalten.
Sie sind weitere bedeutende Zeugnisse früher Kartenentwürfe.

Die ältesten Schriftstücke menschlicher Kultur dokumentieren uns von der Erde als eine Scheibe. Bereits in Mesopotamien kannte man dieses Modell der flachen Scheibe, die auf dem Ozean schwimmt, und auch die frühen griechischen Philosophen Anaximander und Hekataios (ca 600 v. Chr.) folgten dieser Theorie.
In Griechenland wurden auch schon im Altertum die ersten Ideen einer Kugelgestalt der Erde geboren. Pythagoras vertrat im 6. Jh. v. Chr. diese Ansicht vor allem aus ästhetischen Gründen.
Aristoteles nannte dann im 4. Jh. v. Chr. folgende Gründe:

- Sämtliche schweren Körper streben zum Mittelpunkt des Alls.
  Da sie dies von allen Seiten her gleichmäßig tun und die Erde im Mittelpunkt des Alls steht, muss sie eine kugelrunde Gestalt annehmen.
- Bei von der Küste wegfahrenden Schiffen wird der Rumpf vor den Segeln vom Meer verdeckt.
- In südlichen Ländern erscheinen südliche Sternbilder höher über dem Horizont.
- Der Erdschatten bei einer Mondfinsternis ist rund.

Heute können wir sagen, dass der erste Punkt falsch, die anderen 3 richtig sind.

Für die weiteren Epochen war das Weltbild des Ptolemäus (ca. 100 n. Chr.) prägend. In den ältesten Manuskripten seiner Kosmografie finden sich auch Handzeichnungen von Karten, im allgemeinen war sein Werk jedoch ein Verzeichnis astronomischer Positionen mit den Attributen Breite und Länge.
Die Werke von Ptolemäus, obwohl noch stark fehlerhaft, erfuhren nach über 1000 Jahren durch den einsetzenden Buchdruck um 1450 eine erhebliche Verbreitung.
Erst die verstärkte weltweite Seefahrt um 1500 und die Werke Mercators brachten die Wende hin zu mehr Realitätsnähe bei der Kartographie.

Es wurde noch von vielen Gelehrten behauptet, dass die Erde flach sei.
Die Kugelgestalt war laut derer Ansicht mit den Gesetzen der Logik
(Die Menschen fallen doch auf der Unterseite runter.) nicht vereinbar.
Manchmal wurden auch Widersprüche zur Bibel genannt.

Seit dem Mittelalter gilt die Tatsache der Kugelgestalt der Erde als Fakt,
und Aristoteles (384-322 v. Chr.), der auch Belege zur Kugelgestalt erbrachte,
galt als eine der höchsten Autoritäten was Fragen der Naturwissenschaften anbelang.
Auch im Heiligen Römischen Reich symbolisierte der Reichsapfel, eine der
Reichsinsignien, die Weltkugel.

Ferner ist aus römischer Zeit die **Tabula Peutingeriana** erhalten.
Sie ist eine von West nach Ost unnatürlich verzerrte Straßenkarte des römischen
Reiches mit Angabe der Militärstationen und Entfernungsangaben in Meilen.

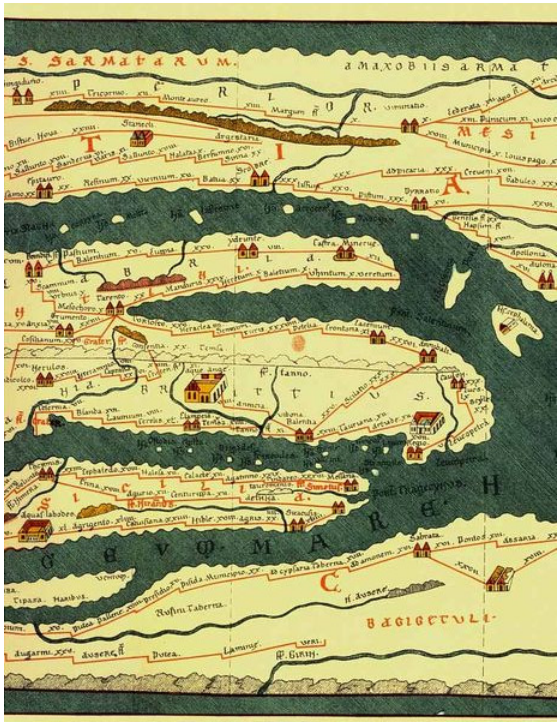


Abbildung 1: Tabula Peutingeriana

Ab dem 16. und 17. Jahrhundert entwickelte sich dann aus den Ideen des
Ptolemäus die moderne Kartographie, wo geometrische Projektionen die Grundlagen
zur Erstellung von Karten sind.

2 Grundlagen

2.1 Allgemeines

2.1.1 Die Erde ist keine Kugel!

Es ist allgemein bekannt, dass die Erde nicht genau der mathematischen Gestalt einer Kugel entspricht. Die Abplattung der Pole ist vor allem durch die Zentrifugalkraft der Erdrotation begründet. Und so stellt sich die Frage, wie man die genaue Form der Erde mathematisch idealisieren kann. Hierzu gibt es mehrere Ansätze:

Die Erde könnte man als Drehfläche interpretieren. Durch die ständige Änderung der Erdrotationsachse sprechen wir von der sogenannten Polhöhenschwankung, was eine ständige kleine Änderung der geografischen Länge und Breite bedeutet.

Weiters könnte man die Erde als Geoid, also als allgemeine Fläche 14. Ordnung, oder als Sphäroid, also als abgeplattetes Drehellipsoid, betrachten.
(Die Abplattung beträgt ca. 1:298.)

Bei allen weiteren Untersuchungen gehen wir von der Erde als Kugel aus. Diese Vereinfachung ist zur Erstellung von geografischen Karten völlig ausreichend.

2.1.2 Probleme der Kartenentwurfslehre

Wenn wir von einem Kartenentwurf sprechen, so meinen wir eine Abbildung eines Teiles der Erde in die Ebene.
Um die Eigenschaften und Besonderheiten verschiedener Entwürfe zu beschreiben, sind folgende Definition notwendig:
Ein Kartenentwurf wird als **flächentreu** bezeichnet, wenn sich die Größe einer Fläche um einen bestimmten Maßstab verkleinert wiedergibt und dieser Maßstab über die gesamte Karte der gleiche ist.
Eine solche Karte weist starke Verzerrungen, vor allem am Kartenrand, auf.
Ein Kartenentwurf wird als **winkeltreu** bezeichnet, wenn die Winkel in der Karte gleich der Winkel im Urbild (auf der Kugel) sind.

Ein Kartenentwurf kann nie längentreu, also flächentreu und winkeltreu zugleich, sein!

Grundsätzlich muss man also die Frage stellen, welche Forderungen an den Kartenentwurf gestellt werden sollen. Es ergibt sich somit, dass verschiedene Entwürfe in verschiedenen Gebieten zum Einsatz kommen.

2.2 Mathematische Grundlagen

2.2.1 Parameterdarstellung von Kurven

$$c : I \to \mathbb{R}^n$$

c sei k-mal stetig differenzierbar (k hinreichend groß)
und das Parameterintervall $I \neq \emptyset$.

Ausgeschrieben: $c(t) = (c_1(t), c_2(t), \ldots, c_n(t))$ (mit $t \in I$)

Wir stellen an die Funktion c folgende Forderungen:

1. c sei injektiv
2. c sei regulär (D.h. $\dot{c} \neq 0 \quad \forall t \in I$.)
3. $\forall I_0 \subset I, I_0$ offen in $\mathbb{R} : \exists M_0 \subset \mathbb{R}^n, M_0$ offen mit $c(I_0) = c(I) \cap M_0$

Erfüllt c alle Forderungen so sprechen wir von einem C^k**-Kurvenstück**.

Im Folgenden beschränken wir uns auf C^k-Parameterdarstellungen von C^k-Kurvenstücken.

Anmerkung: $c_{t_0} : t \to c(t_0) + (t - t_0)\dot{c}(t)$ ist die Tangente an $c(I)$ im Punkt $c(t_0)$.

2.2.2 Parameterdarstellung von Flächen

$$f : U \to \mathbb{R}^n$$

f sei r-mal partiell differenzierbar ($r \leq 1$)
(Partielle Ableitungen sind stetig!)
und der Parameterbereich $U \neq \emptyset$.

Ausgeschrieben: $f(u) = f(u_1, u_2) = (f_1(u_1, u_2), f_2(u_1, u_2), \ldots, f_n(u_1, u_2))$

Wir stelen an die Funktion f folgende Forderungen:

1. f sei injektiv
2. f sei regulär in U (D.h. $\frac{\partial f}{\partial u_1}(a)$ und $\frac{\partial f}{\partial u_2}(a)$ sind l.u. $\forall a \in U$.)
3. $\forall I_0 \subset I, I_0$ offen in $\mathbb{R} : \exists M_0 \subset \mathbb{R}^n, M_0$ offen mit $c(I_0) = c(I) \cap M_0$

Erfüllt f alle Forderungen so sprechen wir von einem C^r**-Flächenstück**.

Anmerkung:
$f_a : u \to f(a) + (u_1 - a_1)\frac{\partial f}{\partial u_1}(a) + (u_2 - a_2)\frac{\partial f}{\partial u_2}(a)$ ist die Tangentialebene an $c(U)$ im Punkt $f(a)$.

2.2.3 Die Einheitssphäre

Am einfachen Beispiel der Parametrisierung der Einheitssphäre kann man sehr deutlich wichtige Details der Kartenentwurfslehre erkennen.

Der Parameterbereich oder das **Parameterrechteck** ist hier:

$$U := (-\pi, \pi) \times (-\frac{\pi}{2}, \frac{\pi}{2})$$

und die Parametrisierung der Einheitssphäre sieht wie folgt aus:

$$x_1(u_1, u_2) = f_1(u_1, u_2) = cos(u_1)cos(u_2)$$

$$x_2(u_1, u_2) = f_2(u_1, u_2) = sin(u_1)cos(u_2)$$

$$x_3(u_1, u_2) = f_3(u_1, u_2) = sin(u_1)$$

$$\text{mit } u = (u_1, u_2) \in U$$

Was zur bekannten Formel

$$x_1^2 + x_2^2 + x_3^2 = 1$$

führt.

u_1 repräsentiert den Parameter für die **geografische Länge**,
u_2 repräsentiert den Parameter für die **geografische Breite**.

$\sum$ ist die längs eines Halbmeridians aufgeschnittene Einheitsspähre: $\sum := f(U)$,
somit ist $\sum$ ein Flächenstück.

Anmerkung:
Die gesamte Kugeloberfläche kann nicht als Flächenstück dargestellt werden!

2.2.4 Polarkoordinaten in der Ebene

Der Parameterbereich der Polarkoordinaten ist $\bar{U} = (-\pi, \pi) \times \mathbb{R}^+$
wobei $\bar{u}_1$ der Polarwinkel und $\bar{u}_2$ der Abstand zum Ursprung ist.
Die kartesischen Koordinaten $\bar{x_1}$ und $\bar{x_2}$ sind dann wie folgt gegeben:

$$\bar{x}_1(\bar{u}_1, \bar{u}_2) = \bar{f}_1(\bar{u}_1, \bar{u}_2) = \bar{u}_2 cos\bar{u}_1$$

$$\bar{x}_2(\bar{u}_1, \bar{u}_2) = \bar{f}_2(\bar{u}_1, \bar{u}_2) = \bar{u}_2 sin\bar{u}_1$$

Wird die negative $\bar{x}_1$-Achse und der 0-Punkt herausgeschnitten, so ist $\bar{f}(\bar{U})$ ein Flächenstück.

3 Verschiedene Entwürfe

3.1 Kegelentwürfe

3.1.1 Allgemeiner Ansatz

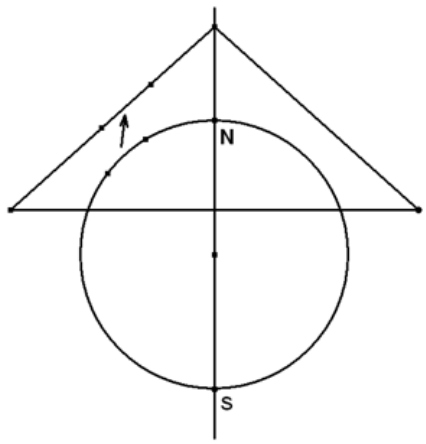


Abbildung 2: Allgemeiner Kegelentwurf

Wir stellen nun an die Abbildung $\sum$ (Erde) $\rightarrow \Delta$ (Drehkegel)
folgende Forderungen:

- Die Breitenkreise auf der Erde werden auf Breitenkreise auf dem Drehkegel abgebildet.
- Der Urpunkt und der Bildpunkt liegen in einer gemeinsamen Ebene mit der Erdachse.
- Es handelt sich um eine reguläre C^1 - Abbildung.

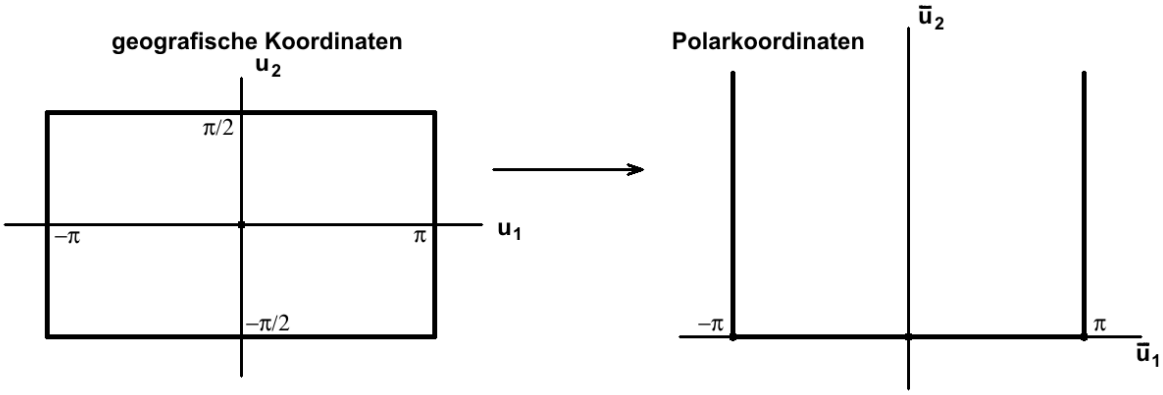
Die Abbildung der geografischen Koordinaten (u_1, u_2) auf die Polarkoordinaten $(\bar{u}_1, \bar{u}_2)$ kann nun mittels folgender Formeln durchgeführt werden:

$$\bar{u}_1 = n * u_1$$

$$\bar{u}_2 = p(u_2)$$

n ist der Stauchungsfaktor der Abbildung,
und für p wird eine geeignete Funktion gewählt.

Abbildung zur Transformation geografische K. → Polarkoordinaten:



3.1.2 Abstandstreuer Kegelentwurf

Bei diesem Entwurf berührt der Drehkegel die Kugel (Erde),
was in folgender Abbildung verdeutlicht wird:

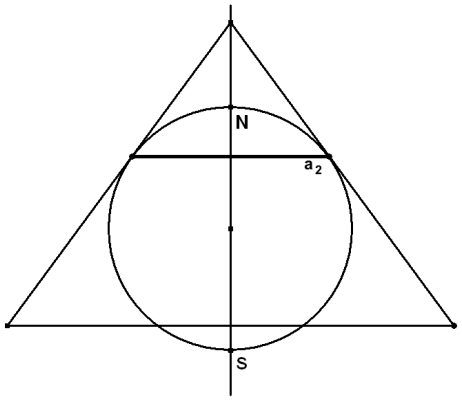


Abbildung 3: Abstandstreuer Kegelentwurf Kegelentwurf

a_2 ist der Berührparallelkreis und erscheint natürlich auf der Kugel
und im Entwurf (Karte) gleich lang.
Mittels der trigonometrischen Formeln kann man leicht den Stauchungsfaktor n
und die Funktion p berechnen.

$$n = sin(a_2) \qquad p = cot(a_2) + a_2 - u2$$

3.1.3 Allgemeiner Azimutalentwurf

Der Ansatz zum Azimutalentwurf ist durch folgende Skizze leicht erklärt:

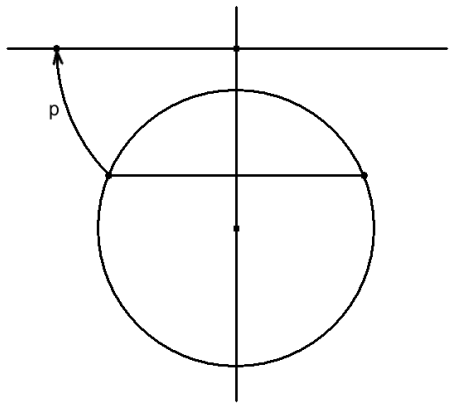


Abbildung 4: Allgemeiner Azimutalentwurf

Der allgemeine Azimutalentwurf ist ein Sonderfall des Kegelentwurfs mit $n = 1$. Wir erhalten hier folgende Formeln:

$$\bar{u}_1(u_1) = u_1$$

$$\bar{u}_2(u_2) = p(u_2)$$

Man erkennt dass $\lim\limits_{u_2 \to \pi/2} \bar{u}_2 = 0$, und daher muss die Funktion im Nordpol stetig ergänzbar sein. Weiters stellen wir die gleichen Forderungen wie im Punkt 3.1.1.

3.1.4 Abstandstreuer Azimutalentwurf

Dieser Entwurf ensteht indem man beim abstandstreuen Kegelentwurf den Berührparallelkreis a_2 gegen $\pi/2$ gehen lässt. Die Bildebene des Azimutalentwurfs berührt somit die Kugel und wir erhalten folgendes für p:

$$p(u_2) = \pi/2 - u_2$$

3.1.5 Allgemeiner Zylinderentwurf

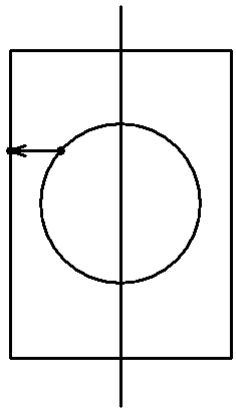


Abbildung 5: Allgemeiner Zylinderentwurf

Wiederum gelten die gleichen Forderungen wie im Punkt 3.1.1.

Beim Zylinderentwurf erhalten wir keine Polarkoordinaten, daher verwenden wir für die Formeln $\bar{x}_1$ und $\bar{x}_2$.

$$\bar{x}_1 = n * u_1$$

$$\bar{x}_2 = g(u_2)$$

O.b.d.A. nehmen wir $g(0) = 0$ und $g(u_2) > 0$ für $u_2 > 0$.
Wir erhalten somit das Bild des Äquators direkt auf der $\bar{x}_1$-Achse, und weiters wird die Nordhalbkugel über und analog die Südhalbkugel unterhalb der $\bar{x}_1$-Achse abgebildet.

