

Höhenkarte und Hintergrundbild für Transport Fever (2)

0. Vorrede

Nachdem ich länger an der Mainz-Wiesbaden-Karte herumgedoktort und bevor ich wieder alles vergessen habe, möchte ich die Erfahrungen dabei aufschreiben. Insbesondere geht es auch um die Fallstricke auf dem Weg: Wer möchte ernsthaft, daß die ganze Karte stufig ist wie von Reisterrassen übersät ist? Wer möchte entweder neue Berge und Täler schaffen oder die Karte 300 m versetzt aufbauen, nur weil der Bach über den Bergrücken statt im Tal verläuft?

In den wenigsten Fällen wird man den gewünschten Landschaftsausschnitt auf wenige Meter genau finden. Deshalb sollte man sich zunächst überlegen, welcher Karte man den Vorrang geben will: Der Höhenkarte (neudeutsch: Heightmap) oder dem Hintergrundbild? Im allgemeinen erhält man geographisch zuverlässige Koordinaten eher mit der Heightmap; auf dieser läßt sich aber schwerer der gewünschte Ausschnitt finden.

Benötigte Programme:

- Graphikprogramm; im Text ist GIMP beschrieben, es läßt sich sicher viel auch mit anderen guten Graphikprogrammen erreichen. DDS-Export muß möglich sein; bei GIMP funktioniert das bei hinreichend alten Versionen mit einem nicht mehr gepflegten Plugin, oder ab Version 2.10 in der Standardversion. Ebenso muß das Speichern von 16-bit .png-Dateien möglich sein.
- MICRODEM eventuell, je nach Höhenkarte
- Hugin (nicht Munin, das ist für etwas völlig anderes!) oder ein anderes Programm, mit dem man sich überlappende Bilder, die nicht 100% passen, aneinanderfügen kann. Oft unter dem Stichwort Panoramabild zu finden.
- Transport Fever 2 – wer hätte es gedacht?

1. Höhenkarte

1.1. Allgemeines

Der zentrale Punkt der Höhenkarte ist, daß sie ein 16 bit Graustufenbild sein muß. Einmal in 8 bit konvertiert, und man hat die erwähnten Reisterrassen. Das läßt sich kaum rückgängig machen, denn Glätten führt nur zu einer Miniversion der Himmelsleiter südlich von Aachen. Also: Vom Original immer eine Sicherungskopie behalten, auf die man zurückgehen kann.

Für die Arbeit mit Höhendaten mit ernsthaftem Anspruch sollte man ein paar Begriffe kennen:

DHM: Digitales Höhenmodell (englisch: DEM, digital elevation model) ist meistens ein Oberbegriff der folgenden beiden

DOM: Digitales Oberflächenmodell (englisch: DSM, digital surface model) bildet die Oberfläche ab, wie sie meistens vom Vermessungsflugzeug oder -satelliten gesehen wird, nämlich mit Häusern und Bäumen.

DGM: Digitales Geländemodell (englisch: DTM, digital terrain model) ist ohne Gebäude und Vegetation.

Es gibt wohl gerade in der deutschen Literatur Unstimmigkeiten, etwas mehr über das Thema steht in der Wikipedia unter *Digitales Höhenmodell*. Ideal für uns ist natürlich ein DGM, aber mit einem guten DOM kann man auch leben.

UTM: Universal Transverse Mercator, bzw. Gauß-Krüger-Projektion: Zwei Methoden, wie man die Oberfläche einer Kugel lokal in die Ebene abbilden kann. Im Grunde wird die Erdoberfläche von Pol zu Pol in Streifen geteilt, in denen die Verzerrung durch die sphärische Geometrie hinreichend klein sind.

Dann spielt die horizontale Auflösung eine Rolle: Transport Fever (2) hat für die Höhenkarte 4×4 m² große Pixel. Eine höhere Auflösung des Kartenmaterials bringt deshalb nichts – wer für die Daten bezahlen will, kann also hier schauen, ob er nicht zu viel bestellt. Mit einer oft frei erhältlichen Auflösung von 25 m kann man auch einiges anfangen: Im normalen Gelände ändert sich die Steigung auf dieser Länge nur selten dramatisch. Bahn- oder Straßendämme werden damit aber nicht abgebildet.

1.2. Beschaffen

Der erste Hinweis, den ich gefunden habe, war der zu terrain.party in einem Tutorial:

<https://www.plejadum.de/virtual-life/gaming/strategie-simulatoren/terrain-importieren-transport-fever-2/>

Das funktionierte alles wie beschrieben und war auch hinreichend einfach. In Transport Fever 2 geladen kam dann aber die Enttäuschung: Die Landschaft sah aus, als hätte eine Horde Riesen Kinder im Sand gebuddelt: Überall unmotivierter Hügel von 30 bis 50 m Höhe. Es kann sein, daß die Höhenkarte woanders bessere Qualität hat.

Es gibt sehr zuverlässige Quellen von Höhendaten, die ein bis zwei Probleme haben: Die Landesvermessungsämter. Das erste Problem sind merkwürdige Dateiformate, das zweite ist, daß sich einige Bundesländer teuer bezahlen lassen, was von unseren Steuergeldern vermessen wurde.

Ich wurde schließlich fündig bei dem EU-finanzierten Copernicus:

<https://www.d-copernicus.de/daten/datenzugang/>

Da muß man sich (kostenlos) anmelden und kann dann Daten herunterladen. Zugriff auf zusätzliche Daten und wohl noch einfacher geht es über OpenDEM:

<https://www.opendem.info>

wo ich tatsächlich die Daten hergeholt habe (Grundlage ist trotzdem die Copernicus-Vermessung) über das Menü **OpenDemEurope** → **Download high-res. tiles**.

1.3. Höhenkarte verfügbar machen

So, jetzt haben wir eine oder mehrere riesige Dateien mit merkwürdigem Format. Es gibt ein freies Programm namens microdem

<https://www.usna.edu/Users/oceano/pguth/website/microdem/microdem.htm>

mit dem man die Dateien für normale Programme vorverdauen kann. Ein Tutorial mit geblockten Bildern findet sich bei Trainz:

<https://www.trainz.de/lexikon/entry/10-hog-digitale-geländemodelle-mit-microdem-vorbereiten/>
von mir ist das allerdings völlig ungetestet, und die Anleitung ist auch schon ein paar Jahre alt.

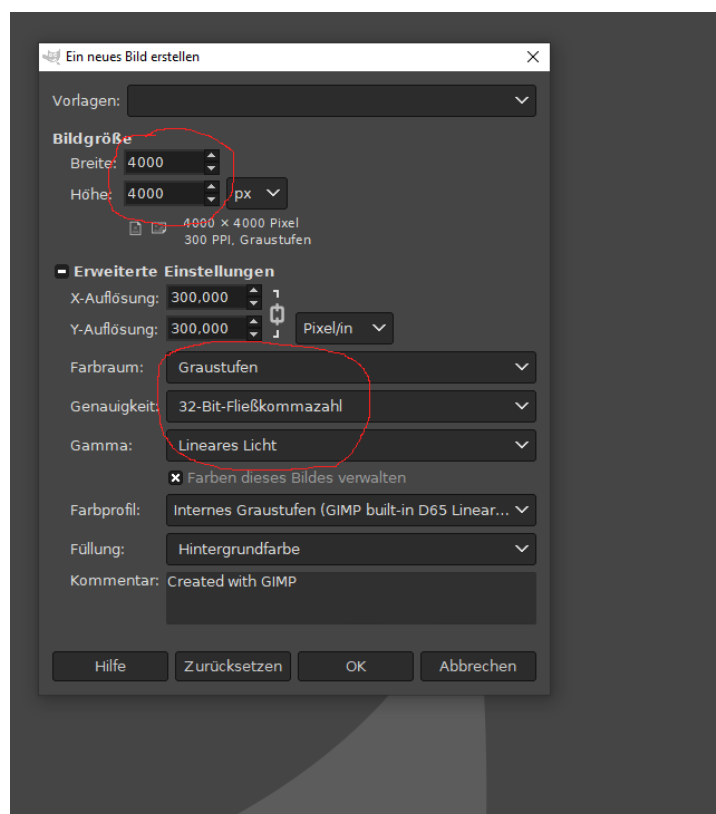
Falls die Daten mehr als eine Datei bzw. ein Bild erfordern, muß man sich merken, welchen Ausschnitt die jeweilige Datei beschreibt: Unten beschreibe ich einen Fall, in dem man erst nach dem Zusammenfügen sieht, ob die Teile an der richtigen Stelle liegen.

