Entwicklung eines Informationssystems für die Fahrradstreckenplanung

vorgelegt von

Slaven Rezić

Diplomarbeit

angefertigt unter Leitung von

Prof. Dr. Erhard Konrad

Zweiter Berichter:

Atilla Akarsular

Berlin, Oktober 1999

Slaven Rezić
Technische Universität Berlin
Matr.-Nr. 133866
Dudenstr. 24
D-10965 Berlin
E-Mail: slaven.rezic@berlin.de
http://user.cs.tu-berlin.de/~eserte

Technische Universität Berlin
Fachbereich Informatik
Wissensbasierte Systeme
Prof. Dr. Erhard Konrad
Franklinstr. 28/29
D-10587 Berlin
http://www.cs.tu-berlin.de

Zusammenfassung

Diese Diplomarbeit beschreibt die Entwicklung eines Informationssystems für die Fahrradstreckenplanung. Die Hauptaufgaben des Systems bestehen in der Bereitstellung von Informationen, die Radfahrer bei der Routensuche benötigen, sowie in der automatischen Routensuche. Im Rahmen der Entwicklung sind mehrere Anwendungen implementiert worden: ein grafisches, auf Perl/Tk basierendes System, eine WWW- und eine Konsolenschnittstelle.

Abstract

This work describes the development of an information system for planning bicycle routes. The main purpose of the system is to provide information needed by bicyclists to find a route, and is also able to suggest the best possible route. To achieve this aim a graphical Perl/Tk-based system, a web interface and a console application were implemented.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Was versteht man unter einem „Informationssystem für die Fahrradstreckenplanung“? In erster Linie ist das System ein Routenplanungssystem, das für die speziellen Bedürfnisse von Radfahrern ausgelegt ist. Systeme für die Routenplanung für Autofahrer und Benutzer des öffentlichen Verkehrs (ÖV) existieren seit vielen Jahren, sowohl als Systeme für PCs[1] als auch im Internet[2].

In diesem Punkt ist das Angebot für Radfahrer eher dürftig. Dabei hat das Fahrrad der Umweltfreundlichkeit und des geringeren Ressourcenverbrauchs wegen Systemvorteile gegenüber dem Auto im Stadtverkehr. Dieses Informationssystem sollte auch dazu beitragen, den Anteil des Radverkehrs am „Modal split“[3] in Zukunft zu steigern.

Ein System für die Fahrradstreckenplanung kann wesentlich vielschichtiger als die bestehenden Streckenplanungssysteme sein. Ist der motorisierte Individualverkehr (MIV) und insbesondere der öffentliche Verkehr relativ homogen, was Geschwindigkeiten[4] und Fahrzeuge angeht, so ist der Radverkehr sehr heterogen. Während der motorisierte Stadtverkehr konstant 50 km/h fährt, liegt die gefahrene Geschwindigkeit bei Radfahrern zwischen 10 km/h und 40 km/h. Während es bei unterschiedlichen Fahrzeugtypen von Autos kaum Unterschiede in der Reichweite gibt, kann diese bei Radfahrern stark variieren[5]. Radfahrer sind von Faktoren wie Steigungen und Straßenbeschaffenheit stärker als Autofahrer betroffen. Außerdem gibt es variable Faktoren wie das Wetter, die den Autofah-

[1] Z.B. AutoRoute Express Europa 2000 von Microsoft ([ARE99]), Fahrinfo der IVU Traffic Technologies AG ([IVU99])

[2] Z.B. Bahnauskunft ([FDB99]) der HaCon Ingenieurgesellschaft ([Hac99]) und VBB-Auskunft (Auskunft für den Verkehrsverbund Berlin-Brandenburg, [FaB99]) von IVU, Routenplaner von Falkplan ([Flk99])

[3] Der „Modal Split“ bezeichnet im Verkehrswesen die Anteile der einzelnen Verkehrsträger am Gesamtverkehr — sei es in der Anzahl der zurückgelegten Wege oder im Verhältnis der Personen- und Güterkilometer zueinander ([Lin94]).

[4] Beim MIV ist diese Aussage für den Stadtverkehr gültig.

[5] So können Radfahrer bereit sein, an einem Tag von wenigen Kilometern bis zu 200 Kilometer und mehr zurückzulegen.

rer zwar auch betreffen, den Radfahrer aber um so mehr, weil er ihm schutzlos ausgeliefert ist oder auch in seiner Fahrzeit beeinflussen kann[6].