3.1.6 Abstandstreuer Zylinderentwurf

Dieser Entwurf ensteht aus dem abstandstreuen Kegelentwurf, indem man a_2 gegen 0 gehen lässt. Der Zylinder berührt also die Kugel (Erde) im Äquator. Dieser Entwurf wird auch als quadratische Plattkarte bezeichnet und hat folgende einfache Formeln:

$$\bar{x}_1 = u_1$$

$$\bar{x}_2 = u_2$$

3.2 Hauptverzerrungen

Wenn wir von Kartenentwürfen sprechen, geht es im allgemeinen um Abbildungen von der Kugeloberfläche in die Ebene. Natürliche treten dabei Verzerrungen auf, die nun genauer untersucht werden.

3.2.1 Erste Grundform für geografische Koordinaten

Die Tangente einer beliebigen Raumkurve $c(u_1, u_2)$ durch einen Flächenpunkt kann als Linearkombination der partiellen Ableitungen nach den beiden Parametern der Parameterdarstellung der Raumkurve (der Fläche) dargestellt werden.
Wir erhalten so den Richtungsvektor der Tangente:

$$\dot{c} = \frac{\partial c}{\partial t} = \frac{\partial c}{\partial u_1}\dot{u}_1 + \frac{\partial c}{\partial u_2}\dot{u}_2 = c_{u_1}\dot{u}_1 + c_{u_2}\dot{u}_2$$

In der Formel der Bogenlänge s

$$s = \int_{t_1}^{t_2} |\dot{c}|\partial t$$

berechnen wir zunächst den Betrag für $|\dot{c}|$.

$$\dot{c}^2 = (c_{u_1}\dot{u}_1 + c_{u_2}\dot{u}_2)^2 = (c_{u_1} * c_{u_1})\dot{u}_1^2 + 2c_{u_1}c_{u_2}\dot{u}_1\dot{u}_2 + (c_{u_2} * c_{u_2})\dot{u}_2^2$$

Die hier auftretenden Koeffizienten

$$E = c_{u_1} * c_{u_1}$$

$$F = c_{u_1} * c_{u_2}$$

$$G = c_{u_2} * c_{u_2}$$

werden **erste Fundamentalgrößen** genannt.

Nach der Definiton des Skalarproduktes kann F auch durch den eingeschlossenen Winkel der beiden Vektoren c_{u_1} und c_{u_2} geschrieben werden:

$$F = c_{u_1} * c_{u_2} = |c_{u_1}||c_{u_2}| \cos\varphi$$

Sind also diese Parameterlinien zueinander orthogonal, wie es beim Parameternetz auf der Kugel (Längen- und Breitengrade) eintritt, so verschwindet die Fundamentalgröße F, und durch Einsetzen in die Formeln der Einheitssphäre erhalten wir die **Erste Grundform für geografische Koordinaten**:

$$E = \cos^2 u_2$$

$$F = 0$$

$$G = 1$$

3.2.2 Längenverzerrung

Das Verhältnis λ der beiden Bogenelemente ∂s (im Urbild) und $\partial \overline{s}$ (im Bild) definiert die Verzerrung der Längen im Punkt (u_1, u_2).

λ wird somit als **Längenverzerrung** bezeichnet.

Die Quadrate der Längenverzerrung entlang der Parameterlinien lauten somit:

$$\lambda_{u_1}^2 = \frac{\overline{E}}{E}$$

und

$$\lambda_{u_2}^2 = \frac{\overline{G}}{G}$$

3.2.3 Längentreue und Winkeltreue

Bei $\lambda = 1$ spricht man von einer **längentreuen Abbildung**.
So eine Abbildung liefert also keine Längenverzerrungen und wird auch als Verbiegung bezeichnet. Die Fundamentalgrößen sind hier jeweils ident.
$(E = \overline{E},\ F = \overline{F},\ G = \overline{G})$

Ein Sonderfall liegt vor, wenn die Längenverzerrung λ nur von der Lage des abzubildenden Punktes nicht aber von der jeweiligen Kurvenrichtung abhängig ist. Man spricht dann von einer **konformen** oder **winkeltreuen Abbildung**.

In diesem Fall muss folgendes gelten:

$$E = \mu^2(u_1, u_2)\overline{E}, \qquad F = \mu^2(u_1, u_2)\overline{F}, \qquad F = \mu^2(u_1, u_2)\overline{F}$$

Denn dann wird ein Winkel im Urbild auf einen Winkel der gleichen Größe im Bild abgebildet. $(\cos\varphi = \cos\overline{\varphi})$

3.2.4 Hauptverzerrungsrichtungen

Bei der Abbildung einer Fläche Φ auf eine Fläche $\overline{\Phi}$ gibt es in jedem Flächenpunkt zwei **Hauptverzerrungsrichtungen** . Diese Richtungen stehen senkrecht aufeinander und bestimmen das Netz der Hauptverzerrungslinien.
Werden diese Hauptverzerrungslinien als Parameterlinien gewählt, so verschwindet wegen der Orthogonalität die Fundamentalgröße F von Φ. Bei der Abbildung bleiben die Hauptverzerrungsrichtungen erhalten, und daher gilt durch die Wahl der Hauptverzerrungsrichtungen als Parameterlinien neben $F = 0$ auch $\overline{F} = 0$.

Die beiden Hauptverzerrungen werden λ_1 und λ_2 genannt.

3.2.5 TISSOTsche Indikatrix

Mit der Gleichung der Tissotschen Indikatrix

$$\lambda_1 x^2 + \lambda_2 y^2 = 1$$

kann man durch die Vorraussetzung, dass die Parameterlinien Hauptverzerrungslinien sind, die Hauptverzerrung im Flächenpunkt P berechnen:

$$\lambda_1^2 = \frac{\overline{E}}{E}$$

$$\lambda_2^2 = \frac{\overline{G}}{G}$$

Wählt man in der Tangentialebene des Punktes P von Φ einen Kreis vom Radius 1, so stellt die Gleichung der Indikatrix das Bild dieses Kreises (in $\overline{\Phi}$) dar.

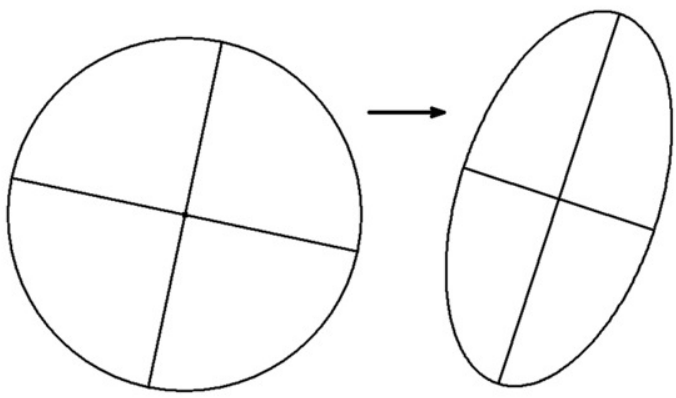


Abbildung 6: Tissotsche Indikatrix

Das Bild diese Kreises ist eine Ellipse mit den Hauptachsen λ_1 und λ_2.

Die Formel von Tissot

$$\frac{\partial s^2}{\partial \overline{s}^2} = \frac{1}{\lambda^2} = \frac{1}{\lambda_1^2}\left(\frac{\overline{E}\partial u_1^2}{\overline{E}\partial u_1^2 + \overline{G}\partial u_2^2}\right) + \left(\frac{\overline{G}\partial u_2^2}{\overline{E}\partial u_1^2 + \overline{G}\partial u_2^2}\right) = \frac{\cos^2\overline{\varphi}}{\lambda_1^2} + \frac{\sin^2\overline{\varphi}}{\lambda_2^2}$$

definiert also in $\overline{\Phi}$ eine Ellipse mit diesen Hauptachsen.

Bei längentreuen Abbildungen sind die Tissotschen Inkatrizen kongruente Kreise vom Radius 1. Bei konformen Abbildungen sind die Indikatrizen nicht kongruente Kreise, hat der Kreis in Φ den Radius 1, so hat die Indikatrix dann den Radius λ.

3.2.6 Winkelverzerrung

Mit Hilfe der Tissotschen Indikatrix lässt sich auch die Winkelverzerrung einer Abbildung berechnen. Aus der Tissotschen Formel folgt

$$\tan\overline{\varphi} = \sqrt{\frac{\overline{G}}{\overline{E}}}\frac{\partial u_2}{\partial u_1}$$

und analog für die Urbildfläche Φ

$$\tan\varphi = \sqrt{\frac{G}{E}}\frac{\partial u_2}{\partial u_1}$$

Wird weiters der Differenzialquotient eliminiert, so erhält man

$$\tan\varphi = \sqrt{\frac{G}{\overline{G}}\frac{\overline{E}}{E}}\tan\overline{\varphi} = \frac{\lambda_1}{\lambda_2}\tan\overline{\varphi}$$

Die Veränderung, die der Winkel φ erfährt, wenn er durch die Abbildung in den Winkel $\overline{\varphi}$ transformiert wird, ist durch die Differenz $\varphi - \overline{\varphi}$ charakterisiert.
Der Maximalwert ω dieser Winkeldifferenz führt auf die Winkelverzerrung.

Nach den Additionstheoremen der trigonometrischen Funktionen gilt

$$\frac{\sin(\varphi - \overline{\varphi})}{\sin(\varphi + \overline{\varphi})} = \frac{1 - \tan\overline{\varphi}\cot\varphi}{1 + \tan\overline{\varphi}\cot\varphi}$$

Somit folgt

$$\sin(\varphi - \overline{\varphi}) = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\sin(\varphi + \overline{\varphi})$$

Da die Hauptverzerrungen λ_1 und λ_2 in einem Flächenpunkt feste Werte besitzen, wird die Winkeldifferenz $\varphi - \overline{\varphi}$ maximal, wenn $\sin(\varphi + \overline{\varphi})$ seinen größten Wert 1 annimmt.
Wir erhalten somit für die maximale Winkelverzerrung:

$$\sin\omega = \pm\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}$$

und definieren unter Berücksichtigung der Vorzeichen die Winkelverzerrung W.

$$W = 2\omega = 2\arcsin\left|\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right|$$

Anmerkung: Bei winkeltreuen Abbildung ist $\lambda_1 = \lambda_2$ und damit natürlich $W = 0$.

3.2.7 Geodätische Abbildungen

Als echt geodätische Abbildungen werden jene bezeichnet, die geodätische Linien der Fläche Φ wieder in geodätische Linien der Fläche $\overline{\Phi}$ transformieren.
Geodätische Linien sind Kurven mit verschwindender geodätischer Krümmung, was geometrisch interpretiert die kürzesten Verbindungskurven zweier Flächenpunkte eines Flächenstücks sind.

Sie erfüllen folgende Gleichung:

$$\kappa_g = s * x'' = 0$$

Bei einer echt geodätischen Abbildung erfüllt natürlich auch die Bildkurve in $\overline{\Phi}$ diese Gleichung.

3.2.8 Theorema egregium

Nach dem Theorema egregium von C.F.GAUSS (1777-1855) bleibt bei längentreuen Abbildungen das Gaußsche Krümmungsmaß erhalten, d.h. werden die Flächen Φ und $\overline{\Phi}$ längentreu aufeinander abgebildet, so haben sie die gleiche Gaußsche Krümmung.
Da aber die Gaußsche Krümmung der Ebene verschwindet, nicht aber bei der Kugel bzw. bei einem Drehellipsoid, stellen wir fest, dass es unmöglich ist,
Karten zu konstruieren, die ein exates Abbild der Erdoberfläche darstellen.
Nur einzelne Kurven können längentreu abgebildet werden.

Alle Kartenentwürfe haben also mehr oder wenig stark ausgeprägte Verzerrungen.
Es ist also bei Karten in großen Maßstäben oft notwendig, Abbildungsfunktion zu verwenden, die nur in gewissen (kleinen) Gebieten der Erdoberfläche gültig bzw. sinnvoll sind.

Es gibt beliebig viele Abbildungsfunktion, die auf allgemeine, konforme oder flächentreue Kartenentwürfe führen. Es ist dagegen **nicht möglich** Karten zu konstruieren, die sowohl konform als auch flächentreu sind.

Echt geodätische Karten können nur von der Kugel als Modell der Erde erstellt werden.
Beim Drehellipsoid ist das nicht möglich.

3.3 Konforme Entwürfe

3.3.1 Konformer Kegelentwurf

Der allgemeine Ansatz eines Kegelentwurfs sieht wie folgt aus:

$$\bar{u}_1 = n * u_1$$

$$\bar{u}_2 = p(u_2)$$

Da bei konformen Entwürfen $\lambda_1 = \lambda_2$ gilt, kommen wir hier konkret auf:

$$\frac{n^2 p^2}{cos^2(u_2)} = (\frac{\partial p(u_2)}{\partial u_2})^2$$

Nach Trennen der Variablen für festes n erhalten wir durch Integration:

$$p(u_2) = C * tan^{\pm n}(\frac{\pi}{4} + \frac{u_2}{2})$$

Wobei für ein seitenrichtiges Bild der Karte das positive Vorzeichen von n gewählt werden muss. Die geometrische Deutung der Konstanten C ist eine Streckung aus dem Ursprung.

Hat der Entwurf weiters einen Berührkreis in a_2 mit $0 < a_2 < \frac{\pi}{2}$ ($n = sin(a_2)$!) kann C einfach bestimmt werden, da folgendes gelten muss:

$$p(a_2) = C * tan^{-n}(\frac{\pi}{4} + \frac{a_2}{2}) = cot(a_2)$$

Also:

$$C = cot(a_2) * tan^{sin(a_2)}(\frac{\pi}{4} + \frac{a_2}{2})$$

3.3.2 Konformer Azimutalentwurf

Der allgemeine Ansatz eines Azimutalentwurfs sieht wie folgt aus:

$$\bar{u}_1(u_1) = u_1$$

$$\bar{u}_2(u_2) = p(u_2)$$

Wenn man das nun mit dem Ansatz eines allgemeinen Kegelentwurfs vergleicht, erkennt man, dass hier $n = 1$ ist und somit die Formel für den konformen Azimutalentwurf so aussieht:

$$p(u_2) = C * cot(\frac{\pi}{4} + \frac{u_2}{2})$$

Die Konstante C repräsentiert hier den Maßstab der Karte.

3.3.3 Konformer Zylinderentwurf

Der allgemeine Ansatz eines Zylinderentwurfs sieht wie folgt aus:

$$\bar{x}_1 = n * u_1$$

$$\bar{x}_2 = g(u_2)$$

Da bei konformen Entwürfen $\lambda_1 = \lambda_2$ gilt, kommen wir hier konkret auf:

$$\frac{n}{cos(u_2)} = \frac{\partial g(u_2)}{\partial u_2}$$

Durch Integration erhalten wir:

$$g(u_2) = \pm n * log(tan(\frac{\pi}{4} + \frac{u_2}{2})) + C$$

Für ein seitenrichtiges Bild der Karte muss hier das positive Vorzeichen von n gewählt werden. n bedeutet hier wieder eine Streckung aus dem Ursprung.

Soll der Entwurf 2 längentreue Breitenkreise in $\pm a_2$ (mit $0 < a_2 < \frac{\pi}{2}$) haben, wählt man einfach $n = cos(a_2)$ und $C = 0$ und erhält dann:

$$g(u_2) = cos(a_2) * log(tan(\frac{\pi}{4} + \frac{u_2}{2}))$$

Anmerkung:
Den nach MERCATOR (1512 - 1594) benannten Entwurf (auch als "Seekarte" bekannt) erhalten wir mit $n = 1$ und $C = 0$:

$$g(u_2) = log(tan(\frac{\pi}{4} + \frac{u_2}{2}))$$

4 Software zur Erstellung von Kartenentwürfen

4.1 Geografische Informationssysteme - GIS

Alle geografischen Informationssysteme sind nach dem gleichen Prinzip aufgebaut: Sie bestehen aus Hard- und Software, sowie diversen Anwendungen, die mittels einer Datenbank raumbezogene Daten analysieren, reorganisieren und modellieren.

Egal ob wirtschaftliches, wissenschaftliches oder rein informatives Interesse, je nach Anwendungsgebiet kommen hier unterschiedliche Programme zum Einsatz. Natürlich hängt auch die richtige Wahl der Source-Daten vom gewünschten Ergebnis der Anwendung ab. Genauigkeit, Inhalt der Daten (beispielsweise nur Umrisse, oder zusätzliche Informationen über die Topologie und Infrastruktur einer Region), etc. sind entscheidend bei der richtigen Wahl der Software und der Source-Daten.

Es gibt Software verschiedenster Hersteller, die entweder lokal auf dem Computer(Desktop) zum Einsatz kommen, oder auch teilweise interaktiv übers Internet (Client/Server-Systeme) betrieben werden. Gerade diese junge Technologie Internet-GIS (oder Web-GIS) wird immer wichtiger, da natürlich das Client/Server-Prinzip beachtliche Vorteile bietet.
(Mehr zu den Vor- und Nachteilen wird später erklärt.)

Im Detail läuft die Kommunikation eines Web-GIS folgendermaßen ab:

Der Client stellt über ein Internetprotokoll eine Anfrage an einen HTTP-Server der diese Anfrage an eine GIS-Schnittstelle weiterleitet. Die Anfrage wird dann nach spezifischen Routinen abgearbeitet und an das jeweilige zugrundeliegende GIS übermittelt. Dort werden die genauen Kommandos interpretiert und auf die Datenbank zugegriffen. Analyseskripten bearbeiten die Datenbank und erstellen einen Bericht bzw. eine Karte, die wiederrum an die GIS-Schnittstelle zurückübermittelt wird. Abschließend erfolgt die Weiterleitung an den HTTP-Server, von dem der Client die fertige Karte erhält.

Bei solchen Web-GIS unterscheidet man 2 Systeme:

1. Geodaten-Server-Systeme
2. Online GIS Systeme

4.1.1 Geodaten-Server-Systeme

Bei **Geodaten-Server-Systemen** besitzt ein Desktop-GIS die Möglichkeit, Daten von einem entfernten Rechner zu laden und diese lokal weiterzuverarbeiten. Der Rechner, der die Daten zum Download bereit stellt, wird als Geodaten-Server bezeichnet.
Da hier die Kartenerstellung, also die gesamte Software und alle mathematischen Routinen zur grafischen Darstellung lokal am Desktop ausgeführt werden, spricht man von einem **'thick client'**.
Eine genaue Recherche über die Source-Daten, d.h. welche Daten genau benötigt werden, was die Daten beinhalten und wo sie sich befinden, sowie spezifische Kenntnisse über die lokale GIS-Software, sind hier unumgänglich und daher ein deutlicher Nachteil dieser Systeme.