Wer trotz des Überlapps, den die UTM-Streifen haben, Karten aus mehreren Streifen nehmen muß, hat jetzt erst einmal einen Haufen Arbeit vor sich, das in einem sauberen Bild zu vereinigen. Ich habe es nicht probiert. Vielleicht hilft microdem weiter. Man kann auch die Karten erst skalieren und dann mittels dem unten beschriebenen Hugin zusammennähen. Wahrscheinlich ist es leichter, sich von seiner Lieblingsgegend zu verabschieden.

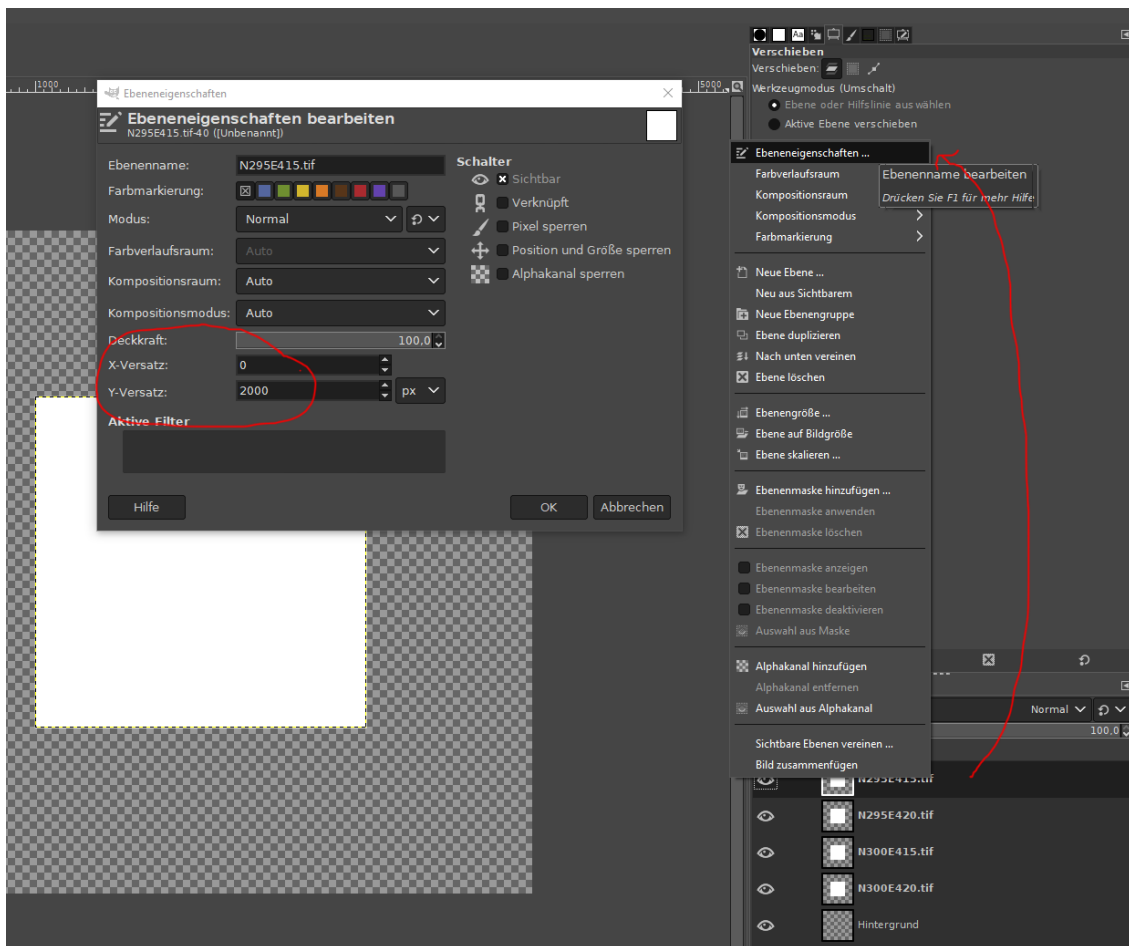
1.3.1 Karten zusammensetzen

Was ich mit heruntergeladen hatte, waren vier TIFF-Dateien, die zwar nicht von den primitiven Malprogrammen, aber von GIMP geöffnet werden kann. Also, mit GIMP geöffnet und einen strahlt eine völlig weiße Fläche an. Woran liegt das? Das Dateiformat, verrät GIMP, ist ein Graustufenbild in 32 bit float. Die Höhen werden richtig in Metern dargestellt, aber üblicherweise steht in diesem Format 0 für schwarz und 1 für weiß. Bereits die Deichkrone ist also noch weißer als weiß. Man muß also die Farbe skalieren, was im nächsten Abschnitt passiert. Wer Spaß daran hat, kann sich mit der Pipette die Farbe als RGB-% auslesen. Die astronomisch hohen Zahlen sind im Grunde cm über dem Amsterdamer Pegel (für Deutschland).

Wenn man mehr als eine Karte hat, müssen sie vorher zusammengefügt werden, und zwar vor jedem Skalieren, damit die Anschlüsse auf gleichem Grauwert bleiben. Sonst gibt es an den Kartengrenzen schnurgrade Felswände. Dazu erzeugt man in GIMP zunächst ein Bild der Größe, die alle Karten richtig nebeneinandergelegt aufnehmen kann. Meine Einzelbilder waren 2000×2000 Pixel groß und lagen im Quadrat, also hatte mein Bild 4000×4000 Pixel, und zwar wie die Einzelbilder in 32-bit float Genauigkeit. Außerdem sollte man lineares Licht aktivieren.



Jedes Bild wird dann in eine eigene Ebene geladen, und die Ebene wird dann an ihre Position geschoben (Schaltfläche Ebeneneigenschaften, X-Versatz und Y-Versatz). Wo möglich, sollte man übrigens diese Positionen als Zahl von Hand eingeben; mit der Maus wird das eine unendliche Fummelei, die dann doch um ein oder zwei Pixel schief geht. Felswand. Alle Ebenen vereinigen (Strg-M) und man hat nur noch ein Bild.



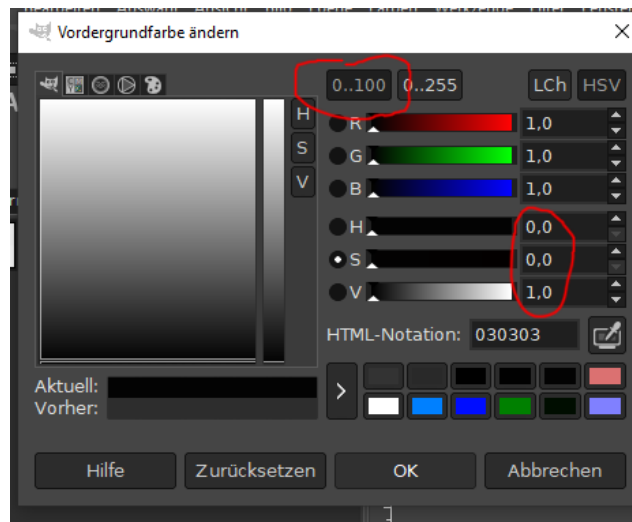
1.3.2 Graustufen skalieren

Wer jetzt schon ein wundervoll sichtbares Graustufenbild hat, kann gleich in den nächsten Abschnitt springen.

