Walnut Grove Map @ SVG - Projektbericht

Autor: Adrian Böhlen
Version 0.1.8
Datum: 28.03.2004

Inhalt:

- 1. Idee und Grundlagenbeschaffung
- 2. Datenerfassung
- 3. Konvertierung zu SVG
- 4. Attributierung
- 5. Symbolisierung
- 6. Weitere Funktionen
- 7. Dokumentierte Probleme

1. Idee und Grundlagenbeschaffung

TV's Walnut Grove, also das Gebiet, wo zwischen 1974 und 1983 die Serie "Little House On The Prairie", basierend auf den Büchern von Laura Ingalls Wilder gedreht wurde, liegt nördlich von Simi Valley, welches von Los Angeles durch das San Fernando Valley und über den Santa Susanna Pass zu erreichen ist.

Beschrieben ist das Gebiet vor allem auf der Website http://employees.oxy.edu/jerry/bigsky.htm. Dort findet sich auch eine einfache, nicht maßstabsgetreue Karte mit den einzelnen "Locations". Bessere Karten hat das Web nicht zu bieten, deshalb kam mir die Idee, selbst eine herzustellen. Anhand der topografischen Karten, die sich über die Seite http://terraserver.homeadvisor.msn.com abrufen lassen, konnte ich mit Hilfe der Beschreibung die betreffende Gegend relativ schnell ausfindig machen, obwohl sich auf diesen, von 1972 stammenden Karten, keinerlei Namen (außer "Big Mountain") und Gebäude finden lassen. Auf demselben Server sind auch schwarz-weiße Luftbilder neueren Datums gespeichert, deren Auflösung so gut ist, dass sich einzelne Gebäude, wie zum Beispiel die Ingalls Farm, noch ganz knapp erkennen lassen. Allerdings trifft das natürlich nur auf die noch existierenden Häuser zu, also nicht auf jene von Walnut Grove. Auf der Seite http://www.mapquest.com/maps konnte ich zusätzlich noch farbige Luftbilder finden, die eine leicht geringere Auflösung besitzen, auf denen sich gewisse Details, wie beispielsweise der Teich bei der Blindenschule , jedoch leichter indentifizieren ließen. Darüber hinaus finden sich zahlreiche Bilder sowie eine einfache Karte auch auf der Seite "Prairiehomestead" (http://www.geocities.com/prairiehomestead/lhotp/lhset), die sich ebenfalls als sehr hilfreich erwiesen.

Ausgerüstet mit diesen Grundlagen ging es anschließend darum, das optimale Format und die geeignete Software zu bestimmen. Da ich mich zu dieser Zeit schon ein bisschen mit dem neuen Web-Format SVG (Scalable Vector Graphics) befasst habe, war für mich sofort klar, dieses Format zu wählen. Es ist ein vom internationalen

Web-Konsortium (W3C) definiertes, offenes, textorientiertes, auf XML basierendes Format, welches eine Alternative zum proprietären Format Flash von Macromedia bilden soll. Programme, die SVG erzeugen, gibt es zur Zeit erst wenige, und die meisten arbeiten noch nicht optimal, sprich, ohne manuelle Bearbeitung geht fast nichts. Dies gilt noch verstärkt, wenn man zusätzliche Funktionen einbauen will und das empfiehlt sich sehr, denn da SVG ein offizielles Web-Format ist, klappt das Zusammenspiel mit anderen Web-Sprachen wie HTML, CSS und JavaScript reibungslos.

---

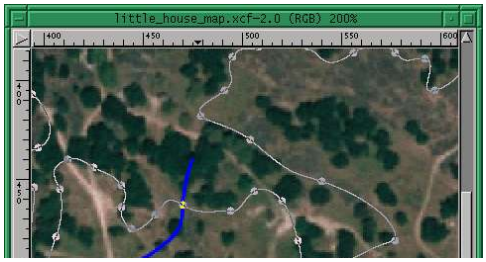
2. Datenerfassung

Um eine Karte im SVG-Format herzustellen, bedarf es vektorieller Daten. Die vorhandenen Quellen waren aber natürlich alle in einem Rasterformat verfügbar, d.h. die Umsetzung in Vektordaten musste von Hand erfolgen. Eine automatische Vektorisierung der Höhenlinien war auf Grund der nicht sonderlich guten Qualität der Rasterkarte nicht möglich. Für die Erfassung kam die Open-Source Software GIMP (Version 1.2) zum Einsatz, welche zwar grundsätzlich für die Rasterbearbeitung vor- gesehen ist, jedoch auch die Erfassung von Vektordaten zulässt. Der erste Schritt nach der Wahl des Ausschnittes war die Digitalisierung der Höhenlinien. Diese haben auf der Originalkarte im Maßstab 1:24'000 einen vertikalen Abstand von 25 Feet, entsprechend ca. 7.5 Metern.