4.1.2 Online GIS Systeme

Als **Online-GIS** werden vor allem
statische und dynamische Map-Server-Systeme bezeichnet.

Bei **statischen** Map-Server-Systemen stellt der Server fertige Karten zur Verfügung, die der lokale Anwender nur mehr mit eingeschränkter Funktionalität bearbeiten bzw. betrachten kann. Der Anwender wird hier als **'thin client'** bezeichnet.
Auch die HTML-Technik der 'imagemaps' verwendet statische Karten.
Die Möglichkeit, einzelne Kartenauschnitte näher zu betrachten ist nicht dynamisch, sondern lediglich eine einfache Zoom-Funktion der statischen Karte.

Die folgende Grafik zeigt eine einfache Anfahrtsskizze aus dem Internet:

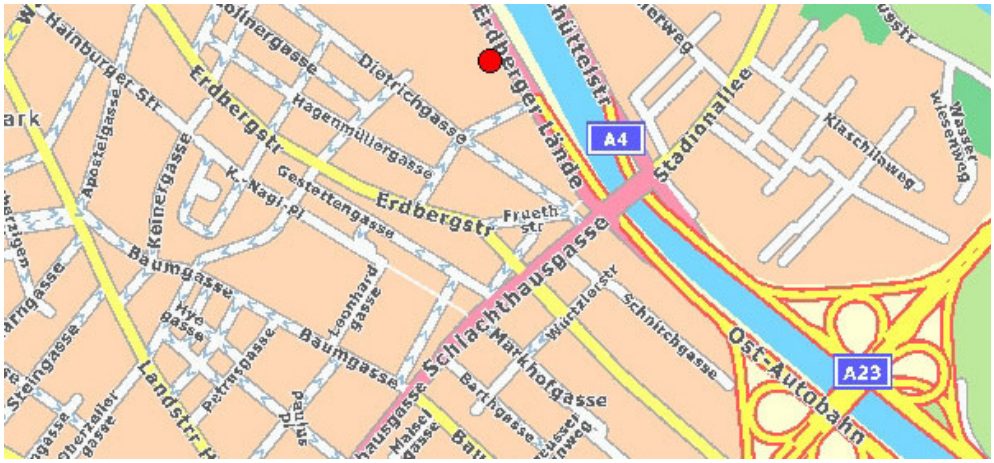


Abbildung 7: Anfahrtsskizze

Bei **dynamischen** Map-Server-Systemen sendet der Client die Parameter des jeweiligen Kartenentwurfs an den Server, der dann die Anfrage auswertet und die Karte dynamisch erstellt.
Als Client-Software wird hier meist ein einfaches PlugIn für einen Webbrowser verwendet, das dem Anwender gewisse Navigationsmöglichkeiten zur Verfügung stellt. Diese Technologie kommt z.B. bei etlichen Online-Informationssystemen zum Einsatz.

Die folgende Grafik zeigt einen dynamischen Map-Server-Client.
(Der Client wird als PlugIn in einem Standard-Webbrowser ausgeführt.)

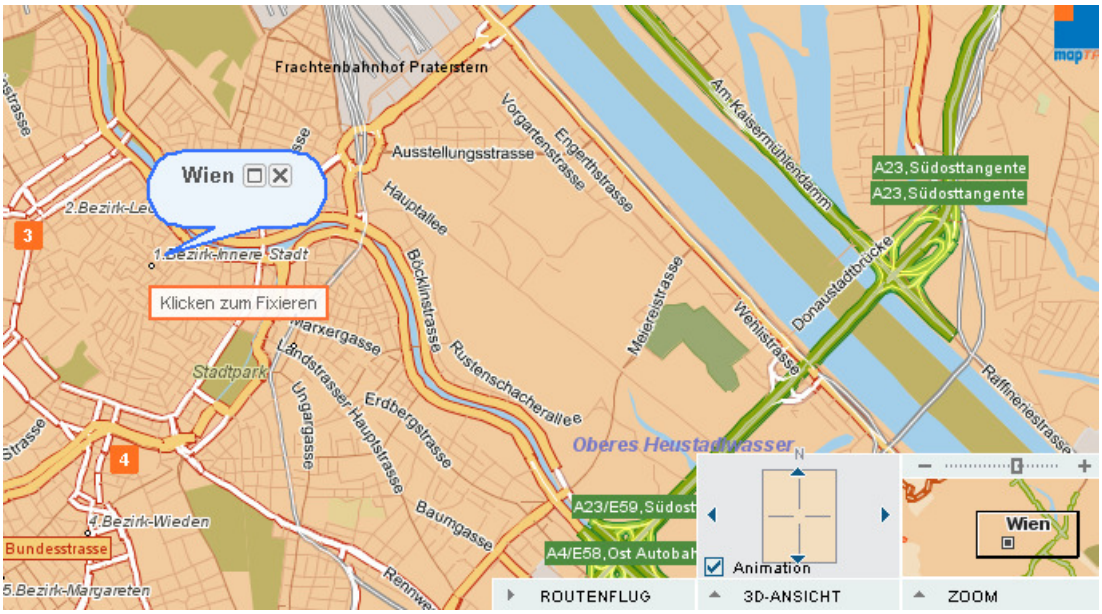


Abbildung 8: dynamischer Map-Server-Client

Der allgemeine Vorteil von Online-GIS liegt darin, das der Anwender Kartenentwürfe ohne spezifisches Fachwissen und ohne spezieller lokaler Software interaktiv erstellen kann. Lediglich eine Internetverbindung ist nötig.

4.2 Datenstrukturen

4.2.1 XML

Die Extensible Markup Language (engl. für 'erweiterbare Auszeichnungssprache'), abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten. Strukturierte Daten findet man z.B. in verschiedenen Dingen wie Datenbanken, Kalkulationstabellen, Adressbüchern, technischen Zeichnungen u.v.m. XML ist ein Satz von Regeln (Konventionen) für die Erstellung von Textdateien zur Strukturierung solcher Daten.
Es handelt sich dabei um keine Programmiersprache, es hilft lediglich dem Computer Daten zu generieren oder einzulesen.
Weiters ist XML plattformunabhängig und wird bevorzugt zum Datenaustausch zwischen unterschiedlichen IT-Systemen verwendet. Datenbankkommunikation übers Internet ist dabei eines der Hauptanwendungsgebiete.
Die Namen der Strukturelemente einer XML-Datei lassen sich dabei frei wählen. XML-Elemente können ganz unterschiedliche Daten enthalten und beschreiben. Sehr oft beinhalten sie reine Textinformationen, oder auch abstrakte Parameter für spezielle Anwendungen.
Ein Grundgedanke hinter XML ist es, Daten und ihre Visualisierung zu trennen, um sie beispielsweise einmal als Tabelle und einmal als Grafik auszugeben, aber für beide Arten der Auswertung die gleiche Source-Datei im XML-Format zu nutzen.

Ein XML-Dokument ist **wohlgeformt**, wenn es sämtliche Regeln für XML einhält. Beispiele solcher Regeln sind folgende:

- Das Dokument besitzt genau ein Wurzelelement.
- Alle Elemente mit Inhalt besitzen ein Beginn- und ein End-Tag, welche mit definierten Symbolen angezeigt werden.
- Die Beginn- und End-Tags sind korrekt verschachtelt.
- Ein Element darf nicht mehrere Attribute mit dem gleichen Namen besitzen.

Durch diesen Aufbau sieht XML ein wenig wie HTML aus. Beide verwenden Tags (durch '<' und '>' geklammerte Wörter) und Attribute (values).
Während HTML festlegt, was jedes Tag und Attribut bedeutet, und oft wie der Text dazwischen in einem Browers aussieht, benutzt XML die Tags nur zur Abgrenzung von Daten und überläßt die Interpretation der Daten allein der jeweiligen Anwendung, die sie verarbeitet.

Soll XML für den Datenaustausch verwendet werden, ist es von Vorteil, wenn das Format mittels einer Grammatik (z. B. einer Dokumenttypdefinition (DTD) oder eines XML-Schemas) definiert ist. Der Standard definiert ein XML-Dokument als gültig, wenn es wohlgeformt ist, den Verweis auf eine Grammatik enthält und das durch die Grammatik beschriebene Format einhält.

Zur Geschichte von XML sei folgendes kurz erwähnt:
Die Entwicklung der XML-Technologie begann 1996, davor gab es aber bereits SGML (Structured Generalized Markup Language), das in den frühen 80er Jahren entwickelt wurde, seit 1986 eine ISO-Norm ist und eine breite Anwendung für große Dokumentationsprojekte fand. Die Entwicklung von HTML begann dann 1990 und die Entwickler von XML nahmen aufgrund der Erfahrung mit HTML einfach die besten Teile von SGML und produzierten etwas, was nicht weniger mächtig als SGML, aber bei weitem geregelter und einfacher in der Anwendung ist.
Allgemein kann man sagen, dass SGML zumeist für technische Dokumentation und viel weniger für andere Daten verwendet wird. Bei XML ist das genau umgekehrt. Anders gesagt: Obwohl XML nur etwa 20 Prozent der Komplexität von SGML besitzt, können damit ca. 80 Prozent der Anwendungsgebiete abgedeckt werden.

Da XML plattformunabhängig und völlig lizenzfrei ist, findet man Zugang zu einer großen und wachsenden Palette an Werkzeugen sowie zu einer Menge an versierten Fachleuten.

Einige Beipiele, wo XML zur Anwendung kommt, sind etwa im Bereich Multimedia MPEG-Dateien oder im Bereich der Kartographie GPSX (GPS Exchange Format) zur GPS-Navigation und natürlich KML-Dateien, die von GOOGLE earth, eines der wohl bekanntesten GIS Programmen, verwendet werden. Dazu im nächsten Abschnitt mehr.

4.2.2 KML - Daten aus GOOGLE earth

Die später genauer dokumentierte selbst entwickelte Software **KARTO** verwendet als Eingabe-Daten Dateien vom Typ **KML**, also jenes Dateiformat, das auch bei GOOGLE earth zu finden ist. KML ist die Abkürzung für Keyhole Markup Language und basiert namentlich auf einem Produkt der ehemaligen kalifornischen Satelittenbildfirma Keyhole Corporation, die von GOOGLE im Oktober 2004 übernommen wurde.
KML ist eine Anwendung von XML 1.0 und dient als spezielles 'Austausch'-Dateiformat für die Client-Programme von GOOGLE earth, also als Kommunikationsschnittstelle zwischen dem GOOGLE earth Server und den Anwenderprogrammen auf den jeweiligen lokalen Computern. In diesen Dateien können neben den geografischen Daten (Punkte, Linien, Polygone, Bilder) auch

andere Parameter wie z.B. Betrachtungswinkel und Entfernungen gespeichert sein, die aber für **KARTO** nicht benötigt, und somit auch nicht weiter erwähnt werden.

Aufgrund dieser Vielfältigkeit und der lizenzfreien Benutzung von GOOGLE earth werden KML-Dateien auch für Reisedokumentationen, Modellierung wissenschaftlicher Inhalte (Geo-Strukturen) oder von Architekten zur Visualisierung von Plänen erstellt.

Die folgende Grafik zeigt einen Auszug einer solchen KML-Datei.

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <Document xmlns="http://earth.google.com/kml/2.0">
    <name>GPS TrackMaker</name>
    <visibility>1</visibility>
  - <Style id="waypoint">
    - <icon>
        <href>root://icons/bitmap-4.png?x=32&y=160&w=32&h=32</href>
      </icon>
    </Style>
  - <Style id="route">
    - <icon>
        <href>root://icons/bitmap-4.png?x=160&y=0&w=32&h=32</href>
      </icon>
    </Style>
  - <Folder>
      <name>Tracklogs</name>
      <visibility>1</visibility>
    - <Placemark>
        <name>FI</name>
      - <Style>
        - <PolyStyle>
            <color>80000000</color>
          </PolyStyle>
        </Style>
      - <Polygon>
          <altitudeMode>relativeToGround</altitudeMode>
        - <outerBoundaryIs>
          - <LinearRing>
              <coordinates>23.0172882080078,60.1012191772461,0
                23.0427761077881,60.1187438964844,0 23.0251369476318,60.1286087036133,0
                22.9674987792969,60.1119384765625,0 22.9061088562012,60.1263885498047,0
                22.8744430541992,60.1455535888672,0 22.8697204589844,60.1791610717773,0
                22.9043025970459,60.2158317565918,0 22.9268035888672,60.2309684753418,0
```

Abbildung 9: Beispiel einer KML-Datei

Wie einfach zu erkennen ist, sind auch hier alle geografische Daten als Punktliste, also als Polygonzug gepeichert. Die später zu sehenden Beispiel-Kartenentwürfe zeigen immer die Länder- und Kontinentalumrisse der gesamten Erde, wobei natürlich auch andere Source-Dateien, z.b. einzelne Länder, möglich sind.
Mathematisch gesehen sind alle denkbaren Punklisten in **KARTO** darstellbar, wobei alle Polygonzüge einer KML-Datei geschlossen sind, d.h. der erste und letzte Punkt sind abschnittsweise ident. Für **KARTO** ist das allerdings nicht unbedingt erforderlich.

4.2.3 Dateninterface

Zur Vereinfachung wurde in **KARTO** ein Dateninterface integriert, dass die Source-KML-Datei einliest und eine einfache Text-Datei (TXT) mit allen notwendigen Informationen erstellt. Diese Text-Datei enthält dann nur mehr die geografischen Koordionaten als Punktliste. Zusammengehörige Längen- und Breitenangaben stehen untereinander, und als Trennung der einzelnen Abschnitte (geschlossene Polygonzüge) wird das Wort 'NEU' verwendet.

Die folgende Grafik zeigt einen Auszug einer solchen TXT-Datei.