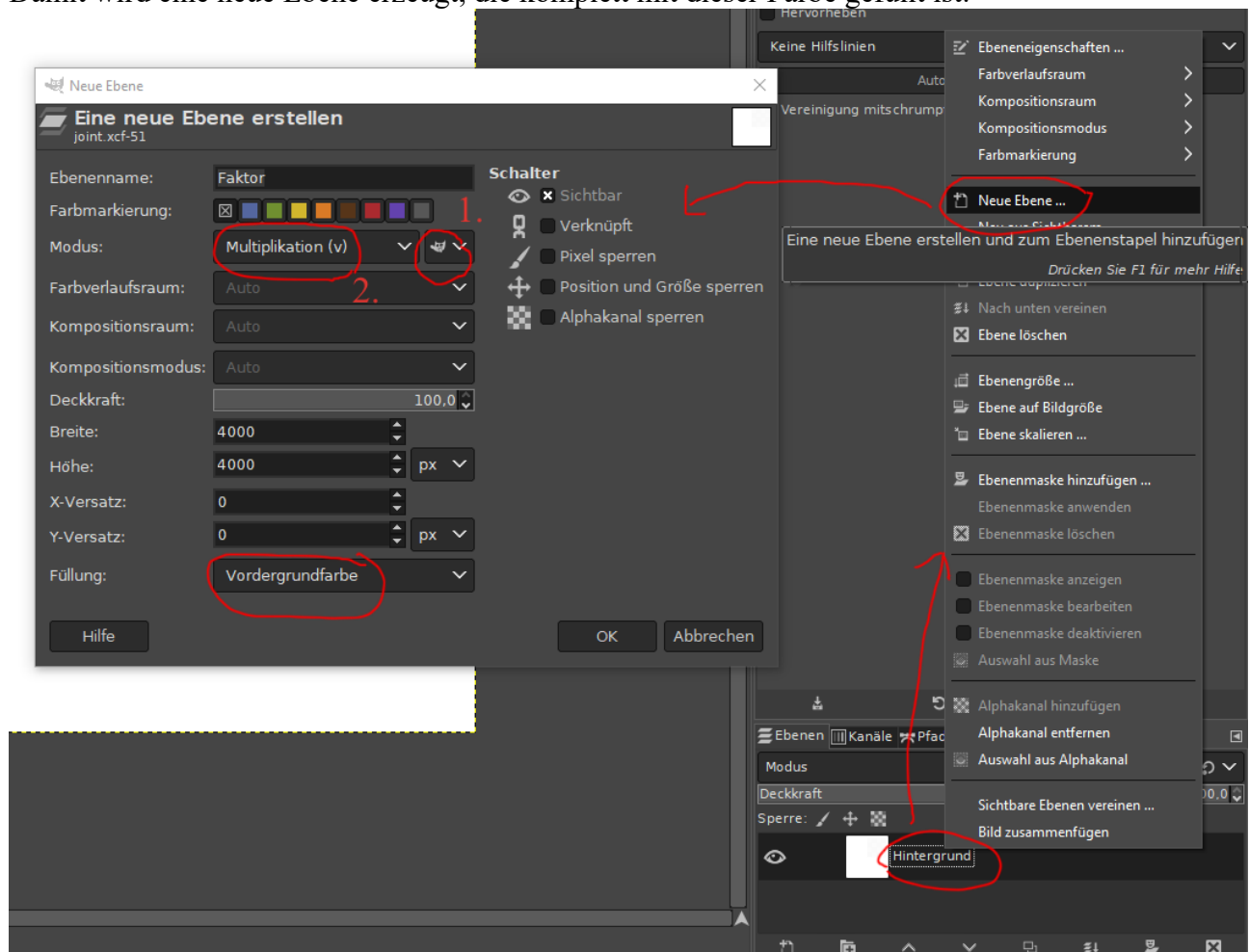
Und noch etwas: Man kann diesen Schritt, wie oben gesagt, auch vor dem Zusammenfügen ausführen. Dann muß man für jedes Einzelbild genau die gleichen Operationen vornehmen.

Ich bin an dieser Stelle in so eine Falle getappt, die mir erst sehr viel später aufgefallen ist, nämlich als ich im Karteneditor den Rhein ausbaggern wollte. Zum Skalieren hatte ich Farben→Automatisch→Kontrastspreizung benutzt (also: bitte nicht tun). Die sieht zwar sinnvoll aus, scheint aber visuell und nicht mathematisch linear skalieren zu wollen. Das Resultat war, daß alle niedrigeren Erhebungen fast wegoptimiert waren: Die Mainzer Zitadelle war gerade mal drei Meter über dem Rhein.

In GIMP läßt sich das Problem (noch) umgehen: Die veralteten Ebenenmodifikatoren „Multiplizieren“ und „Abziehen“ funktionieren mathematisch korrekt – die Standardversionen leider nicht, auch wenn man überall lineares RGB einstellt. Also setzt man in einem ersten Schritt die Vordergrundfarbe auf 1 % Weißwert:



Es ist hier praktisch, die Farbwertskala 0..100 zu benutzen; das sind Prozent der vollen Sättigung. Damit wird eine neue Ebene erzeugt, die komplett mit dieser Farbe gefüllt ist:



Dann wird vielleicht schon einmal ein Teil des Geländes sichtbar – nämlich wenn es unter 100 m liegt. Jetzt rechtsklickt man noch einmal auf eine der Ebenen und wählt in dem gleichen Menü wie vorher das zweitunterste, Sichtbare Ebenen vereinen ... aus. Das wird einfach bestätigt. Spätestens jetzt sollte man sich Gedanken machen, welche Höhen man wie im Spiel importieren will. GIMP ist leider nicht sehr gesprächig, was diese Werte anbelangt, oder ich war zu doof zum Gucken. Man kann diese Höhen aber auch anders finden: Es ist der höchste Berg im betrachteten Gebiet, den man oft sogar in der Wikipedia findet, sowie die Stelle, an der der tiefstgelegene Fluß aus dem Bild

fließt, was man beispielsweise in Google Earth feststellen kann. Es ist kein Problem, den höchsten Wert etwas auf- bzw. den niedrigsten abzurunden.

In meinem Fall war der niedrigste Punkt etwa bei 82 m, der höchste bei 578 m. Dieses Höhenintervall muß in den Bereich von 0 bis 1 gebracht werden. Da ich die Originalhöhen im Spiel haben möchte, lag es nahe, den Import im Spiel von 80 m bis 580 m zu wählen. Das gibt eine Spanne von genau 500 m. Der erste Schritt brachte das Höhenintervall auf den Bereich von 0,8 bis 5,8. Deshalb müssen die Höhen noch einmal durch 5 geteilt werden. Der vorige Schritt wird mit einem Weißwert von 20 % wiederholt. Damit sind wir bei einem Intervall von 0,16 bis 1,16.

Das ginge in diesem Beispiel wohl auch in einem Schritt, aber bei anderen Intervallen ist die Eingabemaske zu grobkörnig: Bei einem Intervall von 500 m könnte ich auch 0,2 % eingeben. Bei 600 m müßte man mit 0,1667 multiplizieren, der nächstmögliche Wert ist aber auch 0,2 %. Division ist nicht sinnvoll möglich, da die Eingabewerte auf 500 % beschränkt sind.

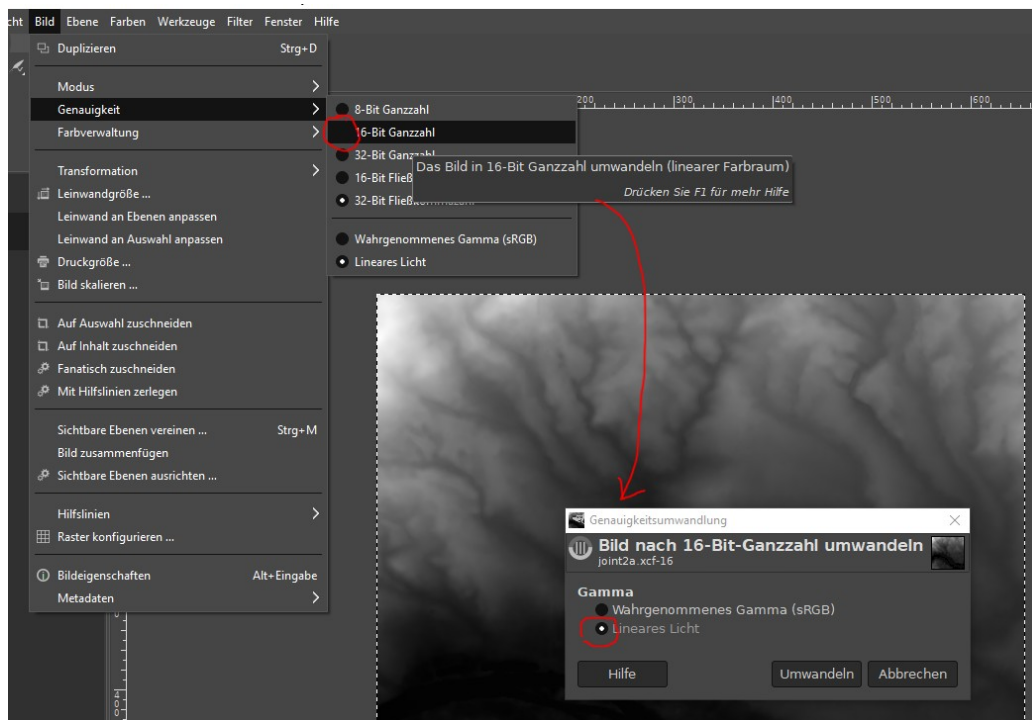
Nach dieser Skalierung müssen noch die 80 m abgezogen werden, was mit dem gleichnamigen Modus (auch die veraltete Version!) und einer Ebene mit 16 % Weißwert passiert. Warum 16 %? Weil $80:100:5 = 0,16$. Warum zieht man die 80 m nicht ab, bevor man sowas ausrechnen muß? Weil bei diesem Modus das Ergebnis so beschnitten wird, daß alle Farbwerte zwischen 0 und 1 liegen.