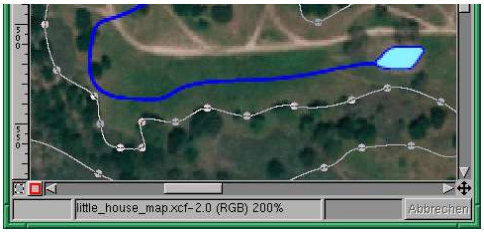
Für die Erfassung des Straßen- und Wegnetzes diente jedoch ausschließlich das Luftbild, welches zuvor anhand der Karte eingepasst werden musste. Hier wurde auch deutlich, dass nebst dem auf der Karte dargestellten Hauptweg eine Vielzahl weiterer Wege existieren.

Da die Sträßchen und Wege im allgemeinen gut erkennbar sind, auch im meist offenen Wald, war das Digitalisieren kein großes Problem.

Für die Erfassung der erwähnten offenen Waldgebiete wären grundsätzlich Baumsymbole am geeignetsten, es hätte aber einen enormen Aufwand bedeutet, diese einzeln abzuknipsen, deshalb entschied ich mich, diese Gebiete so gut wie möglich in



Flächen zusammenzufassen. Dies ist natürlich immer eine Interpretationssache. Bei den Häusern von Walnut Grove lag das Problem darin, dass sie nirgendwo auf einem Luftbild zu sehen sind, da sie zum Zeitpunkt der Aufnahmen ja schon nicht mehr existierten. Es gab also keine andere Möglichkeit, als sie anhand von

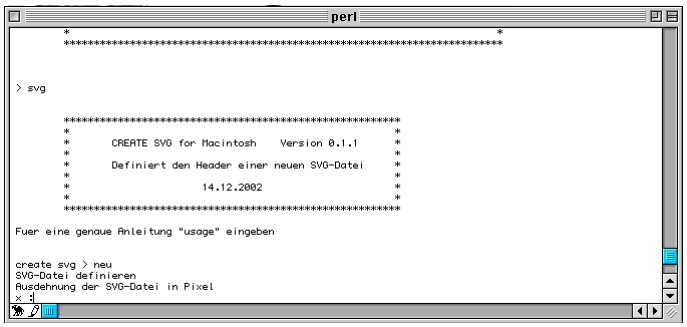


Wegnetz und Vegetation so gut wie möglich in die Gegend einzupassen. Hilfreich waren dabei die Notizen und Skizzen, die ich laufend während der Serie angefertigt habe. Dasselbe gilt auch für den Bach, welcher das Rad der Mühle speist und anschließend hinter Post und Schmiede in Richtung des Teiches abfließt. Auch dieser scheint nicht mehr zu existieren.

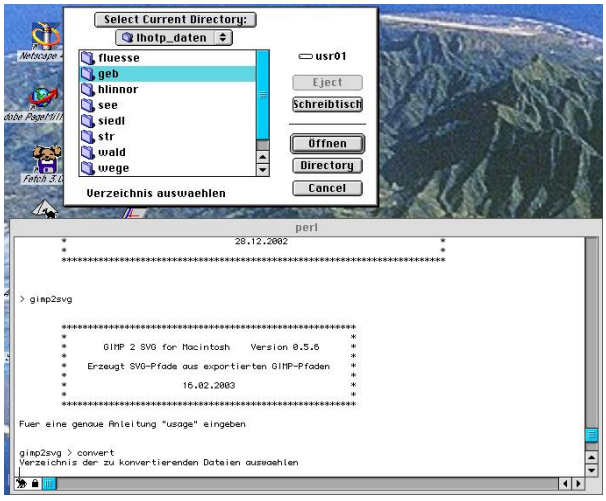
Zum Schluss mussten all dies einzelnen Vektordaten (Pfade) exportiert werden. GIMP generiert dabei ein einfaches ASCII-Format mit den x/y-Koordinaten jedes Punktes. Um nicht den Überblick zu verlieren, erstellte ich diverse nach Themen geordnete Unterverzeichnisse.