```
NEU
23.0172882080078
60.1012191772461
23.0427761077881
60.1187438964844
23.0251369476318
60.1286087036133
22.9674987792969
60.1119384765625
22.9061088562012
60.1263885498047
22.8744430541992
60.1455535888672
22.8697204589844
60.1791610717773
22.9043025970459
60.2158317565918
22.9268035888672
60.2309684753418
22.9677753448486
60.2502746582031
23.017219543457
60.2822189331055
```

Abbildung 10: Beispiel einer von KARTO generierten TXT-Datei

4.3 Software KARTO in BORLAND Delphi

4.3.1 Idee

Die Idee bzw. Aufgabenstellung der Software **KARTO** ist die Erstellung verschiedener Kartenentwürfe, wie sie z.B. im Kapitel 3 bereits vorgestellt wurden. Die erstellten Karten sollen in einer ansprechenden Qualität und variabler Größe am Bildschirm dargestellt, sowohl als Pixelgrafik und Vektorgrafik gespeichert und natürlich auch mit variabler Auflösung und Größe ausgedruckt werden können.

Die Software soll standardgemäß unter aktuellen WINDOWS-Version betrieben werden, und die Benutzeroberfläche soll eine intuitive Verwendung ermöglichen.

Weiters soll **KARTO** soweit flexibel sein, dass nicht nur eine fixe Source-Dateien sondern verschiedene Daten eingelesen und dargestellt werden können.
Als Datei-Format wird XML (siehe Kapitel 4.2 Datenstrukturen) verwendet, was eine gewisse zukunftsorientierte Verwendung garantiert.

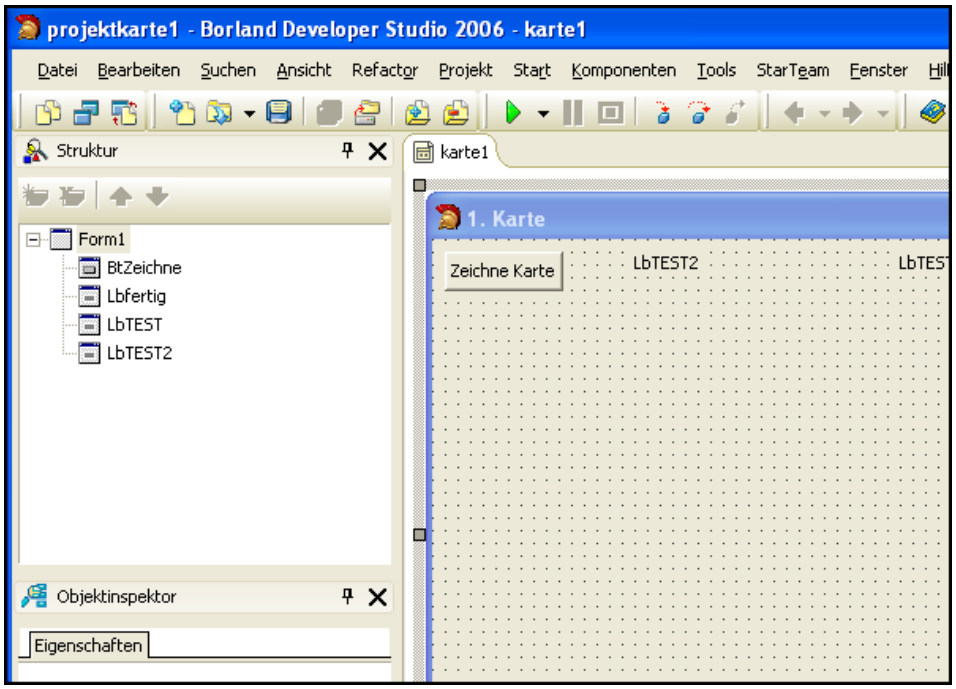
4.3.2 Borland Developer Studio 2006 - Warum Delphi?

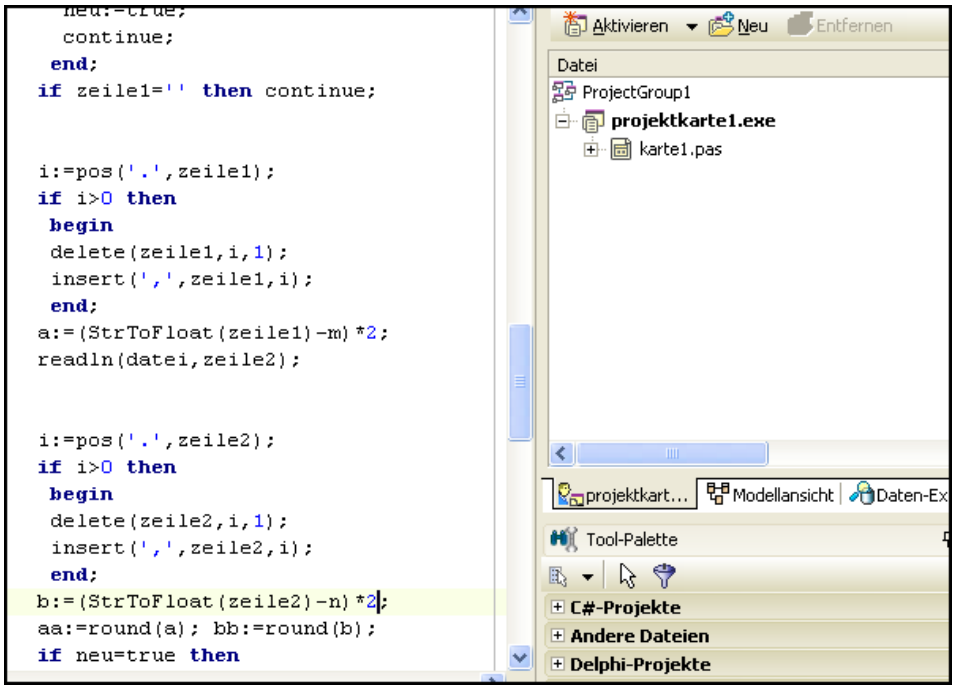
Delphi ist eine von der Firma Borland entwickelte objektorientierte Programmiersprache, die ursprünglich aus der von Niklaus Wirth erstellten Programmiersprache Pascal hervorgegangen ist.
Delphi ist weiters ein Teil der von Borland entwickelten Softwareentwicklungsumgebung **Borland Developer Studio**, welches auch andere Programmiersprachen wie z.B. JAVA oder C++ beinhaltet. Delphi bzw. das Borland Developer Studio ist somit ein zeitgemäßes Werkzeug zur Erstellung von Anwendersoftware für vor allem Windows-basierte Systeme.
Im November 2006 wurden die Entwicklerteams des Studios, darunter auch Delphi, von Borland in eine neue Tochtergesellschaft namens CodeGear ausgegliedert.

Bei der Entwicklung von **KARTO** wurde die derzeit aktuellste Version **Borland Developer Studio 2006 (BDS 2006)**, welches Delphi WIN32 beinhaltet, verwendet, welches mir vom Institut für Diskrete Mathematik und Geometrie der TUWIEN zur Verfügung gestellt wurde.

Da ich bereits seit langem mit Delphi (und auch mit den Vorgängerversionen) arbeite, war es für mich klar auch **KARTO** in Delphi zu entwickeln.

4.3.3 Screen-Shots der Entwicklungsumgebung BDS 2006



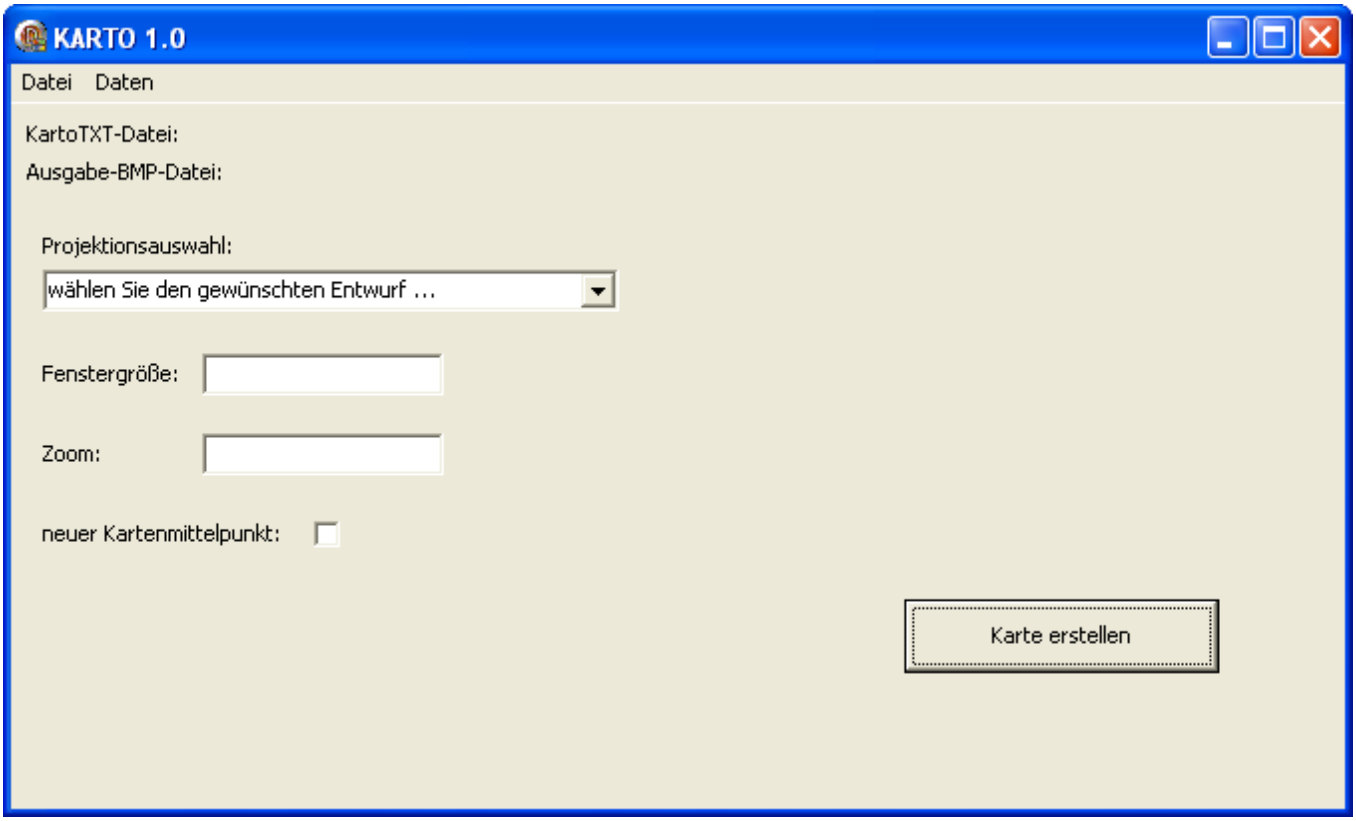


4.4 KARTO - ein kurzes Handbuch

Bei allen Gradangaben in **KARTO** werden Altgrad verwendet.

4.4.1 Hauptprogramm

Screenshot:



Hauptmenu: Datei

Mit **KartoTXT wählen** wird die Eingabe-KartoTXT gewählt.
Die dortigen Informationen dienen zur Erstellung der Karte.
Dieser Menupunkt kann auch mit STR-A aufgerufen werden.

Mit **Ausgabe-BMP-Datei wählen** wird die Bitmap-Datei gewählt, in die die Karte gespeichert wird.
Dieser Menupunkt kann auch mit STR-S aufgerufen werden.

Beenden schließt das Programm.

Hauptmenu: Daten

Mit **KML − > KartoTXT** wird das Tool zum Konvertieren einer KML-Datei in eine KartoTXT-Datei gestartet.

Mit **2 KartoTXT kombinieren** wird das Tool zum Kombinieren 2er KartoTXT-Dateien aufgerufen, um etwa 2 Karten oder eine Karte mit einer Gradnetz-KartoTXT-Datei zu verbinden.

Mit **Gradnetz erstellen** wird das Tool zur Gradnetzerstellung aufgerufen.

Programmfenster:

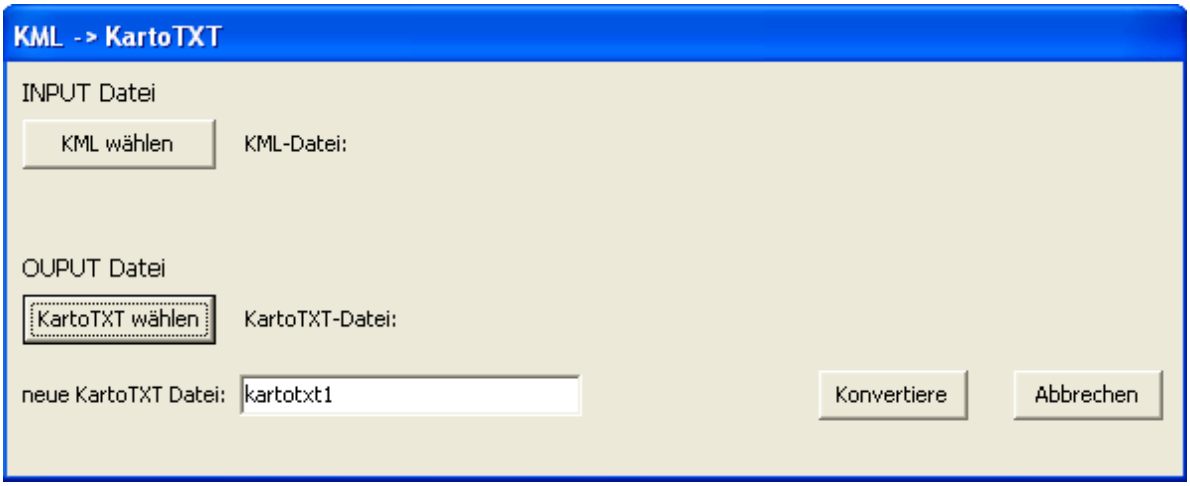
Mit der Auswahlbox **Projektionsauswahl** können verschiedene Kartenentwürfe ausgewählt werden. Nach einer Auswahl werden automatisch die Parameter für die **Fenstergröße** und den **Zoom** vorgeschlagen, die aber natürlich noch editiert werden können. Weiters werden bei manchen Kartenentwürfen weitere Eingabemöglichkeiten wie z.B. 'Berührkreis' oder 'Iterationsschritte' angezeigt.

Mit der Auswahl **neuer Kartenmittelpunkt** werden 2 Eingabefelder angezeigt, wo man den Kartenmittelpunkt festsetzen kann, um z.B. mit höheren Zoom-Werten Detailansichten einer Karte zu erstellen.

Der Button **Karte erstellen** startet die Berechnung, zeigt die Karte in einem neuen Fenster an und speichert die Karte in die gewählte BMP-Datei.

4.4.2 Tool: Konvertiere XML > KartoTXT

Screenshot:



Der Button **KML wählen** dient zur Auswahl der Quell-KML-Datei.

Der Button **KartoTXT wählen** dient zur Auswahl der KartoTXT-Datei, in die die Punktlisten-Informationen aus der gewählten KML-Datei geschrieben werden.

Der Button **Konvertiere** startet die Berechnung und erstellt die KartoTXT-Datei. Nach der Konvertierung wird das Tool-Fenster automatisch geschlossen.

Der Button **Abbrechen** beendet das Tool und schließt das Tool-Fenster.

Anmerkungen:
Die Berechnung der Ziel-KartoTXT-Datei kann bei großen KML-Dateien einige Minuten in Anspruch nehmen. Nach erfolgreicher Konvertierung wird das Tool-Fenster geschlossen.
Für die KML-KartoTXT-Konvertierung sind 2 Übergangs-TXT-Dateien notwendig, deren Name und Ort in der Datei: 'parameter.txt' gespeichert sind.
Diese Parameter-Datei kann mit jedem Standard-Editor geöffnet werden.

4.4.3 Tool: Kombiniere 2 KartoTXT-Dateien

Screenshot:

Die Buttons **1. KartoTXT wählen** und **2. KartoTXT wählen** dienen zur Auswahl der beiden zu kombinierenden KartoTXT-Dateien.

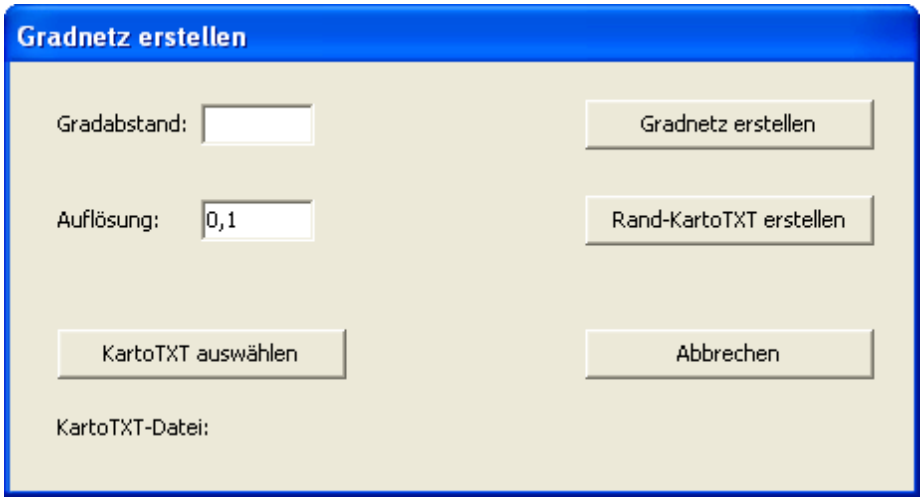
Der Button **Ziel-KartoTXT wählen** dient zur Auswahl der KartoTXT-Datei, in die die zusammengefügte Punktliste der beiden Eingabedateien geschrieben wird.

Der Button **KartoTXT kombinieren** startet die Berechnung und erstellt die Ziel-KartoTXT-Datei.
Nach erfolgreicher Berechnung wird das Tool-Fenster geschlossen.

Der Button **Abbrechen** beendet das Tool und schließt das Tool-Fenster.

4.4.4 Tool: Gradnetz erstellen

Screenshot:



Im Feld **Gradabstand** kann der Abstand der einzelnen Gradnetzlinien (Breitenkreise und Längenkreise) eingegeben werden.
Sinnvolle Eingaben wären hier z.B. 15, 30 oder 45. Andere Werte sind natürlich auch möglich, wobei dann gewisse prägnante Linien z.B. bei $\pm$ 180 Grad möglicherweise nicht erstellt werden. (kein Teiler von 180!)

Im Feld **Auflösung** kann die Anzahl der Punkte entlang einer Gradnetzlinie festgelegt werden. Der vorgeschlagene Wert $0,1$ bedeutet somit, dass ein Breitenkreis von -180 bis $+180$ aus 3600 Teilpunkten (Polygonzug) besteht.
(- analog: Halber Längenkreis von -90 bis 90 aus 1800 Teilpunkten)

Der Button **KartoTXT auswählen** dient zur Auswahl der KartoTXT-Datei, in die die Punktliste des Gradnetztes geschrieben wird.

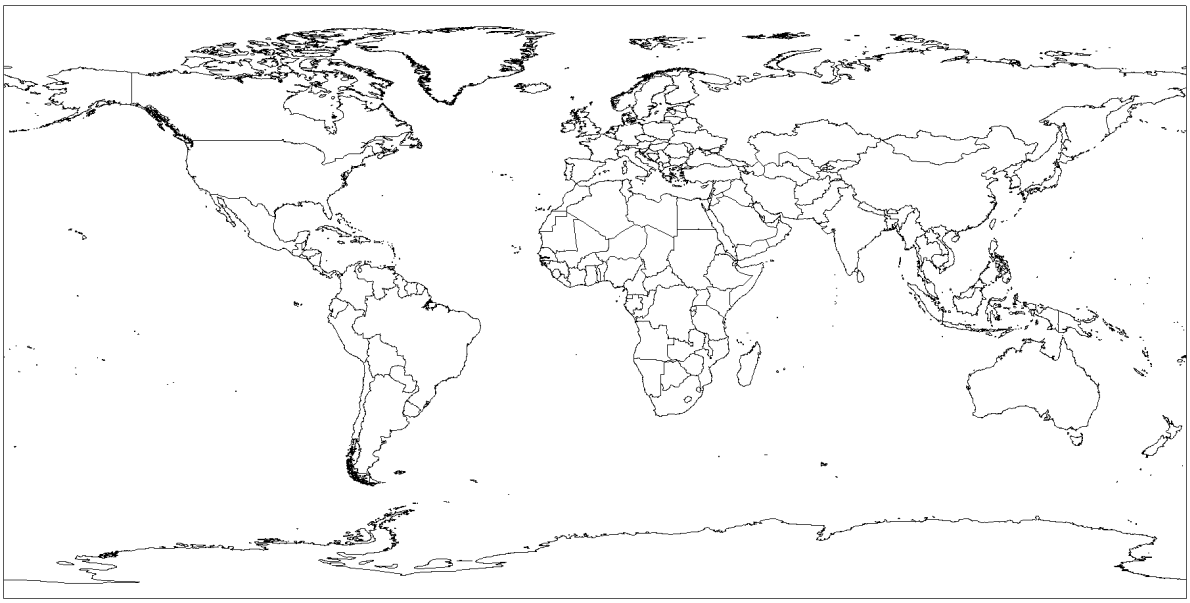
Der Button **Gradnetz erstellen** startet die Berechnung und erstellt die ausgewählte KartoTXT-Datei. Danach wird das Tool-Fenster geschlossen.

Der Button **Rand-KartoTXT erstellen** erstellt eine KartoTXT-Datei, die den gesamten Rand einer Karte - also die Linien $(\pm 180|v)$ und $(u| \pm 90)$ mit $v \in [-90, 90]$ und $u \in [-180, 180]$ - enthält. Der Name der KartoTXT-Datei kann dabei frei gewählt werden. Danach wird das Tool-Fenster geschlossen.

Der Button **Abbrechen** beendet das Tool und schließt das Tool-Fenster.

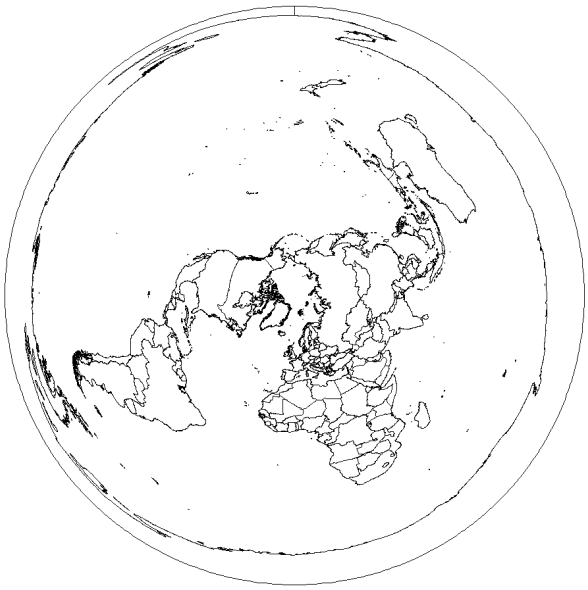
4.4.5 Beispiele von mit KARTO erstellten Kartenentwürfen

Abstandstreuer Zylinderentwurf:

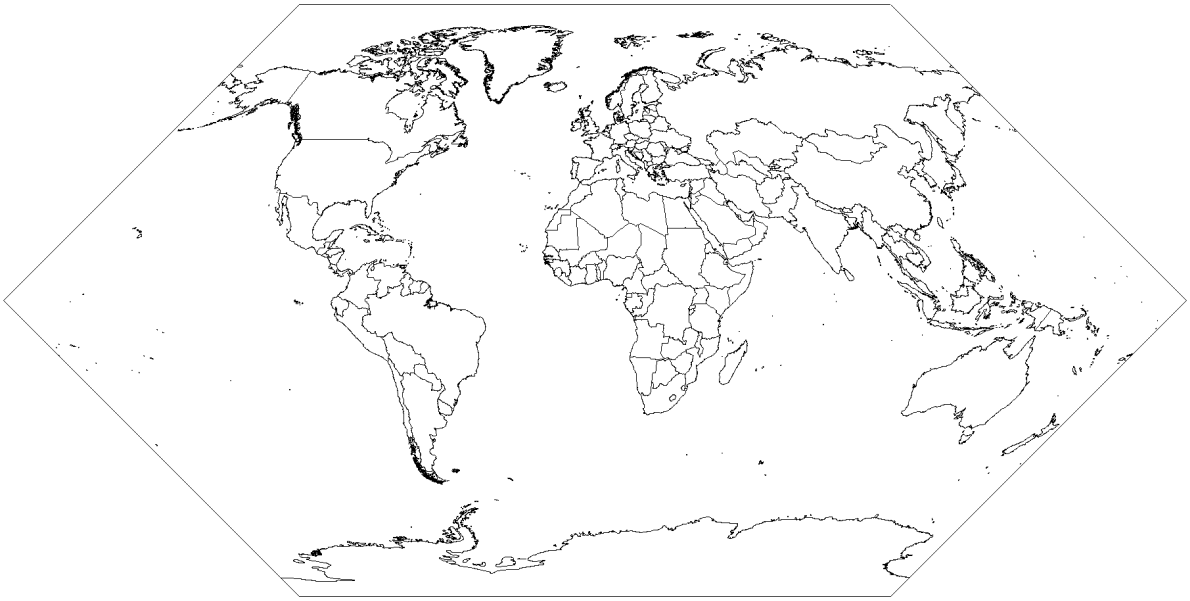


Abstandstreuer Kegelentwurf:

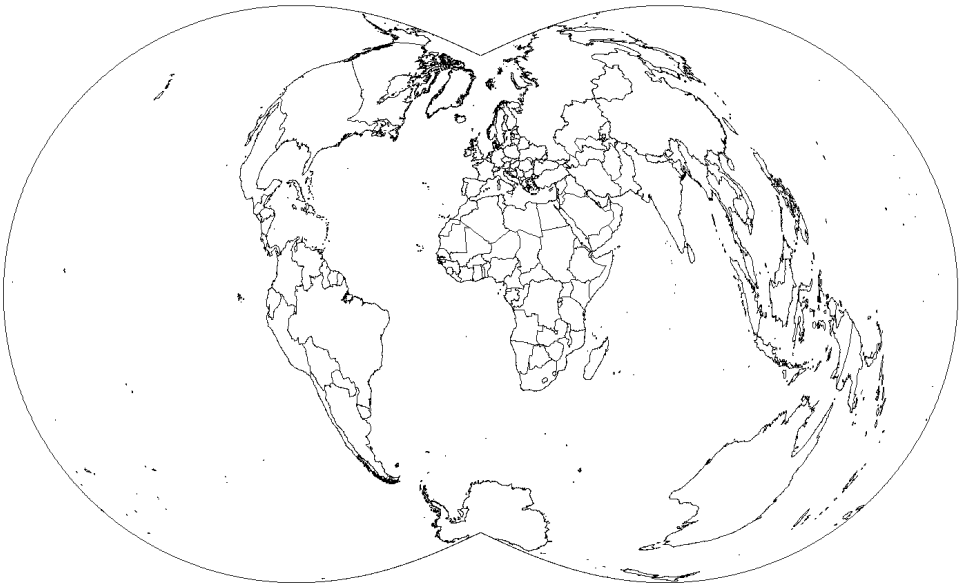
Abstandstreuer Azimutalentwurf:



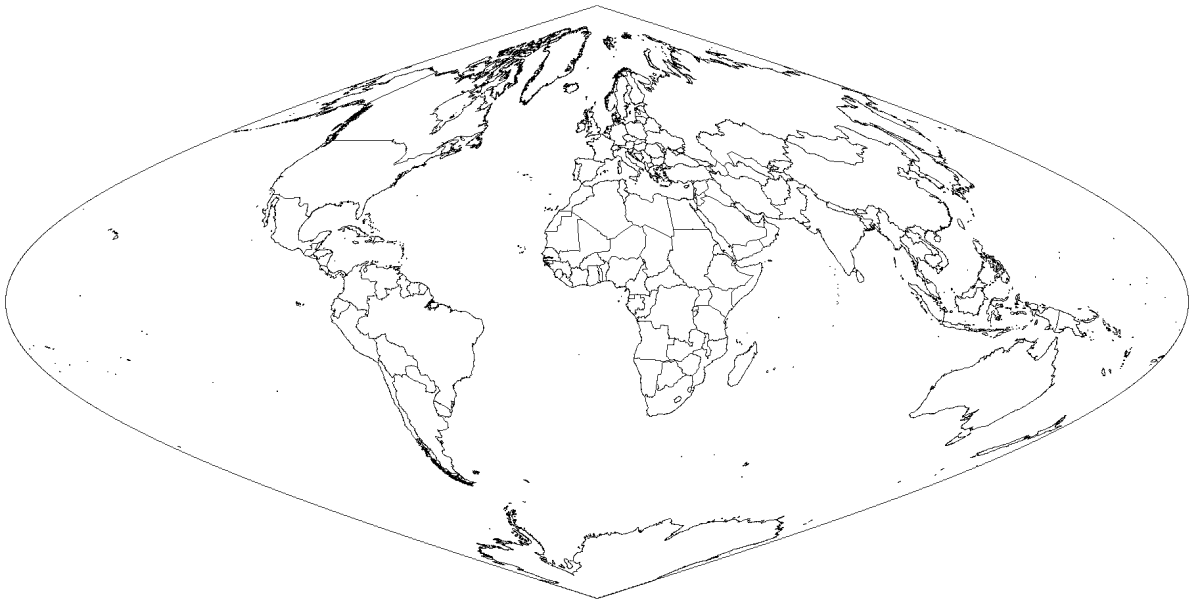
Trapezentwurf von ECKERT:



Orthogonaler polykonischer Entwurf:

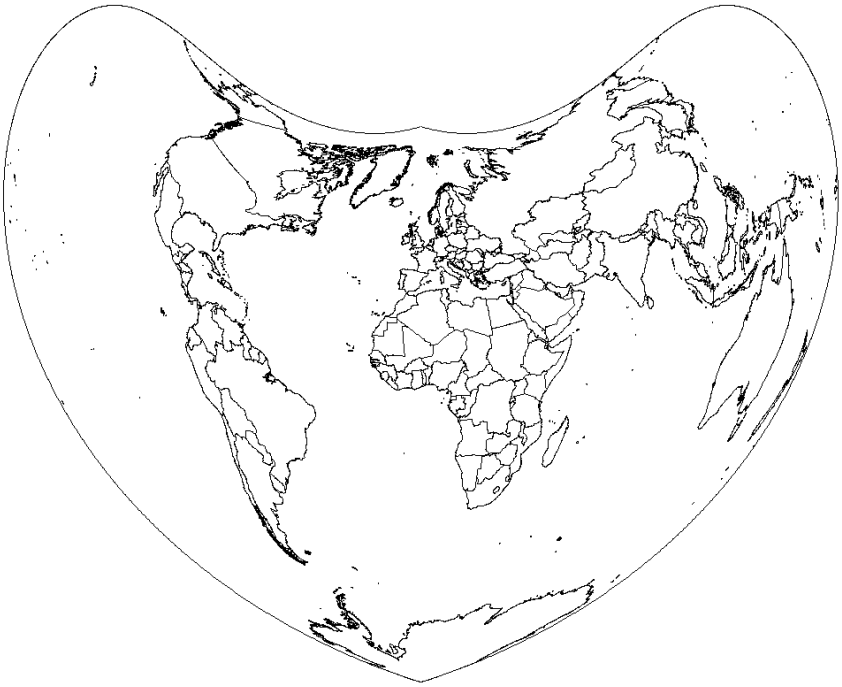


Entwurf von SANSON:



Entwurf von MOLLWEIDE:

Entwurf von BONNE:



4.5 Quellcode von KARTO

4.5.1 Hauptformular