1.3.3 Beschneiden

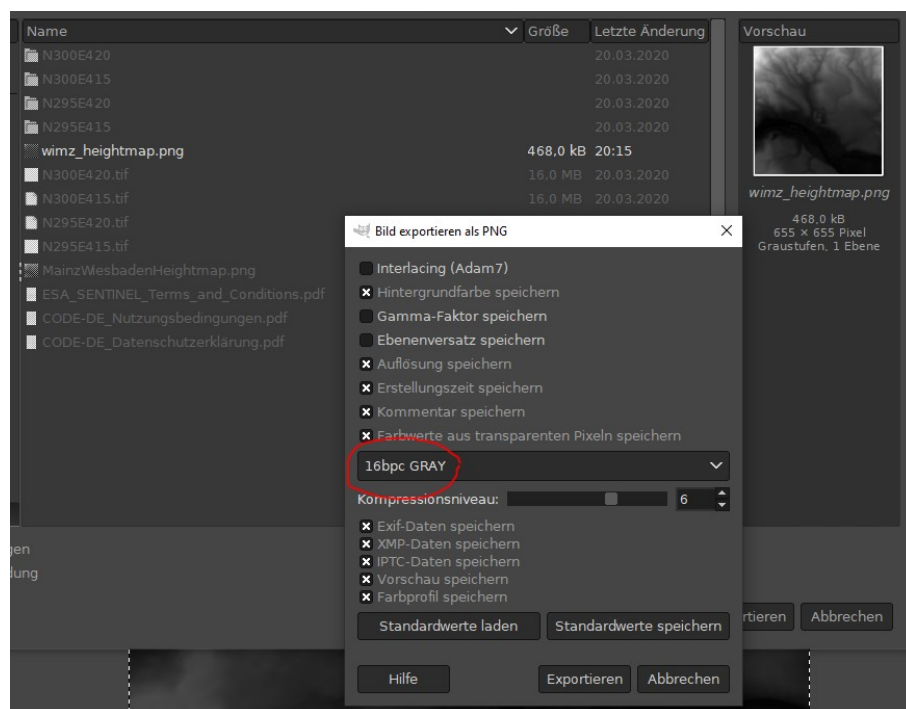
Man kann das Bild entweder gleich auf den richtigen Ausschnitt trimmen oder nur grob beschneiden. Letzteres ist sinnvoll, wenn die Karte oder das Luftbild zum Auswählen benutzt werden sollen. Ich habe die Höhenkarte als Maß genommen, weil ich da sicher bin, daß ein Pixel $25 \times 25 \text{ m}^2$ entspricht und ich eine 1:1-Karte haben wollte. Eine große Karte in Transport Fever (2) hat 4096×4096 Pixel, also exakt 16384 m Kantenlänge. Also muß ich einen Ausschnitt von 655×655 Pixeln aus der Höhenkarte schneiden ($16384 \text{ m} : (25 \text{ m/Pixel}) = 655,36 \text{ Pixel}$). Das Auswahltool von GIMP zeigt glücklicherweise die Größe des gewählten Ausschnitts an. Fürs Wiederfinden lohnt es sich, die Werte der Maske zu notieren. Das Schneiden selbst erledigt der Menüpunkt **Bild→Auf Auswahl zuschneiden**.

Eventuell ist es sinnvoll, hier zwischenspeichern für die Geländekarte. Dann wird das Bild mit **Bild→Bild skalieren** auf die richtige Größe skaliert; für die sehr große Karte also 4097×4097 Pixel (Möglicherweise skaliert Transport Fever 2 korrekt, ich habe aber eher den Eindruck, daß falsche Formate stillschweigend ignoriert werden).

Als mißtrauischer Mensch konvertiere ich erst zu Fuß in 16-Bit Graustufen:



Zum Schluß wird es als .png-Datei gespeichert, wobei unbedingt darauf zu achten ist, daß 16-bit Graustufen ausgewählt werden. Die Heightmap ist damit fertig.



Ein paar unvollständige Antworten auf nicht gestellte Fragen:

F: Warum wirkt das Terrain so flach, obwohl es anscheinend die richtigen Höhen hat?

A: Weil das Spiel der Kamera offensichtlich eine Art Weitwinkelobjektiv verpaßt hat.

F: Warum macht es das?

A: Weil man sonst merkt, daß die Karte arg klein ist. Und weil andernfalls viel zu viele Objekte berechnet werden – oder im Nebel verschwinden – müßten.

F: Was kann ich dagegen tun?

A: Einigermaßen sinnig: Die Höhen beim Import in den Karteneditor übertreiben. Das führt natürlich zu steileren Strecken auf die Berge.

F: Ich will aber trotzdem sowohl realistische Höhen als auch Ansichten!

A: Auch, wenn ich auf die Schnelle nichts gefunden habe, halte ich es für möglich, daß die Spiel-Kamera konfigurierbar ist. Spätestens, indem man in den entsprechenden Vertex-Shadern herumfummelt. Forsehe und teile mit!

Beim Kamera-Tool wählt man einfach eine Brennweite von 35 mm aus.

2. Geländekarte

2.1. Beschaffen

Als Geländekarte kann man, je nach Laune, einen Stadtplan bzw. eine Straßenkarte, eine topographische Karte oder ein Luftbild benutzen. Das Vorgehen ist im Prinzip gleich. In den meisten Fällen braucht man mehrere Einzelbilder, die zusammengesetzt werden müssen. Auch hier gibt es verschiedene Quellen. Google Maps ist eine, wie man höher auflösende Bilder bekommt, ist hier beschrieben:

<https://www.netzwelt.de/news/96171-google-maps-so-erstellen-poster-landkarten.html>

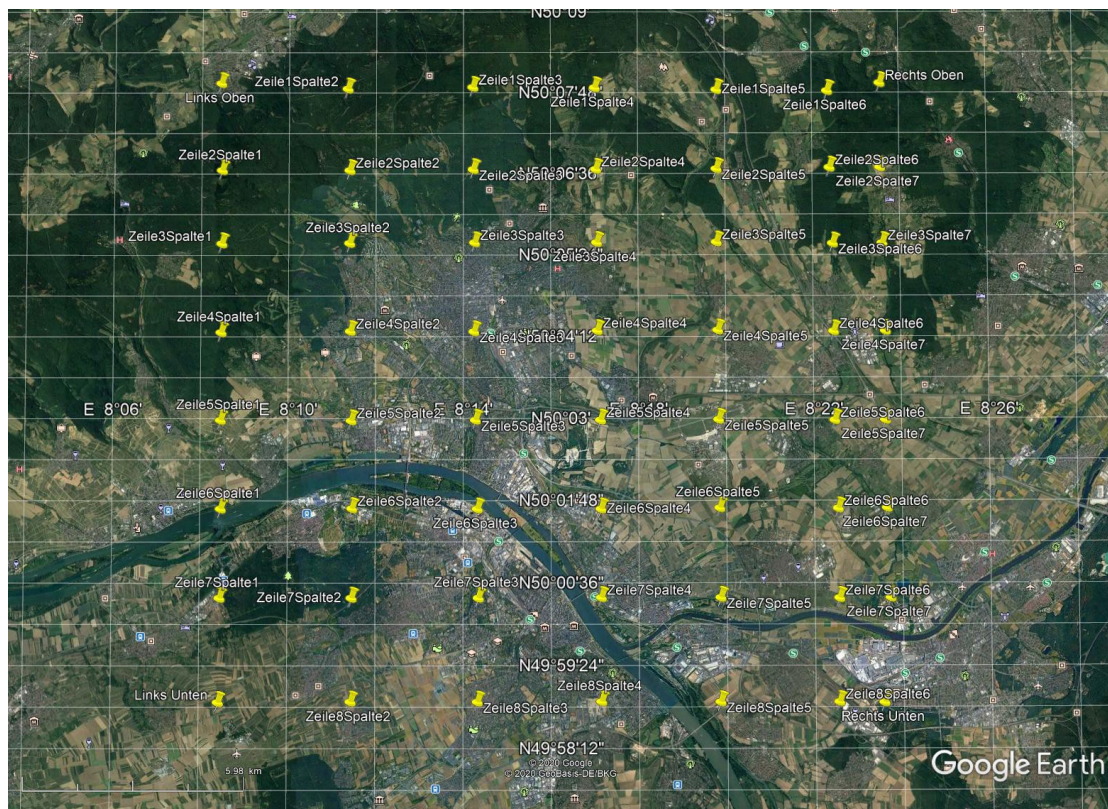
Man sollte hier keine größeren Ausschnitte als ca. 4000×4000 Pixel wählen.