---

3. Konvertierung zu SVG



Als nächstes mussten nun diese dutzenden von Dateien in eine einzige SVG-Datei konvertiert werden. Für diesen Zweck schrieb ich bereits zu einem früheren Zeitpunkt ein Perl-Programm. In einem ersten Schritt wird dabei das "Gerüst" einer SVG-Datei angelegt, mit den Ausmaßen und der Symboldefinition in der Web-Sprache CSS.



Für die Eingabe dieser Parameter dient eine simple Kommandozeile, ebenso beim zweiten Programm, in welchem dann die einzelnen Dateiordner abgearbeitet werden und deren Informationen in die zuvor erstellte SVG-Datei geschrieben werden.

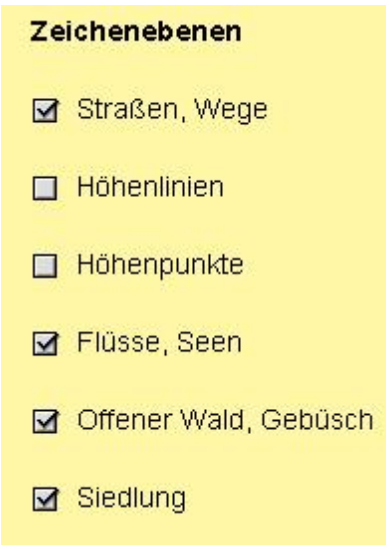
Auf diese Weise lassen sich aber nur linien- und flächenhafte Objekte erzeugen und konvertieren. Die Platzierung der Punkt- und Textobjekte erfolgte deshalb von Hand via Texteditor in der SVG-Datei selbst. Um diesen Vorgang zu erleichtern, baute ich eine mitlaufende Koordinatenanzeige in die Viewer-Datei ein. So lässt sich die gewünschte Lage einfach mit der Maus ermitteln und die Koordinaten brauchen dann nur noch abgelesen und in die SVG-Datei eingetragen zu werden.

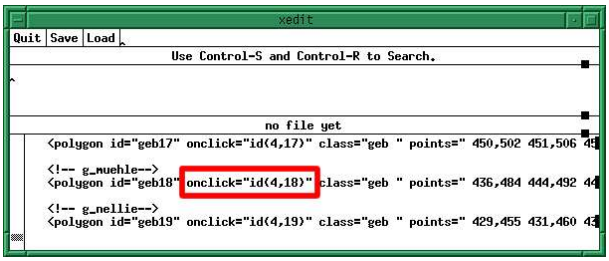
---

4. Attributierung

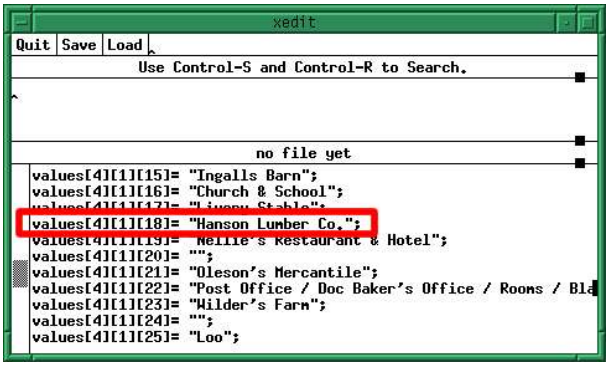
Im letzten Kapitel ist das Stichwort Viewer aufgetaucht: Um die Vorteile des SVG-Formates voll auszuschöpfen, reicht es nicht, diese einfach im Browser zu laden.