```
unit formular_main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, Menus, formular_konvert, formular_zeichnung, StdCtrls, Math,
  formular_gradnetz, formular_kombi;

type
  THauptformular = class(TForm)
    MainMenu1: TMainMenu;
    Datei1: TMenuItem;
    Daten1: TMenuItem;
    KMLKartoTXT1: TMenuItem;
    Beenden1: TMenuItem;
    LbKartoTXT: TLabel;
    OpenDialogKartoTXT: TOpenDialog;
    KartoTXTwhlen1: TMenuItem;
    BtKarteerstellen: TButton;
    CBProjektionsauswahl: TComboBox;
    LbProjektionsauswahl: TLabel;
    LbSkalierung: TLabel;
    EdSkalierung: TEdit;
    LbZoom: TLabel;
    EdZoom: TEdit;
    LbKartenmittelpunkt: TLabel;
    LbMittex: TLabel;
    LbMittey: TLabel;
    EdKartenmittelpunktx: TEdit;
    EdKartenmittelpunkty: TEdit;
    LbErrorKartoTXT: TLabel;
    Checkkartenmittelpunkt: TCheckBox;
    EdBKreis: TEdit;
    LbBKreis: TLabel;
    LbBKreis2: TLabel;
    EdBKreis2: TEdit;
    Gradnetzerstellen: TMenuItem;
    N2KartoTXTkombinieren1: TMenuItem;
    LbBMP: TLabel;
    SaveDialogBMP: TSaveDialog;
    AusgabeBMPDateiwhlen1: TMenuItem;
    SaveDialogParameter: TSaveDialog;
```

```
    procedure AusgabeBMPDateiwhlen1Click(Sender: TObject);
    procedure N2KartoTXTkombinieren1Click(Sender: TObject);
    procedure GradnetzerstellenClick(Sender: TObject);
    procedure CheckkartenmittelpunktClick(Sender: TObject);
    procedure CBProjektionsauswahlChange(Sender: TObject);
    procedure BtKarteerstellenClick(Sender: TObject);
    procedure KartoTXTwhlen1Click(Sender: TObject);
    procedure KMLKartoTXT1Click(Sender: TObject);
    procedure Beenden1Click(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  Hauptformular: THauptformular;
  filename: string;
  bmpoutputfile: string;
  karte: Tbitmap;
  auswahl: boolean;

implementation

{$R *.dfm}

// HAUPTMENU - Programm beenden
procedure THauptformular.AusgabeBMPDateiwhlen1Click(Sender: TObject);
begin
 if SaveDialogBMP.Execute then
  begin
   bmpoutputfile:=SaveDialogBMP.FileName;
   LbBMP.Caption:='Ausgabe-BMP-Datei: ' + bmpoutputfile;
   LbErrorKartoTXT.Visible:=false;
  end;
end;

procedure THauptformular.Beenden1Click(Sender: TObject);
begin
 close();
end;

/// KARTE ERSTELLEN !!!
procedure THauptformular.BtKarteerstellenClick(Sender: TObject);
var
 datei: textfile;
 a,b, skalierung, zoom, mittex, mittey, winkel, r: real;
 rmitte, winkelmitte: real;
 parameterBKreis: real;
 aa,bb,i,l,x,y: integer;
```

```
 zeile1, zeile2: string;
 neu,neu2: boolean;
 index: integer;
 hilf, hilf2, hilf3, hilfm, hilf2m:real;
 n,a2,b2,c: real;
 z,p:real;
 t:real;
 tlow,thigh: real;
 it: integer;
 par1,par2,par3,par4:real;

begin
 if (filename<>'') and auswahl and (bmpoutputfile<>'') then
  begin
   FormularZeichnung.Caption:=CBProjektionsauswahl.Text;
   LbErrorKartoTXT.Visible:=False;
   skalierung:=StrToFloat(EdSkalierung.Text);
   zoom:=StrToFloat(EdZoom.Text);
   mittex:=StrToFloat(EdKartenmittelpunktx.Text);
   mittey:=StrToFloat(EdKartenmittelpunkty.Text);

   // Kartenmittelpunkt (Fenstermittelpunkt)
   x:=round((180*skalierung)+5);
   y:=round((90*skalierung)+5);

   // Größe des Kartenfensters wird festgelegt - Fenster wird angezeigt
   with FormularZeichnung do
    begin
     Height:=trunc(180*skalierung)+45;
     Width:=trunc(360*skalierung)+20;
     Show;
    end;

   // Bitmap wird erstellt und die Größe festgelegt
   karte:=Tbitmap.Create;
   karte.Height:=round(180*skalierung)+20;
   karte.Width:=round(360*skalierung)+20;

   // KartoTXT wird geöffnet
   Assignfile(datei,filename);
   reset(datei);
   neu:=true;
   // KartoTXT wird sequentiell eingelesen
   while not eof(datei) do

    begin
     // Einlesen der 1. Koordinaten
     readln(datei,zeile1);
     if zeile1='NEU' then
      begin
```

```
  neu:=true;
  continue; // Abbruch der Schleife
 end;
if zeile1='' then continue; // Abbruch der Schleife
// Punkt wird durch Komma ersetzt
i:=pos('.',zeile1);
if i>0 then
 begin
  delete(zeile1,i,1);
  insert(',',zeile1,i);
 end;
// Einlesen der 2. Koordinate
readln(datei,zeile2);
// Punkt wird durch Komma ersetzt
i:=pos('.',zeile2);
if i>0 then
 begin
  delete(zeile2,i,1);
  insert(',',zeile2,i);
 end;

//Erstellen der Anzeigekoordinaten nach Projektionsauswahl
parameterBKreis:=StrToFloat(EdBKreis.Text)/180*Pi;
index:=CBProjektionsauswahl.ItemIndex;

case CBProjektionsauswahl.ItemIndex of

 // Abstandstreuer Zylinderentwurf
 0:
 begin
  if CheckKartenmittelpunkt.Checked=true then
   begin
    a:=(StrToFloat(zeile1)-mittex)*zoom;
    b:=(StrToFloat(zeile2)-mittey)*zoom;
   end
   else
   begin
    a:=(StrToFloat(zeile1))*zoom;
    b:=(StrToFloat(zeile2))*zoom;
   end;
 end;
 // Flächentreuer Zylinderentwurf
 1:
 begin
  if CheckKartenmittelpunkt.Checked=true then
   begin
    a:=(StrToFloat(zeile1)-mittex)*zoom;
    b:=sin(StrToFloat(zeile2)*Pi/180)*180/Pi*zoom-
       sin(mittey*Pi/180)*180/Pi*zoom;
   end
```

```
        else
        begin
         a:=(StrToFloat(zeile1))*zoom;
         b:=sin(StrToFloat(zeile2)*Pi/180)*180/Pi*zoom;
        end;
      end;
      // Konformer Zylinderentwurf
      2:
      begin
       if abs(StrToFloat(zeile2))<90 then
       begin
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=(StrToFloat(zeile1)-mittex)*zoom;
         hilf:=Pi/4+(StrToFloat(zeile2))*Pi/180/2;
         hilf2:=Pi/4+(mittey)*Pi/180/2;
         b:=ln(sin(hilf)/cos(hilf))/ln(10)*180/Pi*zoom-
            ln(sin(hilf2)/cos(hilf2))/ln(10)*180/Pi*zoom;
        end
        else
        begin
         a:=(StrToFloat(zeile1))*zoom;
         hilf:=Pi/4+(StrToFloat(zeile2))*Pi/180/2;
         b:=ln(sin(hilf)/cos(hilf))/ln(10)*180/Pi*zoom;
        end;
       end
       else
       begin
        neu:=true;
        continue;
       end;
      end;
      //Abstandstreuer Kegelentwurf
      4:
      begin
       winkelmitte:=(mittex)/180*(-Pi)*sin(parameterBKreis);
       rmitte:=cos(parameterBKreis)/sin(parameterBKreis)+parameterBKreis-
(mittey/180*Pi);
       winkel:=(StrToFloat(zeile1))/180*(-Pi)*sin(parameterBKreis);
       r:=cos(parameterBKreis)/sin(parameterBKreis)+parameterBKreis-
(StrToFloat(zeile2)/180*Pi);
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=(r*sin(winkel+Pi))*zoom-(rmitte*sin(winkelmitte+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom-(rmitte*cos(winkelmitte+Pi))*zoom;
        end
        else
        begin
         a:=(r*sin(winkel+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom+(90*skalierung);
```

```
  end;
end;
// Flächentreuer Kegelentwurf mit 2 längentreuen Breitenkreisen
5:
begin
 a2:=StrToFloat(EdBKreis.Text)*Pi/180;
 b2:=StrToFloat(EdBkreis2.Text)*Pi/180;
 n:=(sin(a2)+sin(b2))/2;
 c:=sqr((cos(a2)/n))+(2*sin(a2))/n;
 winkelmitte:=n*(mittex)*(-Pi)/180;
 rmitte:=sqrt(c-2/n*sin((mittey)*Pi/180));
 winkel:=n*StrToFloat(zeile1)*(-Pi)/180;
 r:=sqrt(c-2/n*sin(StrToFloat(zeile2)*Pi/180));
 if CheckKartenmittelpunkt.Checked=true then
  begin
   a:=(r*sin(winkel+Pi))*zoom-(rmitte*sin(winkelmitte+Pi))*zoom;
   b:=(r*cos(winkel+Pi))*zoom-(rmitte*cos(winkelmitte+Pi))*zoom;
  end
  else
  begin
   a:=(r*sin(winkel+Pi))*zoom;
   b:=(r*cos(winkel+Pi))*zoom;
  end;
end;
// Konformer Kegelentwurf
6:
begin
 a2:=StrToFloat(EdBKreis.Text)*Pi/180;
 n:=sin(a2);
 hilf:=Pi/4+a2/2;
 c:=cos(a2)/sin(a2)*Power(sin(hilf)/cos(hilf),sin(a2));
 hilf2:=Pi/4+StrToFloat(zeile2)*Pi/180/2;
 hilf3:=Pi/4+(mittey)*Pi/180/2;
 winkelmitte:=n*(mittex)*(-Pi)/180;
 rmitte:=c*Power(sin(hilf3)/cos(hilf3),(-1)*n);
 winkel:=n*StrToFloat(zeile1)*(-Pi)/180;
 r:=c*Power(sin(hilf2)/cos(hilf2),(-1)*n);
 if CheckKartenmittelpunkt.Checked=true then
  begin
   a:=(r*sin(winkel+Pi))*zoom-(rmitte*sin(winkelmitte+Pi))*zoom;
   b:=(r*cos(winkel+Pi))*zoom-(rmitte*cos(winkelmitte+Pi))*zoom;
  end
  else
  begin
   a:=(r*sin(winkel+Pi))*zoom;
   b:=(r*cos(winkel+Pi))*zoom;
  end;
end;
// Kegelprojektion aus Kugelmitte
7:
```