Google Earth ist auch eher ein Geduldsspiel, aber wohl etwas einfacher, und man kommt um die meisten „hilfreichen“ Einblendungen herum.

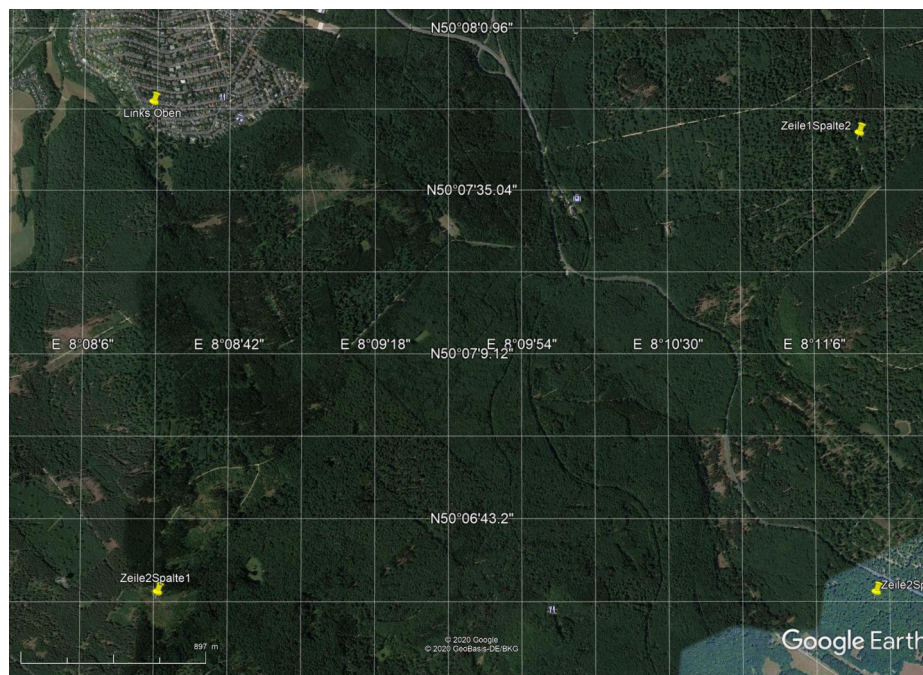
Andere Quellen sind wieder die Landesvermessungsämter, die, je nach Bundesland, auch topographische Karten zur Verfügung stellen; wer aufs realistische Aussehen weniger Wert legt, ist mit einem Stadtplan für die Straßen vielleicht besser bedient.

Bei Google Earth sollte man zuerst in den Optionen unter Navigation *Beim Zoomen nicht automatisch neigen* auswählen und die *Geschwindigkeit beim Anfliegen* maximal stellen, sowie *Beim Drehen oder Zoomen die Erde schrittweise verlangsamen* abwählen; das sind Spielereien, die im Zweifel nur nerven. Außerdem die Höhenverstärkung auf den kleinsten Wert setzen, also 0,01 – das vermeidet Verzerrungen durch das Höhenprofil. Dann sollte man regelmäßig nach Norden ausrichten, damit die Einzelbilder brav parallel liegen. Ich habe zuerst Sichthöhe von ca. 8 km gewählt (rechts unten angezeigt), was für grob für eine Kantenlänge von 2 m pro Pixel ausreicht. Für 1 m nimmt man eine Sichthöhe von 4 km; das war dann auch meine endgültige Map.

Ich habe die Ortsmarkierungen von Google Earth exzessiv benutzt. Sie geben Orientierung und lassen einen auch die Sichthöhe wiederfinden. Bei der ersten Markierung, die man setzt, sollte sie also stimmen und dann bei den weiteren überprüft werden. Mit diesen Markierungen wird ein Raster aufgesetzt.

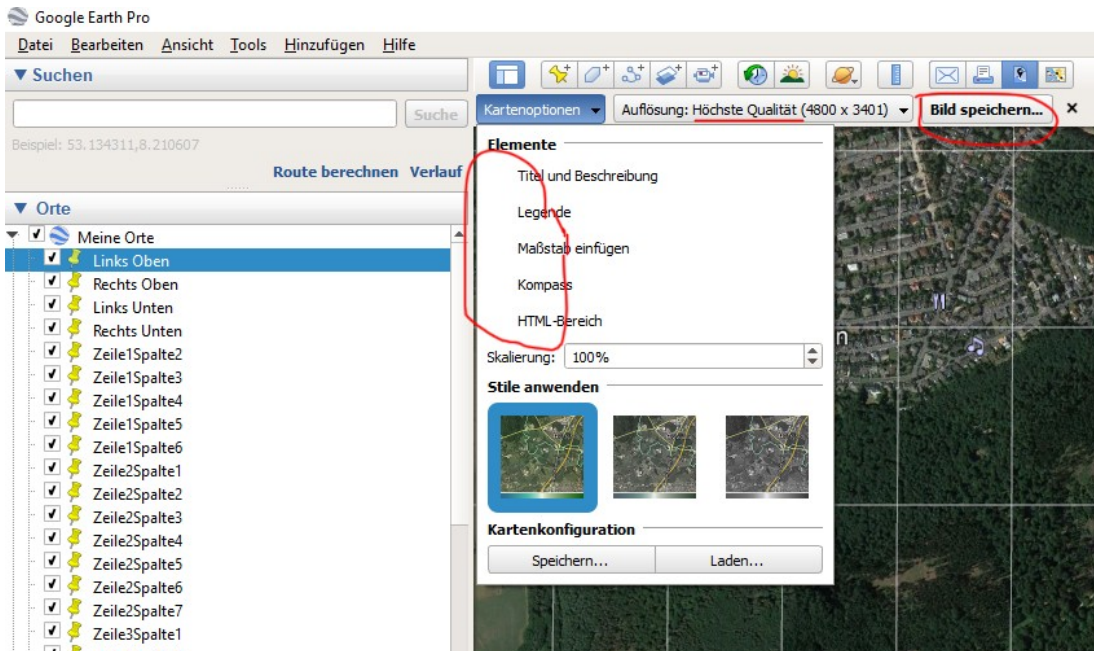


Das Raster muß keineswegs perfekt sein. Es muß aber den gesamten gewünschten Bereich überdecken, und bei der gewählten Sichthöhe müssen jeweils vier Marken auf einem Bild sichtbar sein können:



So wird garantiert, daß die Einzelbilder, in meinem Beispiel 42 Stück, alle ein wenig überlappen. Nach diesen Vorbereitungen wird dann **Datei→Speichern→Bild speichern** gewählt und der Aufforderung, die Ansicht benennen gefolgt oder auch nicht. Rechts neben dem Suchfeld werden die Kartenoptionen geändert (sehr empfohlen: alles abgewählt), die Auflösung gewählt (höchste Qualität, es sei denn, man will schlechtere Auflösung für die gleiche Arbeit) und dann gespeichert mit einem Namen, der möglichst etwas über die Position des Bildes in der Gesamtkarte aussagt.