Deshalb wird sie üblicherweise in eine herkömmliche HTML-Datei eingebettet. Auf diese Weise lassen sich über Checkboxen einzelne Themenebenen ein- und ausblenden. Diese entsprechen den Gruppen der SVG-Datei. Außerdem lassen sich mit JavaScript weitere Funktionen programmieren. So können beispielsweise die Vektordaten mit Informationen verknüpft werden, wie es bei Geografischen Informationssystemen der Fall ist. Dazu ist jedes Objekt mit einer eindeutigen ID zu versehen, aus der sowohl die Nummer der Gruppe, wie auch die Nummer des Objektes abzuleiten ist. Das ganze muss dann mit der onclick-Funktion verbunden werden. Die eigentliche Attributtabelle ist als externe JavaScript-Datei gespeichert und wird in die Viewer-Datei eingebunden. Nebst der bereits bekannten ID enthält jede Zeile noch eine dritte Zahl, welche Auskunft darüber gibt, um welches Attributfeld es sich handelt. Denn es können selbstverständlich mehrere Attribute vergeben werden. Klickt der Benutzer ein Objekt an, wird eine JavaScript-Funktion aufgerufen, die ein kleines HTML-Fensterchen erzeugt, in welchem die zu dem betreffenden Objekt gehörenden Informationen angezeigt werden. Am Beispiel von Hanson's Mühle wird dies nachfolgend erläutert:



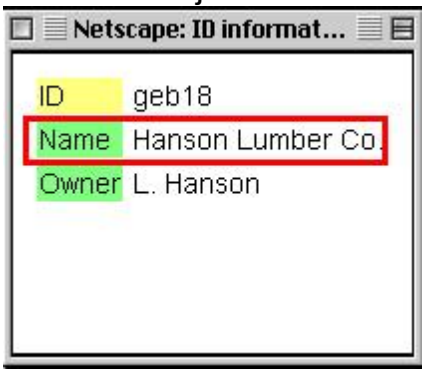


SVG-Datei mit zugehöriger ID (Gruppe 4, Objekt 18)



Attribut-Tabelle mit demselben Objekt (Gruppe 4, Attributfeld 1, Objekt 18)

Wird das Objekt im Browser angeklickt, erhält man folgendes Informationsfenster:



5. Symbolisierung

Es wurde bereits angesprochen, zur Symbolisierung dient ein CSS-Abschnitt im Header der SVG-Datei. Wenn man nun diesen Teil als eigene Datei auslagert und über einen Link einbindet, lassen sich unterschiedliche Symbolisierungen realisieren und sehr einfach auswechseln. Solch ein Zeichenschlüssel ist nach folgendem Prinzip aufgebaut:

```
.schwarz {fill:#1F1A17}
.seef {fill:#99FFFF}
.seek {stroke:#0000FF;stroke-width:1}
.FFD9D2}
.strf {stroke:#FFFFFF;stroke-width:2.5}
.hstrf {stroke:#FFFF00;stroke-width:2.5}
.strk {stroke:#1F1A17;stroke-width:4.5}
```