Ein Informationssystem für Radfahrer soll diese zusätzlichen Faktoren berücksichtigen können. Einige dieser Faktoren können automatisch durch das System in die Streckenplanung einfließen, andere können als Informationen, möglichst grafisch aufbereitet, an den Benutzer weitergegeben werden, damit er — letztendlich meist ein Experte auf dem Gebiet — diese für die Routenplanung beurteilen kann.

Das Informationssystem sollte ein breites Publikum erreichen können. Deshalb ist es sinnvoll, das System offen auszulegen, so daß mehrere Implementierungsvarianten möglich sind.

Übersicht über die Struktur dieser Arbeit Im ersten Kapitel wird beschrieben, was das Informationssystem genau leisten und welche Zielgruppe damit angesprochen werden soll. Aufbauend auf diesen Voraussetzungen wird die Entscheidung für die verwendete Hard- und Software für die Implementierung begründet.

Das zweite Kapitel beschäftigt sich mit der Implementierung des Informationssystems. Es wird auf alle Aspekte der Implementierung eingegangen: die verwendete Datenbasis, die verschiedenen realisierten Anwendungen, von den Anwendungen verwendete gemeinsame Module und Hilfsprogramme, welche insbesondere bei der Datenerfassung notwendig sind.

Im letzten Kapitel wird die Implementierung bewertet und ein Ausblick auf mögliche Erweiterungen des Systems gegeben.

[6]Die Windgeschwindigkeit und -richtung kann sich sowohl negativ als auch positiv auf die Fahrzeit auswirken.

Kapitel 2

Anforderungen an das Informationssystem

2.1 Zielgruppen

Das Informationssystem richtet sich in erster Linie an Radfahrer, auch wenn einige Aspekte andere, sich individuell bewegende Gruppen wie Fußgänger und Autofahrer ebenso betreffen könnten. Räumlich ist das Informationssystem auf Berlin und Umgebung eingegrenzt, so daß es von Berlinern und Berlin-Besuchern genutzt werden kann.

Radfahrer bilden eine recht heterogene Gruppe. Die gefahrenen Geschwindigkeiten sowie die möglichen Reichweiten sind unterschiedlich. Radfahrer benutzen unterschiedliche Räder und können daher unterschiedliche Strecken bevorzugen. Z.B. werden Rennradfahrer durch ihre leichten Räder keine Probleme mit Steigungen haben, während es sich für Fahrer schwerer Hollandräder durchaus lohnen kann, einen Umweg zu fahren, um dadurch Steigungen zu vermeiden. Einen Radfahrer mit einem vollgefederten Fahrrad wird Kopfsteinpflaster weniger stören als einen ohne Federung. Radfahrer bevorzugen oder meiden Radwege, je nach Sicherheitsphilosophie, Komfortempfinden und gefahrener Geschwindigkeit. Die unterschiedlichen Standpunkte können jederzeit in der deutschen Newsgruppe zum Thema Fahrradverkehr (`<URL:news:de.rec.fahrrad>`) nachvollzogen werden.

Zusätzlich zu den verschiedenen Fähigkeiten und Eigenschaften der Radfahrer und Räder kommen unterschiedliche Motive bei der Fahrradstreckenplanung. Manch einer möchte nur die beste Route zu seiner Arbeitsstelle finden, ein Fahrradkurier möchte die schnellste Route zu seinen Auftraggebern erfahren, Radtourenfahrer möchten die Entfernung einer Route abschätzen oder sich Anregungen für eine Route holen.

Die unterschiedlichen Motive schlagen sich auch in der Art der Anwendung des Informationssystems nieder. Die Planung einer Route kann mit einem Programm auf dem heimischen Computer oder im Internet durchgeführt werden. Für die spontane Planung während der Fahrt kann höchstens ein Laptop, besser nur ein Handheld-Computer oder eine auf Mobiltelefonen aufbauende Technologie verwendet werden.

2.2 Was soll das Informationssystem leisten?

Radfahrer sind auf Straßen unterwegs. Das Informationssystem muß somit Informationen über diese Straßen liefern. Radfahrer wollen von einem Punkt zu einem anderen gelangen. Das Informationssystem muß also einen Vorschlag für die Route zwischen beiden Punkten liefern. Es gibt meist nicht nur eine mögliche Route, sondern anhand von verschiedenen Parametern mehrere Möglichkeiten. Straßen werden am besten mittels Stadtplänen dargestellt, somit sollte das Informationssystem ebenfalls diese Darstellungsform anbieten.