Ansicht→Gitternetz abwählen. Nach dem Positionieren *Meine Orte* in der Leiste links abwählen. Beides gibt sonst blöde Einträge im Bildschirmfoto. Ein kleines „Google Earth“ links unten mußte ich von Hand in GIMP abschneiden.



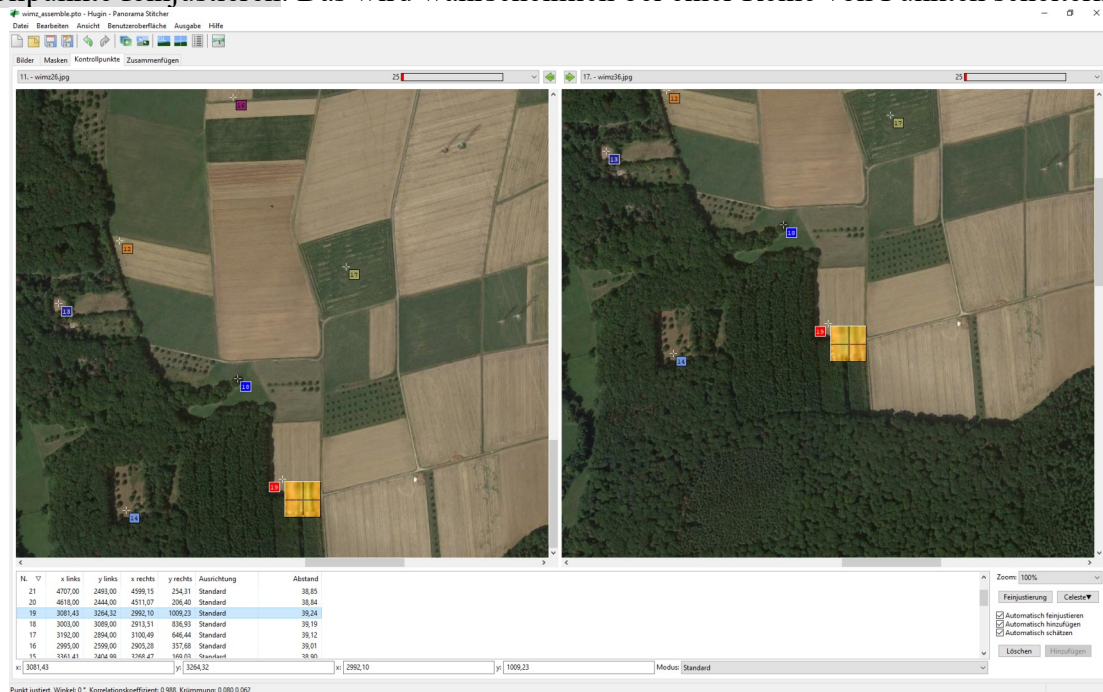
2.2. Eine einzige Karte

Zu diesem Punkt hatte ich also 42 Einzelbilder – da es schon die zweite Version war, recht eng ausgewählt; wer sich nicht sicher ist, macht mehr Bilder oder paßt die Höhenkarte an das Satellitenbild an. Daraus soll ein einzelnes Bild werden. Ganz von Hand geht das net. Hier tritt Hugin in Aktion – es soll wohl eine brauchbare Alternative in Photoshop geben, aber das kenne ich nur vom Hörensagen bzw. Browsen. Das Stichwort hier heißt neudeutsch Stitching. Alles Nötige ist in dem folgenden Tutorials enthalten:

<http://hugin.sourceforge.net/tutorials/scans/en.shtml>

<http://hugin.sourceforge.net/tutorials/perspective/en.shtml>

Da viele Kontrollpunkte hinterher ein besseres Bild geben, werden sie zunächst einmal automatisch gewählt: Im Reiter *Bilder* zwei benachbarte Bilder auswählen und im Bereich links unten auf *Kontrollpunkte erstellen* klicken. Danach erfolgt die Feinjustage mit dem Menüpunkt *Bilder→Alle Kontrollpunkte feinjustieren*. Das wird wahrscheinlich bei einer Reihe von Punkten scheitern:



Erfahren kann man diese Punkte, indem man **Ansicht→Kontrollpunktetabelle** im Menü aufruft oder F3 drückt. Nach dem ersten Ändern von Kontrollpunkten sind diese Ergebnisse leider wieder weg, also aufschreiben, Schirmschuß oder ähnliches. Man bearbeitet die Punkte, indem man den Reiter Kontrollpunkte aufruft, und die beiden angegebenen Bilder in die Ansichten lädt. Wie man in dem Bild sieht, ist dieser Punkt schlecht gewählt: Wenig Farbkontrast und die sichtbaren Strukturen haben alle die gleiche Richtung. Deshalb wird der Kontrollpunkt versetzt (Nummer links anklicken, festhalten, Kreuz an die richtige Position schieben, loslassen). Wir wählen einen markanten Punkt in der Nähe mit etwas größerem Abstand zu den anderen Kontrollpunkten.

Beim Punkt Optimieren sollte man entgegen der Anleitung im ersten Tutorial auch auch den Nickwinkel in die Optimierung einbeziehen. Das korrigiert, falls man in Google Earth in dem einen oder anderen Fall doch nicht senkrecht auf die Landschaft geschaut hat. Bei den meisten Bildern sollte nach dem Optimieren aber der Nickwinkel keinen größeren Betrag als 0,2 haben.

Wer eine Reihe Bilder schief aufgenommen hat, findet im zweiten Tutorial die Anleitung, wie man horizontale und (hier v. a.) vertikale Kontrolllinien setzt.

Nach dem Optimieren erzählt einem Hugin etwas über die Abstände von Kontrollpunkten. Gemeint sind die Abstände des selben Kontrollpunktes auf zwei verschiedenen Bildern, nachdem diese übereinandergelegt wurden. Dies erfährt man wieder über die Kontrollpunktetabelle. Abstände von weniger als einem halben Pixel sind unvermeidlich, fünf Pixel sind auf jeden Fall zu korrigieren (wie oben), bei weniger als zwei Pixeln sollte man sich überlegen, wie viel man tolerieren mag.

Die Vorschau sollte man sich in jedem Fall anzeigen lassen: Wenn das Bild verzerrt oder auch nur merkwürdig erscheint, hat man einen Fehler gemacht. Wenn es einfach nur kissenförmig verzerrt ist, sollte man allerdings erst sicherstellen, daß die Projektion *geradlinig* (*rectilinear*) ist. Wahrscheinlich bei den horizontalen oder vertikalen Linien, die man von Hand gesetzt hat. Das hat mich auch ein paar Runden gekostet.

Während des eigentlichen Zusammenfügens eines großen Bildes hat man Zeit, in der Steuererklärung ein ordentliches Stück voranzukommen oder an einem Tutorial zu tippen.

3. Karten aneinander anpassen

Jetzt nur noch das Satellitenbild zurechtschnippeln und veröffentlichen? Ist leider bei Transport Fever öfters so gewesen. Das hat mir einige überflutete Uferstraßen beschert, was bei TpF1 leider Nachbessern und einen Neubeginn des Spiels erforderte. Also: Bilder anpassen. Das geht z. B. mit GIMP, dessen Weg ich hier beschreibe:

Zunächst lädt man die zugeschnittene Höhenkarte aus dem ersten Teil und konvertiert das Bild in 16-bit RGB. Wer zu wenig Speicher hat, kann es auch mit 8-bit versuchen, dann empfiehlt sich aber vor der Konvertierung, die Helligkeitskurve so zu bearbeiten (**Farben→Kurve...**), daß die wichtigen Strukturen sichtbar werden. Man kann so z. B. Flüsse vom nur wenig höheren Ufer unterscheiden. Dann lädt man die Geländekarte aus Teil 2 in eine eigene Ebene und vergrößert das Bild (**Bild→Leinwand an Ebenen anpassen**), so daß alles seinen Platz hat. Die Höhenkarte auf Transparenz stellen (98% für dunkle Bereiche, ~30% für helle; man wird häufiger wechseln wollen) und in den Vordergrund schieben.

Die Höhenkarte sollte bei mir nicht verzerrt werden, die Geländekarte aber zwischendurch nicht viele Pixel verlieren. Deshalb habe ich zuerst die Graustufenkarte so vergrößert, daß sie die Vorlage für das fertige Bild ist, hier also 16384×16384 Pixel.

Wegen einiger „Features“ von GIMP hier unbedingt speichern!