```
      begin
       if StrToFloat(zeile2) < (-1)*(90-StrToFloat(EdBKreis.Text))+0.001
then
       begin
        neu:=true;
        continue;
       end;
       hilf:=StrToFloat(EdBKreis.Text)*Pi/180;
       par1:=cot(hilf);
       par2:=sin(hilf);
       hilf2:=par1 + tan(hilf-StrToFloat(zeile2)*Pi/180);
       hilf2m:=par1 + tan(hilf-mittey*Pi/180);
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=hilf2*sin(par2*StrToFloat(zeile1)*Pi/180)*zoom-
            hilf2m*sin(par2*mittex*Pi/180)*zoom;
         b:=hilf2*cos(par2*StrToFloat(zeile1)*Pi/180)*zoom*(-1)-
            hilf2m*cos(par2*mittex*Pi/180)*zoom*(-1);
        end
        else
        begin
         a:=hilf2*sin(par2*StrToFloat(zeile1)*Pi/180)*zoom;
         b:=hilf2*cos(par2*StrToFloat(zeile1)*Pi/180)*zoom*(-1);
        end;
      end;
      // Polykonischer Entwurf
      9:
      begin
       hilf:=StrToFloat(zeile1)*Pi/180;
       hilf2:=StrToFloat(zeile2)*Pi/180;
       if CheckKartenmittelpunkt.Checked=true then
        begin
         if abs(hilf2)>0.01*Pi/180 then
          begin
           a:=(cot(hilf2)*sin(hilf*sin(hilf2)))*zoom-

(cot(mittey*Pi/180)*sin(mittex*Pi/180*sin(mittey*Pi/180)))*zoom;
           b:=(hilf2+2*cot(hilf2)*sqr(sin(hilf*sin(hilf2))/2))*zoom-

(mittey*Pi/180+2*cot(mittey*Pi/180)*sqr(sin(mittex*Pi/180*sin(mittey*Pi/180
))/2))*zoom;
          end
          else
          begin
           a:=cos(hilf2)*hilf*zoom-
              cos(mittey*Pi/180)*mittex*Pi/180*zoom;
           b:=(hilf2+2*cos(hilf2)*hilf/2*hilf*sin(hilf2)/2)*zoom-

(mittey*Pi/180+2*cos(mittey*Pi/180)*mittex*Pi/180/2*mittex*Pi/180*sin(mitte
y*Pi/180)/2)*zoom;
```

```
          end;
        end
        else
        begin
         if abs(hilf2)>0.01*Pi/180 then
          begin
           a:=(cot(hilf2)*sin(hilf*sin(hilf2)))*zoom;
           b:=(hilf2+2*cot(hilf2)*sqr(sin(hilf*sin(hilf2)/2)))*zoom;
          end
          else
          begin
           a:=cos(hilf2)*hilf*zoom;
           b:=(hilf2+2*cos(hilf2)*hilf/2*hilf*sin(hilf2)/2)*zoom;
          end;
        end;
      end;
      // Orthogonaler polykonischer Entwurf
      10:
      begin
       hilf:=StrToFloat(zeile1)*Pi/180;
       hilf2:=StrToFloat(zeile2)*Pi/180;
       if CheckKartenmittelpunkt.Checked=true then
        begin
         if abs(hilf2)<0.01*Pi/180 then
          begin
           // Äquatornähe
           a:=hilf*zoom-mittex*Pi/180*zoom;
           b:=0-mittey*Pi/180*zoom;
          end
          else
          begin
           if abs(hilf2)>89.99*Pi/180 then
            begin
             // Polnähe
             a:=0-mittex*Pi/180*zoom;
             b:=Pi/2*sign(hilf2)*zoom-Pi/2*sign(mittey*Pi/180)*zoom;
            end
            else
            begin
             // sonst
             a:=cot(hilf2)*sin(2*arctan(hilf/2*sin(hilf2)))*zoom-

cot(mittey*Pi/180)*sin(2*arctan(mittex*Pi/180/2*sin(mittey*Pi/180)))*zoom;
             b:=(cot(hilf2)+hilf2-
cot(hilf2)*cos(2*arctan(hilf/2*sin(hilf2))))*zoom-
                 (cot(mittey*Pi/180)+mittey*Pi/180-
cot(mittey*Pi/180)*cos(2*arctan(mittex*Pi/180/2*sin(mittey*Pi/180))))*zoom;
            end;
          end;
        end
```

```
        else
        begin
         if abs(hilf2)<0.01*Pi/180 then
          begin
           // Äquatornähe
           a:=hilf*zoom;
           b:=0;
          end
          else
          begin
           if abs(hilf2)>89.99*Pi/180 then
            begin
             // Polnähe
             a:=0;
             b:=Pi/2*sign(hilf2)*zoom;
            end
            else
            begin
             // sonst
             a:=cot(hilf2)*sin(2*arctan(hilf/2*sin(hilf2)))*zoom;
             b:=(cot(hilf2)+hilf2-
cot(hilf2)*cos(2*arctan(hilf/2*sin(hilf2))))*zoom;
            end;
          end;
        end;
      end;
      // Abstandstreuer Azimutalentwurf
      12:
      begin
       winkelmitte:=mittex/180*(-Pi);
       rmitte:=Pi/2-(mittey/180*Pi);
       winkel:=(StrToFloat(zeile1))/180*(-Pi);
       r:=Pi/2-(StrToFloat(zeile2)/180*Pi);
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=(r*sin(winkel+Pi))*zoom-(rmitte*sin(winkelmitte+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom-(rmitte*cos(winkelmitte+Pi))*zoom;
        end
        else
        begin
         a:=(r*sin(winkel+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom;
        end;
      end;
      // Flächentreuer Azimutalentwurf
      13:
      begin
       winkelmitte:=mittex/180*(-Pi);
       rmitte:=sqrt(2-2*sin(mittey*Pi/180));
```

```
 winkel:=(StrToFloat(zeile1))/180*(-Pi);
 r:=sqrt(2-2*sin(StrToFloat(zeile2)*Pi/180));
 if CheckKartenmittelpunkt.Checked=true then
  begin
   a:=(r*sin(winkel+Pi))*zoom-(rmitte*sin(winkelmitte+Pi))*zoom;
   b:=(r*cos(winkel+Pi))*zoom-(rmitte*cos(winkelmitte+Pi))*zoom;
  end
  else
  begin
   a:=(r*sin(winkel+Pi))*zoom;
   b:=(r*cos(winkel+Pi))*zoom;
  end;
end;
// Stereographische Projektion
14:
begin
 if StrToFloat(zeile2) < -89.99 then continue;
 hilf:=2*cot(Pi/4+StrToFloat(zeile2)*Pi/180/2);
 hilfm:=2*cot(Pi/4+mittey*Pi/180/2);
 if CheckKartenmittelpunkt.Checked=true then
  begin
   a:=hilf*sin(StrToFloat(zeile1)*Pi/180)*zoom-
      hilfm*sin(mittex*Pi/180)*zoom;
   b:=(-1)*hilf*cos(StrToFloat(zeile1)*Pi/180)*zoom-
      (-1)*hilfm*cos(mittex*Pi/180)*zoom;
  end
  else
  begin
   a:=hilf*sin(StrToFloat(zeile1)*Pi/180)*zoom;
   b:=(-1)*hilf*cos(StrToFloat(zeile1)*Pi/180)*zoom;
  end;
end;
// Gnomonische Projektion
15:
begin
 if StrToFloat(zeile2)<=0 then
  begin
   neu:=true;
   continue;
  end;
 if mittey<>0 then
  begin
   winkelmitte:=mittex/180*(-Pi);
   rmitte:=cos(mittey*Pi/180)/sin(mittey*Pi/180);
  end;
 winkel:=(StrToFloat(zeile1))/180*(-Pi);
 r:=cos(StrToFloat(zeile2)*Pi/180)/sin(StrToFloat(zeile2)*Pi/180);
 if CheckKartenmittelpunkt.Checked=true then
  begin
   a:=(r*sin(winkel+Pi))*zoom-(rmitte*sin(winkelmitte+Pi))*zoom;
```

```
         b:=(r*cos(winkel+Pi))*zoom-(rmitte*cos(winkelmitte+Pi))*zoom;
        end
        else
        begin
         a:=(r*sin(winkel+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom;
        end;
      end;
      // Orthographische Projektion
      16:
      begin
       if StrToFloat(zeile2)<0 then continue;
       winkelmitte:=mittex/180*(-Pi);
       rmitte:=cos(mittey*Pi/180);
       winkel:=(StrToFloat(zeile1))/180*(-Pi);
       r:=cos(StrToFloat(zeile2)*Pi/180);
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=(r*sin(winkel+Pi))*zoom-(rmitte*sin(winkelmitte+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom-(rmitte*cos(winkelmitte+Pi))*zoom;
        end
        else
        begin
         a:=(r*sin(winkel+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom;
        end;
      end;
      // Perspektiver Entwurf
      17:
      begin
       if StrToFloat(zeile2)<0 then continue;
       z:=StrToFloat(EdBKreis.Text);
       p:=StrToFloat(EdBKreis2.Text);
       winkelmitte:=mittex/180*(-Pi);
       rmitte:=((z-p)*cos(mittey*Pi/180))/(z-sin(mittey*Pi/180));
       winkel:=(StrToFloat(zeile1))/180*(-Pi);
       r:=((z-p)*cos(StrToFloat(zeile2)*Pi/180))/(z-
sin(StrToFloat(zeile2)*Pi/180));
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=(r*sin(winkel+Pi))*zoom-(rmitte*sin(winkelmitte+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom-(rmitte*cos(winkelmitte+Pi))*zoom;
        end
        else
        begin
         a:=(r*sin(winkel+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom;
        end;
      end;
      // Stereographische Doppelprojektion von SOLOVJEW
```

```
      18:
      begin
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=tan((Pi/2-
StrToFloat(zeile2)*Pi/180)/4)*sin(StrToFloat(zeile1)*Pi/180)*zoom-
            tan((Pi/2-mittey*Pi/180)/4)*sin(mittex*Pi/180)*zoom;
         b:=(-1)*tan((Pi/2-
StrToFloat(zeile2)*Pi/180)/4)*cos(StrToFloat(zeile1)*Pi/180)*zoom-
            (-1)*tan((Pi/2- mittey*Pi/180)/4)*cos(mittex*Pi/180)*zoom;
        end
        else
        begin
         a:=tan((Pi/2-
StrToFloat(zeile2)*Pi/180)/4)*sin(StrToFloat(zeile1)*Pi/180)*zoom;
         b:=(-1)*tan((Pi/2-
StrToFloat(zeile2)*Pi/180)/4)*cos(StrToFloat(zeile1)*Pi/180)*zoom;
        end;
      end;
      // Entwurf von SANSON
      20:
      begin
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=(StrToFloat(zeile1))*cos(StrToFloat(zeile2)*Pi/180)*zoom-
(mittex)*cos(mittey*Pi/180)*zoom;
         b:=(StrToFloat(zeile2)-mittey)*zoom;
        end
        else
        begin
         a:=(StrToFloat(zeile1))*cos(StrToFloat(zeile2)*Pi/180)*zoom;
         b:=(StrToFloat(zeile2))*zoom;
        end;
      end;
      // Entwurf von BONNE
      21:
      begin
       a2:=StrToFloat(EdBKreis.Text)*Pi/180;
       c:=cos(a2)/sin(a2)+a2;
       winkelmitte:=cos(mittey*(-Pi)/180)/(c-mittey*Pi/180)*mittex*(-
Pi)/180;
       rmitte:=c-mittey*Pi/180;
       winkel:=cos(StrToFloat(zeile2)*(-Pi)/180)/(c-
StrToFloat(zeile2)*Pi/180)*StrToFloat(zeile1)*(-Pi)/180;
       r:=c-StrToFloat(zeile2)*Pi/180;
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=(r*sin(winkel+Pi))*zoom-(rmitte*sin(winkelmitte+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom-(rmitte*cos(winkelmitte+Pi))*zoom;
        end
```

```
        else
        begin
         a:=(r*sin(winkel+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom+(90*skalierung);
        end;
      end;
      // Entwurf von STAB-WERNER
      22:
      begin
       a2:=StrToFloat(EdBKreis.Text)*Pi/180;
       c:=Pi/2;
       winkelmitte:=cos(mittey*(-Pi)/180)/(c-mittey*Pi/180)*mittex*(-
Pi)/180;
       rmitte:=c-mittey*Pi/180;
       winkel:=cos(StrToFloat(zeile2)*(-Pi)/180)/(c-
StrToFloat(zeile2)*Pi/180)*StrToFloat(zeile1)*(-Pi)/180;
       r:=c-StrToFloat(zeile2)*Pi/180;
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=(r*sin(winkel+Pi))*zoom-(rmitte*sin(winkelmitte+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom-(rmitte*cos(winkelmitte+Pi))*zoom;
        end
        else
        begin
         a:=(r*sin(winkel+Pi))*zoom;
         b:=(r*cos(winkel+Pi))*zoom+(55*skalierung);
        end;
      end;
      // Entwurf von MOLLWEIDE
      23:
      begin
       // iterative berechnung von t der Gleichung Pi*sin(...)=2t+sin(2t)
       it:=trunc(StrToFloat(EdBKreis.Text)); // # Iterationsschritte
       tlow:=-2*Pi;thigh:=2*Pi;
       for i := 0 to it do
        begin
          t:=(tlow+thigh)/2;
          hilf:=sin(2*t)+2*t-Pi*sin(StrToFloat(zeile2)*Pi/180);
          if hilf<0 then
           begin
            tlow:=t;
           end;
          if hilf>0 then
           begin
            thigh:=t;
           end;
        end;
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=(StrToFloat(zeile1)-mittex)*zoom;
```

```
         b:=(StrToFloat(zeile2)-mittey)*zoom;
        end
        else
        begin
         a:=2*sqrt(2)/Pi*StrToFloat(zeile1)*Pi/180*cos(t)*zoom;
         b:=sqrt(2)*sin(t)*zoom;
        end;
      end;
      // Entwurf von AITOFF
      24:
      begin

hilf:=cos(StrToFloat(zeile1)*Pi/180/2)*cos(StrToFloat(zeile2)*Pi/180);
       hilfm:=cos(mittex*Pi/180/2)*cos(mittey*Pi/180);
       if abs(hilf) < 0.7 then
        begin
         hilf:=ArcTan(hilf/sqrt(1-sqr(hilf)));
        end
        else
        begin
         hilf:=Pi/2-ArcTan(sqrt(1-sqr(hilf))/hilf);
        end;
       if abs(hilfm) < 0.7 then
        begin
         hilfm:=ArcTan(hilfm/sqrt(1-sqr(hilfm)));
        end
        else
        begin
         hilfm:=Pi/2-ArcTan(sqrt(1-sqr(hilfm))/hilfm);
        end;
       hilf2:=sqrt(1-
sqr(cos(StrToFloat(zeile1)*Pi/180/2))*sqr(cos(StrToFloat(zeile2)*Pi/180)));
       hilf2m:=sqrt(1-sqr(cos(mittex*Pi/180/2))*sqr(cos(mittey*Pi/180)));
       if CheckKartenmittelpunkt.Checked=true then
        begin
        if abs(hilf2) > 0.0001 then
          begin
           a:=2*(Pi/2-
hilf)*sin(StrToFloat(zeile1)*Pi/180/2)*cos(StrToFloat(zeile2)*Pi/180)/hilf2
*zoom-
              2*(Pi/2-
hilfm)*sin(mittex*Pi/180/2)*cos(mittey*Pi/180)/hilf2m*zoom;
           b:=(Pi/2-hilf)*sin(StrToFloat(zeile2)*Pi/180)/hilf2*zoom-
              (Pi/2-hilfm)*sin(mittey*Pi/180)/hilf2m*zoom;
          end
          else
          begin
           a:=0-2*(Pi/2-
hilfm)*sin(mittex*Pi/180/2)*cos(mittey*Pi/180)/hilf2m*zoom;
           b:=0-(Pi/2-hilfm)*sin(mittey*Pi/180)/hilf2m*zoom;
```

```
          end;
        end
        else
        begin
         if abs(hilf2) > 0.0001 then
          begin
           a:=2*(Pi/2-
hilf)*sin(StrToFloat(zeile1)*Pi/180/2)*cos(StrToFloat(zeile2)*Pi/180)/hilf2
*zoom;
           b:=(Pi/2-hilf)*sin(StrToFloat(zeile2)*Pi/180)/hilf2*zoom;
          end
          else
          begin
           a:=0;
           b:=0;
          end;
        end;
      end;
      // Entwurf von HAMMER
      25:
      begin
       if CheckKartenmittelpunkt.Checked=true then
        begin

a:=2*sqrt(2)*cos(StrToFloat(zeile2)*Pi/180)*sin(StrToFloat(zeile1)*Pi/180/2
)/sqrt(1+cos(StrToFloat(zeile2)*Pi/180)*cos(StrToFloat(zeile1)*Pi/180/2))*z
oom-

2*sqrt(2)*cos(mittey*Pi/180)*sin(mittex*Pi/180/2)/sqrt(1+cos(mittey*Pi/180)
*cos(mittex*Pi/180/2))*zoom;

b:=sqrt(2)*sin(StrToFloat(zeile2)*Pi/180)/sqrt(1+cos(StrToFloat(zeile2)*Pi/
180)*cos(StrToFloat(zeile1)*Pi/180/2))*zoom-

sqrt(2)*sin(mittey*Pi/180)/sqrt(1+cos(mittey*Pi/180)*cos(mittex*Pi/180/2))*
zoom;
        end
        else
        begin

a:=2*sqrt(2)*cos(StrToFloat(zeile2)*Pi/180)*sin(StrToFloat(zeile1)*Pi/180/2
)/sqrt(1+cos(StrToFloat(zeile2)*Pi/180)*cos(StrToFloat(zeile1)*Pi/180/2))*z
oom;

b:=sqrt(2)*sin(StrToFloat(zeile2)*Pi/180)/sqrt(1+cos(StrToFloat(zeile2)*Pi/
180)*cos(StrToFloat(zeile1)*Pi/180/2))*zoom;
        end;
      end;
      // Entwurf von WINKEL
      26:
```