Die automatische Suche einer Route sollte nicht die einzige mögliche Anwendung sein. Nicht immer ist es erforderlich, die kürzeste oder beste Route zwischen zwei Punkten zu wählen. Es sollte vielmehr auch möglich sein, mit dem System Rundreisen oder Ausflüge planen zu können. Bei dieser Anwendung würde das Informationssystem Informationen über die Straßen und die Umgebung bereitstellen und der Benutzer würde selbständig eine Route auswählen.

Dem Informationssystem zusätzlich zu den Straßendaten vorliegende Daten sollte der Benutzer optimal verwenden können. Einerseits sollte die reine Ausgabe dieser Daten möglich sein, wobei sich hierfür die Darstellung von zusätzlichen grafischen Elementen auf einem Stadtplan anbietet. Diese können als zusätzliche Hilfe bei der Orientierung auf dem Stadtplan oder als Entscheidungskriterium für eine Route dienen. Anderseits sollten die Daten auch in der Routensuche verwendet werden. Wegen der oben beschriebenen Heterogenität der Zielgruppe ist es schwer, dafür allgemeingültige Formeln zu finden. Deshalb sollte es für den Benutzer möglich sein, eigene Parameter für die Suche einzugeben oder die Benutzung weiterer Daten gänzlich auszuschalten.

Andere Informationen wie U- und S-Bahnstrecken könnten in den Entscheidungsprozeß für die Streckenführung einbezogen werden. Diese Information könnte für Konzepte wie Bike+Ride verwendet werden[1].

Es sollte auch möglich sein, bereits gefahrene Strecken nachträglich zu analysieren. In diesem Fall kann man nicht die automatische Suchfunktion verwenden, weil in der Regel nicht die kürzeste Strecke gewählt wurde. Eine Analyse könnte die gefahrene Streckenlänge und den durchschnittlichen Leistungsaufwand berechnen, außerdem sollte ein Vergleich mit einer optimalen Strecke möglich sein. Die Archivierung der Daten für gefahrene Strecken wäre wünschenswert.

[1] In Zukunft könnte eine Kombination dieses Informationssystems und einer Online-Fahrplanauskunft für öffentliche Verkehrsmittel dem Benutzer mitteilen, welches der beiden Verkehrsmittel vorzuziehen ist oder ob sich eine Kombination lohnen würde.

Erforderlich wären verschiedene Sichten auf das Informationssystem. Geübte Benutzer können anhand eines angezeigten Stadtplans sehr schnell die Start- und Zielpunkte festlegen, während andere Benutzer die Endpunkte einer Route mit Straßennamen eingeben wollen. Das gleiche gilt für die Ausgabe: es sollte möglich sein, eine Route sowohl als grafisches Element im Stadtplan anzeigen zu lassen als auch als textuelle Fahrtenbeschreibung ausgeben zu können.

Das System sollte einerseits möglichst einfach und intuitiv zu bedienen sein, auf der anderen Seite sollten auch mächtigere Funktionen für den geübten Benutzer existieren.

2.3 Zielsystem

2.3.1 Verwendete Hard- und Software

Die Heterogenität der Zielgruppen spiegelt sich auch bei der Wahl des Systems, auf der das Informationssystem entwickelt werden soll, wieder. Um eine möglichst weite Verbreitung erzielen zu können, ist eine Entwicklung für das World Wide Web (WWW) sinnvoll, da nahezu für jede Rechnerplattform WWW-Browser vorhanden sind und die Zahl der Menschen, die Zugriff auf das WWW haben, immer größer wird. Allerdings haben Programme, die über das WWW laufen, auch Nachteile: Große Datenmengen können noch nicht in akzeptablen Zeiten an den Benutzer übertragen werden. Durch die Trennung zwischen Client und Server sind Antwortzeiten teilweise sehr hoch, was sich bei einem interaktiven Gebrauch negativ niederschlägt. Eine Lösung für Personalcomputer hat demgegenüber den Nachteil, daß plattformübergreifende und grafisch orientierte Programme aufwendig zu schreiben sind, sowie das Programm verteilt und installiert werden muß. Vorteilhaft ist hierbei die Möglichkeit des uneingeschränkten interaktiven Arbeitens.

Weiterhin ist es durchaus sinnvoll, das Informationssystem auf portablen Computern anzubieten. Laptops sind noch zu groß und unhandlich, um sie beim Radfahren mitzutragen. Handheld-Computer sind von der Handlichkeit günstiger, doch die darauf laufende Hardware und Software ist eingeschränkter als bei Desktop-PCs[2]. Handheld-Computer, die über ein Mobiltelefon am Internet angeschlossen sind, sowie Mobiltelefone mit eingebauten WWW-Browser können eine WWW-Implementierung des Informationssystems nutzen. Das Telekommunikationsunternehmen Mannesmann Arcor plant mit Talkingweb ein System, mit dem man per Telefon gewöhnliche HTML[3]-Seiten aus dem WWW hören und navigieren kann ([Tw99]). Eine Anpassung einer WWW-Implementierung an diese Technologie sollte in Zukunft auch möglich sein.