Jetzt kommt nämlich die nicht immer zuverlässig arbeitende Perspektivkorrektur ins Spiel. Zunächst wird das Graustufenbild wieder intransparent gemacht und in den Hintergrund geschoben. Dann wir die Perspektivkorrektur aufgerufen: **Werkzeuge→Transformationen→Perspektive**. Die Idee dieser Korrektur ist, daß man die Ecken des Bildes verschieben kann und der Rest nachfolgt. Damit kann man das Hintergrundbild an die Höhenkarte anpassen, so daß die Flüsse überall in ihren

Betten und die Bäche in den Tälern verlaufen. Man kann die obere Ebene auf Transparenz schalten (30% oder weniger dürfte passen) und so die Höhenkarte wieder sichtbar machen.

Wer beim Stitching sauber gearbeitet hat, wird hier praktisch alle wichtigen Elemente übereinander legen können, sonst hat man eine schwere Zeit (oder geht besser zum Stitching zurück).

Wenn alles paßt, unter einem neuen(!) Namen speichern und das Bild auf Artefakte prüfen: Bei mir sind mehrfach Flecken aus dem Graustufenbild in die Geländekarte gewandert, was das ganze unbrauchbar machte. Das passierte besonders gerne, als ich versuchte, das transparente Graustufenbild in der oberen Ebene zu halten.

Dann wird das ganze Bild auf die Fläche des Graustufenbildes beschnitten. In GIMP:

Geländekarte ausblenden (Auge rechts in der Liste wegklicken), so daß nur die Graustufenkarte gezeigt wird, dann Bild→Auf Inhalt zuschneiden. Es sollte eine Leinwand der Größe des Graustufenbildes übrigbleiben. Dann Geländekarte sichtbar machen und das Graustufenbild löschen oder ausblenden. Mit Bild→Bild skalieren auf die gewünschte Größe für das Spiel bringen. Mein Weg für Transport Fever 2 verlangt hier ein Vielfaches von 4096 Pixeln als Kantenlänge. Jetzt werden Hilfslinien mit Bild→Hilfslinien→Neue Hilfslinie gesetzt, und zwar horizontal und vertikal so, daß quadratische Kacheln von 4096×4096 Pixeln entstehen. Die Positionen habe ich wieder von Hand eingetragen (4096, 8192, 12288, ...). Mit Bild→Mit Hilfslinien zerlegen werden diese Kacheln zu einzelnen Bildern gemacht. Diese werden nacheinander aktiviert, vertikal gespiegelt (Ebene→Transformation→Vertikal spiegeln) und als .dds-Datei exportiert. Dabei wählt man

Compression: BC1 / DXT1

Mipmaps: Generate mipmaps

Wrap mode: Mirror oder Clamp könnte das beste sein, verhindert aber leider doch nicht den Streifen im Spiel. Ich fürchte, der Wert wird nicht benutzt.

4. Einbau in Transport Fever 2

4.1 Höhenkarte

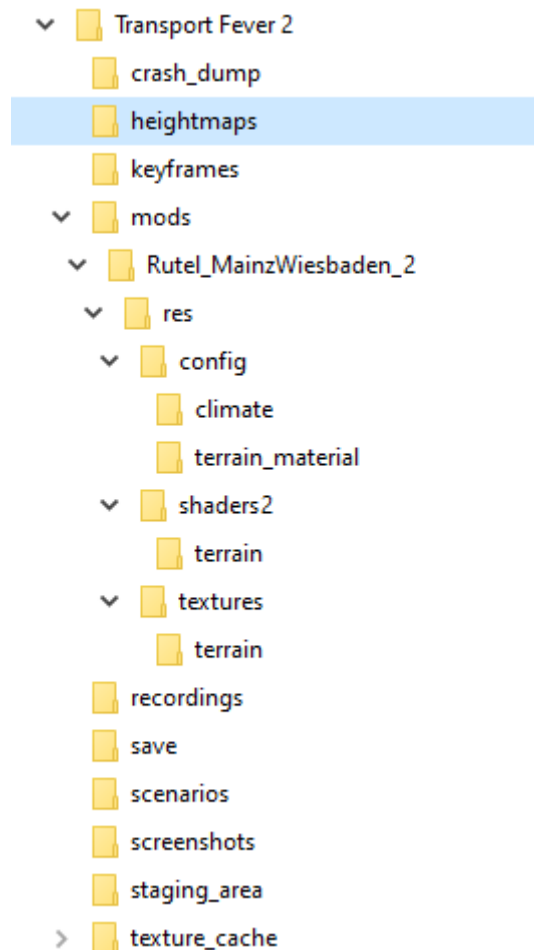
Das ist einfach: In den Benutzerordner schieben, und dann kann sie bei der Kartenerstellung ausgewählt werden. Der Benutzerordner ist bei meiner GOG-Installation

C:\Users\...\AppData\Roaming\Transport Fever 2\heightmaps

4.2 Geländekarte

Das erfordert deutlich mehr Arbeit. Es ist nämlich nötig, eine Schmalspur-Mod zu erstellen. Dazu baut man im Benutzerordner C:\Users\...\AppData\Roaming\Transport Fever 2\mods den umseitigen Verzeichnisbaum auf.

Den Namen der Mod, bei mir Rutel_MainzWiesbaden_2, ändert man sinnvoll mit dem eigenen gewählten Namen und dem Namen der Karte ab – und, bei der Erstfassung, mit _1 am Ende. Im Verzeichnis des Hauptspiels verändert man bitte nichts, auch nicht im mod-Unterschied, wie noch bei Transport Fever 1.



Bei den folgenden Dateilistings sind die gleich bleibenden Zeilen schwarz, die sinnvoll zu ändernden blau, neu hinzukommende rot, an der Stelle von Kürzungen steht ein grüner Kommentar, der ersatzlos gelöscht werden kann.

Im Stammverzeichnis der Mod stehen die beiden Dateien mod.lua und strings.lua.

mod.lua:

```
function data()
return {
  info = {
    name = _("modName"),
    description = _("modDesc"),
    minorVersion = 0,
    severityAdd = "NONE",
    severityRemove = "NONE",
    authors = {
      {
        name = 'MyName',
        role = 'CREATOR',
      },
    },
  },
  -- options = ..
  runFn = function (settings)
  end
  -- checkActiveFn = ..
}
end
```

und **strings.lua**:

```
function data()
return {
  en = {
    modName = "Mainz Wiesbaden Image",
    modDesc = "Adds a satellite image of Mainz and Wiesbaden area as ground
texture.",
  },
  de = {
    modName = "Bild Mainz Wiesbaden",
    modDesc = "Setzt ein Satellitenbild der Gegend um Mainz und Wiesbaden als
Bodentextur.",
  }
}
end
```

Für die Datei in res\config\climate kopiert man sich die Datei für die gewünschte Klimazone aus dem Hauptspiel und ändert sie wie folgt ab. Die Mainz-Wiesbaden-Karte benutzt temperate.clima.lua, die anderen Klimazonen unterscheiden sich in Details.

temperate.clima.lua:

```
local layersutil = require "terrain/layersutil"
local maputil = require "maputil"
local vec2 = require "vec2"
local tu = require "texutil"

function data()

return {
  id = "temperate",
  name = _("Temperate"),
  desc = _("Mixed forest vegetation"),
  mapColoring = {
    -- texture = {
      -- levels = { -100.0, 130.0, 260.0, 370.0, 450.0, 500.0 },
      -- fileName = "terrain/level_colors.tga",
    -- },
    ambientColor = maputil.MakeColor{ 205, 219, 255 },
    sunColor = maputil.MakeColor{ 255, 245, 240 },
  },
  groundTextures = {
    waterGround = "water_ground.lua",
    waterShore = "water_shore.lua",
    farmland = "farmland.lua",
    farmlandBorder = "farmland_border.lua"
  },
  skirt = {
    colorTex =
tu.makeTextureMipmapClampVertical("terrain/skirt_color.tga", false),
    detailTex = tu.makeTextureMipmapRepeat("terrain/skirt_detail.tga",
true),
    waterTex = tu.makeTextureMipmapClamp("terrain/water_skirt.tga",
false)
  },
  vehicleSet = "europe",
  order = 1,
  updateFn = function(params)
    local result = {
      layers = layersutil.Layer.new(),
```

```

    }
    local heightmap = "heightmap"
    local mkTemp = layersutil.TempMaker.new()
    -- mkTemp.doDebug = true
    -- Hier wurde gekürzt, weniger wichtig
    -- #####
    -- ##### River bed
    local layer8Map = mkTemp:Get()
    result.layers:Map(heightmap, layer8Map, { 65.0, 81.0 }, { 1.0, 0.0},
true)

    -- #####
    -- ##### MIX
    result.mixingLayer = {
        backgroundMaterial = "map01.lua",
        layers = {
            -- Hier wurde gekürzt, wichtig
            -- River materials
            {
                map = layer8Map,
                dither = true,
                material = "river_bed.lua",
            },
        }
    }

    mkTemp:RestoreAll(result.mixingLayer)
    mkTemp:Finish()

    -- maputil.PrintGraph(result)
    return result
end
}
end

```

Die Änderung in der ersten blau markierten Zeile kommt daher, daß ich den Wasserspiegel auf 82 m gesetzt habe; wer ihn bei 0 m läßt, muß nichts ändern.