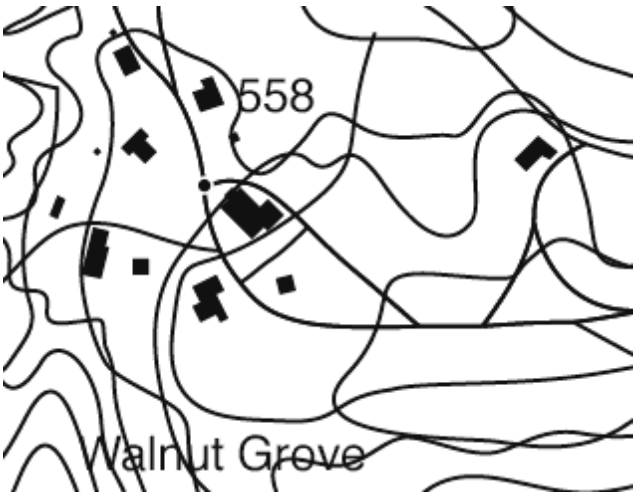
Definition

Symbolname

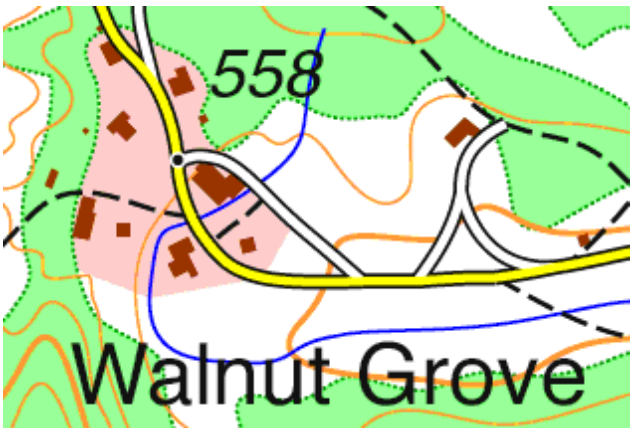
Über folgenden Link wird die Datei eingebunden:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet href="wg-map_var01.css" type="text/css"?>
```

Je nach Definition der verschiedenen Symbolnamen kann die Karte sehr unterschiedlich aussehen:



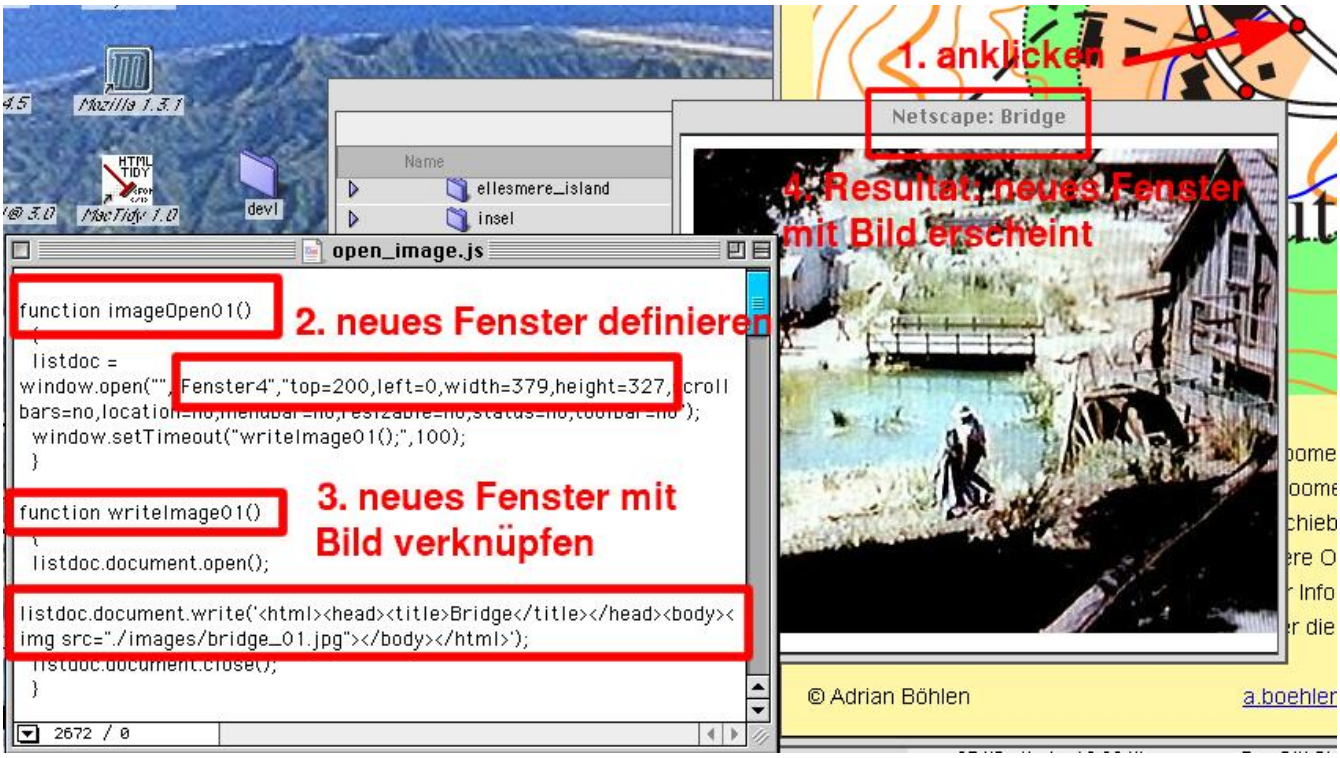
Keine besondere Symbolisierung



Mögliche Symbolisierung

6. Weitere Funktionen

Nach demselben Prinzip wie die Generierung des Informationsfensters arbeitet auch die Funktion zum Anzeigen der Bilder. Dabei lassen sich von einigen Punkten, die sich im Kartenbild einblenden lassen, Bilder aufrufen, indem man sie anklickt. Bei den Bildern handelt es sich um gewöhnliche, mit GIMP nachbearbeitete Bildschirmfotografien, um keine Copyrights zu verletzen.



---

7. Dokumentierte Probleme