```
      begin
       par1:=cos(StrToFloat(EdBKreis.Text)*Pi/180);

hilf:=cos(StrToFloat(zeile1)*Pi/180/2)*cos(StrToFloat(zeile2)*Pi/180);
       hilfm:=cos(mittex*Pi/180/2)*cos(mittey*Pi/180);
       if abs(hilf) < 0.7 then
        begin
         hilf:=ArcTan(hilf/sqrt(1-sqr(hilf)));
        end
        else
        begin
         hilf:=Pi/2-ArcTan(sqrt(1-sqr(hilf))/hilf);
        end;
       if abs(hilfm) < 0.7 then
        begin
         hilfm:=ArcTan(hilfm/sqrt(1-sqr(hilfm)));
        end
        else
        begin
         hilfm:=Pi/2-ArcTan(sqrt(1-sqr(hilfm))/hilfm);
        end;
       hilf2:=sqrt(1-
sqr(cos(StrToFloat(zeile1)*Pi/180/2))*sqr(cos(StrToFloat(zeile2)*Pi/180)));
       hilf2m:=sqrt(1-sqr(cos(mittex*Pi/180/2))*sqr(cos(mittey*Pi/180)));
       if CheckKartenmittelpunkt.Checked=true then
        begin
        if abs(hilf2) > 0.0001 then
          begin
           a:=(StrToFloat(zeile1)*Pi/180*par1+2*(Pi/2-
hilf)*sin(StrToFloat(zeile1)*Pi/180/2)*cos(StrToFloat(zeile2)*Pi/180)/hilf2
)/2*zoom-
              (mittex*Pi/180*par1+2*(Pi/2-
hilfm)*sin(mittex*Pi/180/2)*cos(mittey*Pi/180)/hilf2m)/2*zoom;
           b:=(StrToFloat(zeile2)*Pi/180+(Pi/2-
hilf)*sin(StrToFloat(zeile2)*Pi/180)/hilf2)/2*zoom-
              (mittey*Pi/180+(Pi/2-
hilfm)*sin(mittey*Pi/180)/hilf2m)/2*zoom;
          end
          else
          begin
           a:=0-(mittex*Pi/180*par1+2*(Pi/2-
hilfm)*sin(mittex*Pi/180/2)*cos(mittey*Pi/180)/hilf2m)/2*zoom;
           b:=0-(mittey*Pi/180+(Pi/2-
hilfm)*sin(mittey*Pi/180)/hilf2m)/2*zoom;
          end;
        end
        else
        begin
         if abs(hilf2) > 0.0001 then
          begin
```

```
          a:=(StrToFloat(zeile1)*Pi/180*par1+2*(Pi/2-
hilf)*sin(StrToFloat(zeile1)*Pi/180/2)*cos(StrToFloat(zeile2)*Pi/180)/hilf2
)/2*zoom;
          b:=(StrToFloat(zeile2)*Pi/180+(Pi/2-
hilf)*sin(StrToFloat(zeile2)*Pi/180)/hilf2)/2*zoom;
         end
         else
         begin
          a:=0;
          b:=0;
         end;
       end;
     end;
     // Umbezifferter SANSON-Entwurf
     27:
     begin
      par1:=sqrt(3);
      par2:=sqr(par1);
      hilf:=par1*sin(StrToFloat(zeile2)*Pi/180)/2;
      hilfm:=par1*sin(mittey*Pi/180)/2;
      if abs(hilf) < 0.7 then
       begin
        hilf:= ArcTan(hilf/sqrt(1- sqr(hilf)));
       end
       else
       begin
        hilf:=sign(hilf)*Pi/2-ArcTan(sqrt(1-sqr(hilf))/hilf);
       end;
      if abs(hilfm) < 0.7 then
       begin
        hilfm:= ArcTan(hilfm/sqrt(1- sqr(hilfm)));
       end
       else
       begin
        hilfm:=sign(hilfm)*Pi/2-ArcTan(sqrt(1-sqr(hilfm))/hilfm);
       end;
      if CheckKartenmittelpunkt.Checked=true then
       begin
        a:=par2*(2/3)*StrToFloat(zeile1)*Pi/180*sqrt(1-
(3/4)*sin(StrToFloat(zeile2)*Pi/180*sin(StrToFloat(zeile2)*Pi/180)))*zoom-
           par2*(2/3)*mittex*Pi/180*sqrt(1-
(3/4)*sin(mittey*Pi/180*sin(mittey*Pi/180)))*zoom;
        b:=par2*hilf*zoom-
           par2*hilfm*zoom;
       end
       else
       begin
        a:=par2*(2/3)*StrToFloat(zeile1)*Pi/180*sqrt(1-
(3/4)*sin(StrToFloat(zeile2)*Pi/180*sin(StrToFloat(zeile2)*Pi/180)))*zoom;
        b:=par2*hilf*zoom;
```

```
        end;
      end;
      // Entwurf von APIANUS
      29:
      begin
       if CheckKartenmittelpunkt.Checked=true then
        begin
         if abs(StrToFloat(zeile1)*Pi/180) > 0.001 then
          begin

hilf:=sqrt(abs(sqr(sqr(Pi))+8*sqr(Pi)*sqr(StrToFloat(zeile1)*Pi/180)+16*sqr
(StrToFloat(zeile1)*Pi/180)*(sqr(StrToFloat(zeile1)*Pi/180)-
4*sqr(StrToFloat(zeile2)*Pi/180))));

hilfm:=sqrt(abs(sqr(sqr(Pi))+8*sqr(Pi)*sqr(mittex*Pi/180)+16*sqr(mittex*Pi/
180)*(sqr(mittex*Pi/180)-4*sqr(mittey*Pi/180))));
           a:=(((-
1)*sqr(Pi)+4*sqr(StrToFloat(zeile1)*Pi/180)+hilf)/(8*StrToFloat(zeile1)*Pi/
180))*zoom-
                (((-
1)*sqr(Pi)+4*sqr(mittex*Pi/180)+hilfm)/(8*mittex*Pi/180))*zoom;
          end
          else
          begin
           a:=0-(((-
1)*sqr(Pi)+4*sqr(mittex*Pi/180)+hilfm)/(8*mittex*Pi/180))*zoom;
          end;
         b:=StrToFloat(zeile2)*Pi/180*zoom-
            mittey*Pi/180*zoom;
        end
        else
        begin
         if abs(StrToFloat(zeile1)*Pi/180) > 0.001 then
          begin

hilf:=sqrt(abs(sqr(sqr(Pi))+8*sqr(Pi)*sqr(StrToFloat(zeile1)*Pi/180)+16*sqr
(StrToFloat(zeile1)*Pi/180)*(sqr(StrToFloat(zeile1)*Pi/180)-
4*sqr(StrToFloat(zeile2)*Pi/180))));
           a:=(((-
1)*sqr(Pi)+4*sqr(StrToFloat(zeile1)*Pi/180)+hilf)/(8*StrToFloat(zeile1)*Pi/
180))*zoom;
          end
          else
          begin
           a:=0;
          end;
         b:=StrToFloat(zeile2)*Pi/180*zoom;
        end;
      end;
      // Entwurf 2 von APIANUS
```

```
      30:
      begin
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=StrToFloat(zeile1)*Pi/180/Pi*sqrt(sqr(Pi)-
4*sqr(StrToFloat(zeile2)*Pi/180))*zoom-
            mittex*Pi/180/Pi*sqrt(sqr(Pi)-4*sqr(mittey*Pi/180))*zoom;
         b:=StrToFloat(zeile2)*Pi/180*zoom-
            mittey*Pi/180*zoom;
        end
        else
        begin
         a:=StrToFloat(zeile1)*Pi/180/Pi*sqrt(sqr(Pi)-
4*sqr(StrToFloat(zeile2)*Pi/180))*zoom;
         b:=StrToFloat(zeile2)*Pi/180*zoom;
        end;
      end;
      // Trapezentwurf von ECKERT
      31:
      begin
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=StrToFloat(zeile1)*(1-abs(StrToFloat(zeile2))/180)*zoom-
            mittex*(1-abs(mittey)/180)*zoom;
         b:=StrToFloat(zeile2)*zoom-mittey*zoom;
        end
        else
        begin
         a:=StrToFloat(zeile1)*(1-abs(StrToFloat(zeile2))/180)*zoom;
         b:=StrToFloat(zeile2)*zoom;
        end;
      end;
      // Entwurf 3 von ECKERT
      32:
      begin
       if CheckKartenmittelpunkt.Checked=true then
        begin

a:=StrToFloat(zeile1)*Pi/180*sqrt(4*Pi/(4+Pi))*(1+cos(StrToFloat(zeile2)*Pi
/180))/Pi*zoom-
            mittex*Pi/180*sqrt(4*Pi/(4+Pi))*(1+cos(mittey*Pi/180))/Pi*zoom;
         b:=sqrt(4*Pi/(4+Pi))*sin(StrToFloat(zeile2)*Pi/180)*zoom-
            sqrt(4*Pi/(4+Pi))*sin(mittey*Pi/180)*zoom;
        end
        else
        begin

a:=StrToFloat(zeile1)*Pi/180*sqrt(4*Pi/(4+Pi))*(1+cos(StrToFloat(zeile2)*Pi
/180))/Pi*zoom;
         b:=sqrt(4*Pi/(4+Pi))*sin(StrToFloat(zeile2)*Pi/180)*zoom;
```

```
        end;
      end;
      // Entwurf 5 von ECKERT
      33:
      begin
       if CheckKartenmittelpunkt.Checked=true then
        begin

a:=1/sqrt(Pi+2)*StrToFloat(zeile1)*Pi/180*(1+cos(StrToFloat(zeile2)*Pi/180)
)*zoom-1/sqrt(Pi+2)*mittex*Pi/180*(1+cos(mittey*Pi/180))*zoom;
         b:=2/sqrt(Pi+2)*StrToFloat(zeile2)*Pi/180*zoom-
2/sqrt(Pi+2)*mittey*Pi/180*zoom;
        end
        else
        begin

a:=1/sqrt(Pi+2)*StrToFloat(zeile1)*Pi/180*(1+cos(StrToFloat(zeile2)*Pi/180)
)*zoom;
         b:=2/sqrt(Pi+2)*StrToFloat(zeile2)*Pi/180*zoom;
        end;
      end;
      // Entwurf von KAWRASKY
      35:
      begin
       par1:=1.455;
       par2:=0.75;
       par3:=par1*par2;
       par4:=89.5; // GRENZE

       if CheckKartenmittelpunkt.Checked=true then
        begin
         if abs(StrToFloat(zeile2))<par4 then
          begin

a:=StrToFloat(zeile1)*Pi/180/par3*cos(StrToFloat(zeile2)*Pi/180)/cos(par2*S
trToFloat(zeile2)*Pi/180)*zoom-

mittex*Pi/180/par3*cos(mittey*Pi/180)/cos(par2*mittey*Pi/180)*zoom;
          end
          else
          begin
           a:=(-
1)*mittex*Pi/180/par3*cos(mittey*Pi/180)/cos(par2*mittey*Pi/180)*zoom;
          end;
         b:=par1*sin(par2*StrToFloat(zeile2)*Pi/180)*zoom-
            par1*sin(par2*mittey*Pi/180)*zoom;
        end
        else
        begin
         if abs(StrToFloat(zeile2))<par4 then
```

```
          begin

a:=StrToFloat(zeile1)*Pi/180/par3*cos(StrToFloat(zeile2)*Pi/180)/cos(par2*S
trToFloat(zeile2)*Pi/180)*zoom;
          end
          else
          begin
           a:=0;
          end;
         b:=par1*sin(par2*StrToFloat(zeile2)*Pi/180)*zoom;
        end;
      end;
      // Entwurf von PREPETIT-FOUCAULT
      36:
      begin
       if CheckKartenmittelpunkt.Checked=true then
        begin

a:=StrToFloat(zeile1)*Pi/180*cos(StrToFloat(zeile2)*Pi/180)*sqr(cos(StrToFl
oat(zeile2)*Pi/180))*zoom-
            mittex*Pi/180*cos(mittey*Pi/180)*sqr(cos(mittey*Pi/180))*zoom;

b:=2*sin(StrToFloat(zeile2)*Pi/180/2)/cos(StrToFloat(zeile2)*Pi/180/2)*zoom
-
            2*sin(mittey*Pi/180/2)/cos(mittey*Pi/180/2)*zoom;
        end
        else
        begin

a:=StrToFloat(zeile1)*Pi/180*cos(StrToFloat(zeile2)*Pi/180)*sqr(cos(StrToFl
oat(zeile2)*Pi/180))*zoom;

b:=2*sin(StrToFloat(zeile2)*Pi/180/2)/cos(StrToFloat(zeile2)*Pi/180/2)*zoom
;
        end;
      end;
      // Entwurf von COLLIGNON
      37:
      begin
       if CheckKartenmittelpunkt.Checked=true then
        begin
         hilf:=sqrt(Pi)*(1-sqrt(2)*sin(Pi/4-
abs(StrToFloat(zeile2)*Pi/180)/2));
         hilf2:=sqrt(Pi)*(1-sqrt(2)*sin(Pi/4-abs(mittey*Pi/180)/2));
         a:=StrToFloat(zeile1)/180*(sqrt(Pi)-hilf)*zoom-
            mittex/180*(sqrt(Pi)-hilf2)*zoom;
         b:=hilf*sign(StrToFloat(zeile2))*zoom-
            hilf2*sign(mittey)*zoom;
        end
        else
```

```
        begin
         hilf:=sqrt(Pi)*(1-sqrt(2)*sin(Pi/4-
abs(StrToFloat(zeile2)*Pi/180)/2));
         a:=StrToFloat(zeile1)/180*(sqrt(Pi)-hilf)*zoom;
         b:=hilf*sign(StrToFloat(zeile2))*zoom;
        end;
      end;
      // Entwurf von WIECHEL
      38:
      begin
       hilf:=pi/4-(StrToFloat(zeile2)*pi/180)/2;
       hilfm:=pi/4-(mittey*pi/180)/2;
       if CheckKartenmittelpunkt.Checked=true then
        begin
         a:=sin(hilf)*sin(StrToFloat(zeile1)*pi/180+hilf)*zoom-
            sin(hilfm)*sin(mittex*pi/180+hilfm)*zoom;
         b:=(-1)*sin(hilf)*cos(StrToFloat(zeile1)*pi/180+hilf)*zoom-
            (-1)*sin(hilfm)*cos(mittex*pi/180+hilfm)*zoom;
        end
        else
        begin
         a:=sin(hilf)*sin(StrToFloat(zeile1)*pi/180+hilf)*zoom;
         b:=(-1)*sin(hilf)*cos(StrToFloat(zeile1)*pi/180+hilf)*zoom;
        end;
      end;

     end;

     // Runden der Koordinaten für Bildschirmanzeige
     aa:=round(a);
     bb:=round(b);

     // Erstellen der Karte - Bildschirm und BMP gleichzeitig !!!
     if neu=true then
      begin
       FormularZeichnung.canvas.moveto(x+aa,y-bb);
       karte.Canvas.MoveTo(x+aa,y-bb);
       neu:=false;
      end
      else
      begin
       FormularZeichnung.canvas.LineTo(x+aa,y-bb);
       karte.Canvas.LineTo(x+aa,y-bb);
      end;

    end;

   closefile(datei);

   // SPEICHERN DES BITMAPS
```

```
   karte.SaveToFile(bmpoutputfile);
   Karte.Free;

  end
  else
  begin
   // FEHLERMELDUNGEN
   LbErrorKartoTXT.Caption:='  ';
   if filename='' then
LbErrorKartoTXT.Caption:=LbErrorKartoTXT.Caption+'keine KartoTXT gewählt!
';
   if not auswahl then
LbErrorKartoTXT.Caption:=LbErrorKartoTXT.Caption+'keine Projektion gewählt!
';
   if bmpoutputfile='' then
LbErrorKartoTXT.Caption:=LbErrorKartoTXT.Caption+'keine BMP-Datei gewählt!
';
   LbErrorKartoTXT.Caption:=LbErrorKartoTXT.Caption+' ';
   LbErrorKartoTXT.Visible:=True;
  end;

end;

// ÄNDERUNGEN DER PROJEKTIONSAUSWAHL - Ansichts-Parameter werden
vorgeschlagen
procedure THauptformular.CBProjektionsauswahlChange(Sender: TObject);
begin
 auswahl:=true;
 LbErrorKartoTXT.Visible:=false;
 LbBkreis.Visible:=false;
 LbBkreis2.Visible:=false;
 EdBkreis.Visible:=false;
 EdBKreis2.Visible:=false;
 case CBProjektionsauswahl.ItemIndex of
  // Leere Auswahl
  3,8,11,19,28,34:
  begin
   auswahl:=false; // keine sinnvolle Auswahl '***'
  end;
  0:
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='3';
  end;
  1:
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='3';
  end;
  2:
```

```
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='3';
end;
4:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='120';
 EdBKreis.Text:='22,5';
 EdBkreis.Visible:=true;
 LbBkreis.Visible:=true;
 LbBkreis.Caption:='Berührkreis:';
end;
5:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='120';
 EdBkreis.Text:='50';
 EdBKreis2.Text:='90';
 LbBKreis2.Visible:=true;
 LbbKreis.Visible:=true;
 EdBkreis.Visible:=true;
 EdBkreis2.Visible:=true;
 LbBkreis.Caption:='1. Berührkreis:';
 LbBKreis2.Caption:='2. Berührkreis';
end;
6:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='120';
 EdBkreis.Text:='50';
 LbbKreis.Visible:=true;
 EdBkreis.Visible:=true;
 LbBkreis.Caption:='Berührkreis:'
end;
7:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='120';
 EdBkreis.Text:='50';
 LbbKreis.Visible:=true;
 EdBkreis.Visible:=true;
 LbBkreis.Caption:='Berührkreis:'
end;
9:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='100';
end;
10:
```

```
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='120';
end;
12:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='80';
end;
13:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='130';
end;
14:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='50';
end;
15:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='100';
end;
16:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='270';
end;
17:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='80';
 EdBkreis.Text:='1,5';
 EdBKreis2.Text:='5';
 LbBKreis2.Visible:=true;
 LbbKreis.Visible:=true;
 EdBkreis.Visible:=true;
 EdBkreis2.Visible:=true;
 LbBkreis.Caption:='Projektionszentrum (0,0,z):';
 LbBKreis2.Caption:='Bildebene (p)';
end;
18:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='250';
end;
20:
begin
 EdSkalierung.Text:='3';
```