Bei mehr oder weniger Kartenteilen ist die Zahl in der for-Schleife natürlich entsprechend zu ändern. Dann kommen ins Verzeichnis res\config\terrain_material für jeden Kartenteil eine Datei map*n*.lua (bei mir vier Dateien map1.lua bis map4.lua) mit dem Aufbau nach den Dateien aus dem entsprechenden Verzeichnis des Hauptspiels:

map01.lua:

```

function data()
return {
    name = _("Mainz Wiesbaden Kachel 0 0"),

    detailColorTexture = "terrain/wimz00.dds",
    detailMetalGlossAoHTexture =
"terrain/grass_light_green_metal_gloss_ao_h.dds",
    detailNormalTexture = "terrain/grass_light_green_nrml.dds",
    overlayTexture = "terrain/overlay_0.dds",

    detailSize = 1,
    overlaySize = 4,
    overlayStrength = 4 + 1001,

    order = 9001
}

```

```
}  
end
```

Bei einer anderen Anordnung als 4×4 Karten mit einer Auflösung von 1 m sind die Werte wie folgt zu ändern (es reicht bei der mapXX.lua, die in temperate.climate.lua als `backgroundMaterial` gesetzt wird; die anderen werden nicht benutzt): `detailSize` gibt die Anzahl Pixel pro Meter an (bei einer Karte mit halber Auflösung, d. h. bei 2×2 m großen Pixeln also 0.5). Je größer die Zahl, desto kleiner wird die Karte im Spiel angezeigt. `overlaySize` gibt die Anzahl der Kartenspalten und `overlayStrength` die der Kartenzeilen im Gesamtbild an. Der letzte Wert wird um 1001 erhöht, damit der Shader hinterher die Information hat, daß es sich tatsächlich um diese Geländekartenbilder handelt. Die Namen passen nicht; ich mußte die Variablen von einem eigentlich anderen Anwendungszweck kapern. Dabei mag ich keine Kapern.

Der Parameter `order` gibt bei allen Maps an, wo sich die Textur bei den Brushes findet. Eine hohe Zahl bedeutet, daß sie ziemlich weit rechts bzw. hinten ist, was hier sinnvoll ist. Benutzen will sie wahrscheinlich so schnell niemand. Die Karte, die alphabetisch als erste steht, ist diejenige links unten. Danach geht es zeilenweise nach oben. Bei mehr als zehn Karten muß also die erste `map01.lua` und nicht `map1.lua` heißen.

Das Verzeichnis `res\shaders2\terrain` enthält die Datei

level_color.fs:

```
#version 150  
  
... (hier wurde gekürzt; in der Datei nicht löschen!)  
  
LAYOUT_SAMPLER(1, 26) uniform sampler2DArray detailColorArrayTex;  
LAYOUT_SAMPLER(1, 27) uniform sampler2DArray detailNrmlArrayTex;  
LAYOUT_SAMPLER(1, 28) uniform sampler2DArray detailGlossArrayTex;  
LAYOUT_SAMPLER(1, 29) uniform sampler2DArray overlayArrayTex;  
  
const float texpixel = 1.0/4096.0;  
  
LAYOUT_UBO(1, 15) uniform DataBlock {  
    vec4 data[255];  
} u_dataBlock;  
  
vec3 overlay(vec4 col0, vec3 col1);  
  
vec3 transformNormal(vec3 normal) {  
    return normal;  
}  
  
vec3 getLevelColor(vec3 pos, int iLevel, vec3 normal,  
    out vec2 nrml, out vec3 glossAoH, vec3 dposdx, vec3 dposdy) {  
    vec3 doSizeoStrength = u_dataBlock.data[iLevel].xyz;  
  
    if (doSizeoStrength.z > 1000) {  
        float scale = doSizeoStrength.x;  
        vec2 tiles = doSizeoStrength.yz;  
        tiles.y -= 1001;  
  
        pos.xy = pos.xy * scale * texpixel + 0.5*tiles.xy;  
        dposdx = dposdx * scale * texpixel;  
        dposdy = dposdy * scale * texpixel;  
  
        nrml = vec2(0.5, 0.5);  
        glossAoH = vec3 (0.0, 0.2, 0.0);  
        int mapx = int(floor(pos.x));
```

```

        int mapy = int(floor(pos.y));
        int idx = mapx + int(tiles.x)*mapy + iLevel;
        vec2 subp = fract(pos.xy);
        vec4 col0 = textureGrad(detailColorArrayTex, vec3(subp.xy, idx),
dposdx.xy, dposdy.xy);

        return col0.rgb;
    }
    else {
        pos = pos * doSizeoStrength.x;
        dposdx = dposdx * doSizeoStrength.x;
        dposdy = dposdy * doSizeoStrength.x;

        nrml = textureGrad(detailNrmlArrayTex, vec3(pos.xy, iLevel),
dposdx.xy, dposdy.xy).rg;
        vec3 col = textureGrad(overlayArrayTex, vec3(pos.xy *
doSizeoStrength.y, iLevel), dposdx.xy * doSizeoStrength.y, dposdy.xy *
doSizeoStrength.y).rgb;
        vec4 col0 = textureGrad(detailColorArrayTex, vec3(pos.xy, iLevel),
dposdx.xy, dposdy.xy);
        col0.a = doSizeoStrength.z;
        glossAoH = textureGrad(detailGlossArrayTex, vec3(pos.xy, iLevel),
dposdx.xy, dposdy.xy).gba;
        glossAoH.z += u_dataBlock.data[iLevel].w;
        return overlay(col0, col.rgb);
    }
}

```

Sie paßt hoffentlich für jede Kartenanordnung und -auflösung. In gleicher Weise wird auch **level_color_tripplanar.fs**, die sich ebenfalls an der entsprechenden Stelle im Hauptspiel befindet, abgeändert und im Verzeichnis der Mod gespeichert. Schlußendlich enthält das Verzeichnis \res\textures\terrain die erzeugten Geländebilder, bei mir wimz00.dds bis wimz33.dds (die Ziffern geben Zeile und Spalte an).

5. Im Spiel

Die Mod wird wie jede andere Mod aktiviert. Sie ist aber nur sinnvoll zusammen mit der entsprechenden Höhenkarte. Konflikte erwarte ich unter Umständen bei Mods, die Brushes hinzufügen. Bei den Wassertexturen von Relozu und dem Street Mod Pack von majuen habe ich keine Schwierigkeiten gefunden, das kann aber auch Zufall sein. Ziemlich sicher bei denen bei Mods, die an den beiden modifizierten Shadern herumfummeln. Oder bei entsprechenden Updates seitens Urban Games – was schon zweimal passiert ist.

Als Klimazone wählt man die, deren .lua verändert wurde, hier also gemäßigt.

6. Problem meinerseits

Ich verstehe noch nicht, in welcher Reihenfolge die einzelnen Terrain-Texturen in detailColorArrayTex des Shaders geladen werden. Daher der blöde Trick mit den 1001 in mapxx.lua. Daß die Texturen in alphabetischer Reihenfolge der Dateinamen hintereinander auftauchen, ist experimentell bestimmt und im April 2020 von Urban Games bestätigt. Die Werte order und priority in der mapXX.lua haben keinen Einfluß auf diese Sortierung.

Immerhin funktioniert das jetzt bei mir seit einem Jahr.