```
 EdZoom.Text:='3';
end;
21:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='140';
 EdBKreis.Text:='30';
 EdBkreis.Visible:=true;
 LbBkreis.Visible:=true;
 LbBkreis.Caption:='isom. Breitenkreis:';
end;
22:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='110';
 EdBKreis.Text:='30';
 EdBkreis.Visible:=true;
 LbBkreis.Visible:=true;
 LbBkreis.Caption:='isom. Breitenkreis:';
end;
23:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='150';
 EdBKreis.Text:='10';
 EdBkreis.Visible:=true;
 LbBkreis.Visible:=true;
 LbBkreis.Caption:='# Iterationsschritte:';
end;
24:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='150';
end;
25:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='180';
end;
26:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='150';
end;
27:
begin
 EdSkalierung.Text:='3';
 EdZoom.Text:='80';
end;
29:
```

```
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='150';
  end;
  30:
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='150';
  end;
  31:
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='3';
  end;
  32:
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='180';
  end;
  33:
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='180';
  end;
  35:
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='180';
  end;
  36:
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='140';
  end;
  37:
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='150';
  end;
  38:
  begin
   EdSkalierung.Text:='3';
   EdZoom.Text:='250';
  end;

 end;
end;

// Auswahl ob ein individueller Kartenmittelpunkt gewählt wird
procedure THauptformular.CheckkartenmittelpunktClick(Sender: TObject);
```

```
begin
 if Checkkartenmittelpunkt.Checked=false then
  begin
   LbMittex.Visible:=false;
   LbMittey.Visible:=false;
   EdKartenmittelpunktx.Visible:=false;
   EdKartenmittelpunkty.Visible:=false;
  end
  else
  begin
   LbMittex.Visible:=true;
   LbMittey.Visible:=true;
   EdKartenmittelpunktx.Visible:=true;
   EdKartenmittelpunkty.Visible:=true;
  end;
end;

// HAUPTMENU - Tool zur Gradnetzerstellung
procedure THauptformular.GradnetzerstellenClick(Sender: TObject);
begin
 FormularGradnetz.LbError.Visible:=false;
 FormularGradnetz.show;
end;

// HAUPTMENU - Auswahl der KartoTXT-Datei
procedure THauptformular.KartoTXTwhlen1Click(Sender: TObject);
begin
 OpenDialogKartoTXT.execute;
 filename:=OpenDialogKartoTXT.FileName;
 LbKartoTXT.Caption:='KartoTXT-Datei: ' + filename;
 LbErrorKartoTXT.Visible:=false;
end;

// HAUPTMENU - Tool zur Erstellung einer KartoTXT-Datei
procedure THauptformular.KMLKartoTXT1Click(Sender: TObject);
begin
 FormularKonvertiere.LbError.Visible:=False;
 FormularKonvertiere.show;
end;

// HAUTPMENU - Tool zur Kombination 2er Karto-TXT-Dateien
procedure THauptformular.N2KartoTXTkombinieren1Click(Sender: TObject);
begin
 FormularKombiniere.LbError.Visible:=false;
 FormularKombiniere.Show;
end;

end.
```

4.5.2 Formular Konvertiere XML > KartoTXT

```
unit formular_konvert;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;

type
  TFormularKonvertiere = class(TForm)
    OpenDialogKML: TOpenDialog;
    BtKML: TButton;
    LbKML: TLabel;
    LbINPUT: TLabel;
    LbOUTPUT: TLabel;
    BtKartoTXT: TButton;
    LbKartoTXTneu: TLabel;
    EdKartoTXT: TEdit;
    LbKartoTXT: TLabel;
    BtKonvertiere: TButton;
    BtAbbrechen: TButton;
    SaveDialogKartoTXT: TSaveDialog;
    LbError: TLabel;
    procedure BtKonvertiereClick(Sender: TObject);
    procedure BtAbbrechenClick(Sender: TObject);
    procedure EdKartoTXTChange(Sender: TObject);
    procedure BtKartoTXTClick(Sender: TObject);
    procedure BtKMLClick(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  FormularKonvertiere: TFormularKonvertiere;
  inputfile, outputfile: string;

implementation

{$R *.dfm}

procedure TFormularKonvertiere.BtAbbrechenClick(Sender: TObject);
begin
 inputfile:='';outputfile:='';
```

```
 LbKML.Caption:='KML-Datei:';
 LbKartoTXT.Caption:='KartoTXT-Datei:';
 close();
end;

procedure TFormularKonvertiere.BtKartoTXTClick(Sender: TObject);
begin
 LbError.Visible:=False;
 SaveDialogKartoTXT.execute;
 LbKartoTXT.Caption:='KartoTXT-Datei: ' + SaveDialogKartoTXT.FileName;
 outputfile:=SaveDialogKartoTXT.FileName;
end;

procedure TFormularKonvertiere.BtKMLClick(Sender: TObject);
begin
 LbError.Visible:=False;
 OpenDialogKML.execute;
 LbKML.Caption:='KML-Datei: ' + OpenDialogKML.FileName;
 inputfile:=OpenDialogKML.FileName;
end;

procedure TFormularKonvertiere.BtKonvertiereClick(Sender: TObject);
var
 datei, datei2: textfile;
 wort, wort2, wort3, zwischen1, zwischen2: string;
 text, text2, text3: char;
 a, i, k: integer;
 neu: boolean;
begin
LbError.Visible:=False;
AssignFile(datei,'c:/karto/parameter.txt');
Reset(datei);
Readln(datei,zwischen1);
Readln(datei,zwischen2);
Closefile(datei);
if (inputfile<>'') and (outputfile<>'') then
begin
 AssignFile(datei,inputfile);
 Reset(datei);
 i:=0;k:=0;
 Assignfile(datei2,zwischen1);
 rewrite(datei2);closefile(datei2);
 while not eof(datei) do
    begin
     Read(datei,text);
     if text='c' then
      begin
       wort:=text;
       for a := 0 to 10 do
        begin
```

```
          read(datei,text2);
          wort:=wort + text2;
        end;
      end;
     if wort='coordinates>' then
      begin
       k:=k+1;
       append(datei2);
       writeln(datei2,'NEU');
       closefile(datei2);
       repeat
         wort:='';
         repeat

            read(datei,text3);
            wort:=wort+text3;
          until (text3=',') or (text3='<') or (text3=' ');
         if length(wort)<>1 then
         begin
          append(datei2);
          writeln(datei2,Copy(wort,1,length(wort)-1));
          closefile(datei2);
         end;
       until text3='<';
      end;
     i:=i+1;
    end;
    // jetzt kommt der 2. schritt
Assignfile(datei,zwischen1);
Assignfile(datei2,zwischen2);
rewrite(datei2);closefile(datei2);
reset(datei);
neu:=false;
while not eof(datei) do
begin
  readln(datei,wort);

  if (wort<>'NEU') or (neu=false) then
   begin

   append(datei2);
   writeln(datei2,wort);
   closefile(datei2);

   if wort='NEU' then neu:=true else neu:=false;

   end;

end;
closefile(datei);
```

```
 // jetzt kommt der 3.schritt
 Assignfile(datei,zwischen2);
 Assignfile(datei2,outputfile);
 rewrite(datei2);closefile(datei2);
 reset(datei);
 neu:=false;
 while not eof(datei) do
 begin
  readln(datei,wort);

  if wort='NEU' then
   begin
    append(datei2);
    writeln(datei2,wort);
    closefile(datei2);
    readln(datei,wort);
    readln(datei,wort2);
    readln(datei,wort3);
   end
   else
   begin
    readln(datei,wort2);
    readln(datei,wort3);
   end;

   append(datei2);
   writeln(datei2,wort);
   writeln(datei2,wort2);
   closefile(datei2);

 end;
 closefile(datei);
 inputfile:='';
 outputfile:='';
 close();

end
else
begin
  LbError.Visible:=true;
end;
end;

procedure TFormularKonvertiere.EdKartoTXTChange(Sender: TObject);
begin
 LbKartoTXT.Caption:='KartoTXT-Datei: ' +EdKartoTXT.Text;
end;

end.
```

4.5.3 Formular Erstelle Gradnetz

```
unit formular_gradnetz;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;

type
  TFormularGradnetz = class(TForm)
    EdGradabstand: TEdit;
    BtGradnetzErstellen: TButton;
    LbGradabstand: TLabel;
    SaveDialogGradnetz: TSaveDialog;
    BtAbbrechen: TButton;
    EdAufloesung: TEdit;
    LbAufloesung: TLabel;
    BtKartoTXT: TButton;
    LbKartoTXT: TLabel;
    BtKartoTXTRand: TButton;
    SaveDialogKartoTXTRand: TSaveDialog;
    LbError: TLabel;
    procedure BtKartoTXTRandClick(Sender: TObject);
    procedure BtKartoTXTClick(Sender: TObject);
    procedure BtGradnetzErstellenClick(Sender: TObject);
    procedure BtAbbrechenClick(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  FormularGradnetz: TFormularGradnetz;
  outputfile: string;

implementation

{$R *.dfm}

procedure TFormularGradnetz.BtAbbrechenClick(Sender: TObject);
begin
 LbError.Visible:=false;
 close()
end;
```

```
procedure TFormularGradnetz.BtGradnetzErstellenClick(Sender: TObject);
var
 datei: textfile;
 abstand, counter, i, k: integer;
 aufloesung: real;

begin
 if (outputfile<>'') and (EdGradabstand.Text<>'') then
  begin
   LbError.Visible:=false;
   AssignFile(datei,outputfile);
   rewrite(datei);
   abstand:=StrToInt(EdGradabstand.Text);
   counter:=round(180/abstand);
   aufloesung:=StrToFloat(EdAufloesung.Text);
   // Längengrade werden geschrieben
   for i := -counter to counter do
    begin
     writeln(datei,'NEU');
     for k := round(-90/aufloesung) to round(90/aufloesung) do
      begin
       writeln(datei,FloatToStr(i*Abstand));
       writeln(datei,FloatToStr(k*aufloesung));
      end;
    end;
    // Breitengrade werden geschrieben
    for i := round(-counter/2) to round(counter/2) do
    begin
     writeln(datei,'NEU');
     for k := round(-180/aufloesung) to round(180/aufloesung) do
      begin
       writeln(datei,FloatToStr(k*aufloesung));
       writeln(datei,FloatToStr(i*abstand));
      end;
    end;
   closefile(datei);
   close();
  end
  else
  begin
   LbError.Visible:=true;
   if (outputfile='') and (EdGradAbstand.Text='') then
    begin
     LbError.Caption:=' Keine Karto-TXT Datei gewählt! Kein Gradabstand
eingegeben!  ';
    end;
   if (outputfile='') and (EdGradAbstand.Text<>'') then
    begin
     LbError.Caption:=' Keine Karto-TXT Datei gewählt!  ';
```

```
    end;
   if (outputfile<>'') and (EdGradAbstand.Text='') then
    begin
     LbError.Caption:=' Kein Gradabstand eingegeben!  ';
    end;
  end;
end;

procedure TFormularGradnetz.BtKartoTXTClick(Sender: TObject);
begin
 LbError.Visible:=false;
 SaveDialogGradnetz.Execute;
 outputfile:=SaveDialogGradnetz.FileName;
 LbKartoTXT.Caption:='KartoTXT-Datei: ' + outputfile;
end;

procedure TFormularGradnetz.BtKartoTXTRandClick(Sender: TObject);
var
 datei: textfile;
 dateiname: string;
 aufloesung: real;
 fix: real;
 k: integer;

begin
 LbError.Visible:=false;
 if SaveDialogKartoTXTRand.Execute then
  begin
   dateiname:= SaveDialogKartoTXTRand.FileName;
   AssignFile(datei,dateiname);
   aufloesung:=StrToFloat(EdAufloesung.Text);
   rewrite(datei);
   // Längengrade werden geschrieben
   writeln(datei,'NEU');
   fix:=-180;
   for k := round(-90/aufloesung) to round(90/aufloesung) do
   begin
    writeln(datei,FloatToStr(fix));
    writeln(datei,FloatToStr(k*aufloesung));
   end;
   writeln(datei,'NEU');
   fix:=180;
   for k := round(-90/aufloesung) to round(90/aufloesung) do
    begin
     writeln(datei,FloatToStr(fix));
     writeln(datei,FloatToStr(k*aufloesung));
    end;
   // Breitengrade werden geschrieben
   writeln(datei,'NEU');
   fix:=-90;
```

```
   for k := round(-180/aufloesung) to round(180/aufloesung) do
    begin
     writeln(datei,FloatToStr(k*aufloesung));
     writeln(datei,FloatToStr(fix));
    end;
   writeln(datei,'NEU');
   fix:=90;
   for k := round(-180/aufloesung) to round(180/aufloesung) do
    begin
     writeln(datei,FloatToStr(k*aufloesung));
     writeln(datei,FloatToStr(fix));
    end;
  closefile(datei);
  close();
 end;

end;

end.
```

4.5.4 Formular Kombiniere 2 KartoTXT-Dateien

```
unit formular_kombi;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;

type
  TFormularKombiniere = class(TForm)
    OpenDialogKartoTXT1: TOpenDialog;
    OpenDialogKartoTXT2: TOpenDialog;
    SaveDialogKartoTXT: TSaveDialog;
    BtKartoTXT1: TButton;
    BtKartoTXT2: TButton;
    BtKartoTXT: TButton;
    BtKombiniere: TButton;
    BtAbbrechen: TButton;
    EdKartoTXT1: TEdit;
    EdKartoTXT2: TEdit;
    EdKartoTXT: TEdit;
    LbKartoTXT1: TLabel;
    LbKartoTXT2: TLabel;
    LbKartoTXT: TLabel;
    LbError: TLabel;
    procedure BtKartoTXTClick(Sender: TObject);
    procedure BtKartoTXT2Click(Sender: TObject);
    procedure BtKartoTXT1Click(Sender: TObject);
    procedure BtKombiniereClick(Sender: TObject);
    procedure BtAbbrechenClick(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  FormularKombiniere: TFormularKombiniere;
  inputfile1, inputfile2, outputfile: string;

implementation

{$R *.dfm}

procedure TFormularKombiniere.BtAbbrechenClick(Sender: TObject);
```

```
begin
 close();
end;

procedure TFormularKombiniere.BtKartoTXT1Click(Sender: TObject);
begin
 LbError.Visible:=false;
 OpenDialogKartoTXT1.Execute;
 inputfile1:=OpenDialogKartoTXT1.FileName;
 EdKartoTXT1.Text:=inputfile1;
end;

procedure TFormularKombiniere.BtKartoTXT2Click(Sender: TObject);
begin
 LbError.Visible:=false;
 OpenDialogKartoTXT2.Execute;
 inputfile2:=OpenDialogKartoTXT2.FileName;
 EdKartoTXT2.Text:=inputfile2;
end;

procedure TFormularKombiniere.BtKartoTXTClick(Sender: TObject);
begin
 LbError.Visible:=false;
 SaveDialogKartoTXT.Execute;
 outputfile:=SaveDialogKartoTXT.FileName;
 EdKartoTXT.Text:=outputfile;
end;

procedure TFormularKombiniere.BtKombiniereClick(Sender: TObject);
var
 datei1,datei2,datei3: textfile;
 zeile: string;

begin
 LbError.Visible:=false;
 if (inputfile1<>'') and (inputfile2<>'') and (outputfile<>'') then
  begin

   Assignfile(datei1,inputfile1);
   Assignfile(datei2,inputfile2);
   Assignfile(datei3,outputfile);
   reset(datei1);
   reset(datei2);
   rewrite(datei3);
   while not eof(datei1) do
    begin
      readln(datei1,zeile);
      writeln(datei3,zeile);
    end;
   while not eof(datei2) do
```

```
    begin
     readln(datei2,zeile);
     writeln(datei3,zeile);
    end;
   closefile(datei1);
   closefile(datei2);
   closefile(datei3);
   close();
  end
  else
  begin
   LbError.Visible:=true;

  end;
end;
end.
```

5 Anhang

5.1 Special Thanks

Zuerst möchte ich mich für die umfangreiche Betreung bei Herrn Prof. Hans Havlicek bedanken, der mir bei allen Anliegen und Fragen fachlich weitergeholfen und dabei auch auf neue Ideen gebracht hat.
Durch ihn wurde mein Interesse an Visualisierung und Kartenentwurfslehre geweckt, was mich auch in den letzten Monaten meines Studiums motiviert hat, verschiedene Lehrveranstaltungen über Geometrie und ähnliche Inhalte zu besuchen.

Ein weiterer Dank gilt Herrn Prof. Hans-Peter Paukowitsch (Institut für Diskrete Mathematik und Geometrie - TU Wien), dessen Lehrveranstaltungen über Teile des Arbeitsgebietes meiner Diplomarbeit mir weitere Einblicke über Geometrie und deren Anwendungen gegeben haben.

Weiters möchte ich mich bei Herrn Dipl. Ing. Markus Jobst (Wissenschaftlicher Mitarbeiter am Institut für Geoinformation und Kartographie - TU Wien) bedanken. Bei der Entwicklung meiner Software **KARTO** hat mir sein Wissen über aktuelle Anwendungen computergestützter Kartenvisualisierung sehr weitergeholfen. Durch seine Empfehlung zur Verwendung von KML-Dateien, ist eine zukunftsorientiere Verwendung von **KARTO** gesichert.

5.2 Persönliche Anmerkungen

Da ich mich weiterhin mit dem Thema Kartenentwürfe beschäftigen will, ist es mir auch ein persönliches Anliegen, die Software **KARTO** weiterzuentwickeln, um noch diverse Ideen und Tools zu integrieren. Anregungen dazu gibt es viele.

Weiters sei anzumerken, dass die Liste der möglichen Kartenentwürfe in **KARTO** bei weitem(!) nicht vollständig, sondern vielmehr nur ein Einblick in die Vielfältigkeit solcher Karten ist.

Die Welt ist keine Scheibe! ... Darum ist das ganze Thema so interessant !

5.3 Literaturverzeichnis

HOSCHEK J.: *Mathematische Grundlagen der Kartographie*
2. Auflage, BI-Wissenschaftsverlag, Mannheim 1984

WAGNER K.: *Kartographische Netzentwürfe*
2. Auflage, Mannheim: Bibliographisches Institut 1962

HENNEMANN K.: *Kartographie und GIS - Eine Einführung*
Wissenschaftliche Buchgesellschaft, Darmstadt 2006

SCHNEIDER U.: *Die Macht der Karten -*
Die Entwicklung der Kartographie vom Mittelalter bis heute
Sonderausgabe (2. überarbeitete Ausgabe), Primus Verlag, Darmstadt 2006

KOHLSTOCK P.: *Kartographie - Eine Einführung*
Schöningh Verlag, Reihe: UTB-Wissenschaften Uni-Taschenbücher, Paderborn 2004

VONHOEGEN H.: *Einstieg in XML*
4. aktualisierte Auflage, Galileo Press, Bonn 2007

WARKEN E.: *Delphi 2006*
Eddison Wesley Verlag, München 2006

DO CARMO M.: Differentialgeometrie von Kurven und Flächen
3. Auflage, Vieweg Verlag, Braunschweig 1998