[2] Trotzdem ist eine Portierung für das Betriebssystem PalmOS ([Pal99]) bereits angedacht.
[3] Hypertext Markup Language ([HTM99])

2.3.2 WWW-Implementierung

Bei einer Implementierung für das WWW kann man HTML als kleinsten gemeinsamen Nenner ansehen, und zwar die Version 2.0, die alle WWW-Browser interpretieren dürften. Gängige WWW-Browser wie Netscape Navigator ([Net99]) oder Microsoft Internet Explorer (MSIE, [MSI99]) erlauben zusätzlich die Verwendung der clientseitigen Programmiersprachen Java und Javascript, sowie HTML-Erweiterungen wie DHTML[4] und CSS[5].

Da die Datenmengen recht groß sind[6], ist klar, daß die Verarbeitung auf Serverseite stattfinden sollte. Außerdem sind die auf Clientseite benutzbaren Sprachen Java und Javascript zu langsam, um eine Routensuche in vernünftiger Zeit durchführen zu können (siehe Effizienzbetrachtungen auf Seite 12 im Abschnitt 2.3.3).

Für die Wahl der Programmiersprache auf Serverseite ist man von jeglicher Einschränkung enthoben. Hierbei wurde Perl ausgewählt, einerseits weil sich Perl als CGI[7]-Sprache im WWW durchgesetzt hat, andererseits weil durch die Entscheidung für Perl/Tk als Programmiersprache für das Stand-Alone-System die Wiederverwendbarkeit von einigen Modulen und der Datenbasis möglich war. Die CGI-Version sollte, abhängig vom Browsertyp, den HTML-Code als reine Textversion (für den textorientierten WWW-Browser Lynx und andere, meist ältere Browser) bis hin zu einer Version mit einfachen DHTML-Elementen und grafischer Ein- und Ausgabe für Netscape und Internet Explorer erzeugen können.

2.3.3 Stand-Alone-Version

Bei der Wahl der Programmiersprache für die Stand-Alone-Version war die Entscheidung schwieriger. Die dominierende Betriebssystemfamilie heutzutage ist Microsoft Windows (siehe Zahlen auf Seiten 103). Daneben gibt es als wichtige Plattformen Macintosh und Unix, wobei im letzteren Fall das freie Unix-Derivat Linux immer größere Verbreitung erfährt.[8] Um möglichst zukunftssicher zu sein, bietet sich eine plattformunabhängige Lösung an. Für eine minimale Unterstützung fiel die Entscheidung auf die Plattformen Windows und Unix. Eine Unterstützung für den Apple Macintosh wäre wünschenswert, aber nicht unbedingt notwendig.

---

[4]Dynamic HTML ([DHT97])

[5]Cascading Style Sheets ([CSS99])

[6]Das Straßenverzeichnis Berlins (`data/Berlin.small.data`) ist 330 kB groß, die Koordinatendatei für das Straßennetz (`data/strassen`), welche die Minimalvoraussetzung für eine Routensuche darstellt, 160 kB

[7]CGI: Common Gateway Interface ([CGI98]), ein vom NCSA ([NCS99]) entwickelter Standard für die Erzeugung von dynamischen HTML-Seiten

[8]In [Lnx99] wird die Anzahl der Linux-Benutzer auf zwischen 5 bis 15 Millionen geschätzt. In [Lxa99] werden die Annahmen des Linux-Distributors RedHat dargestellt: 7,5 Millionen im März 1998, 10 Millionen im Oktober gleichen Jahres und 15 Millionen im April 1999.

Programmiersprachen

In der folgenden Liste werden einige Programmiersprachen aufgeführt, die sich für die Implementierung der Stand-Alone-Lösung anbieten.

C/C++: C ([Kuk89]) ist eine relativ alte, 1972 entwickelte und weit verbreitete Sprache. Es existiert praktisch für jede Computerplattform ein C-Compiler. C-Programme sind im allgemeinen schnell. Als Nachteil erweist sich, daß es keine plattformunabhängige Standardisierung für grafische Benutzeroberflächen gibt, wie sie bei der Ein-/Ausgabe über textorientierte Konsolen mit der `stdio`-Bibliothek existiert.

Abhilfe kann die Verwendung von plattformübergreifenden Toolkits bringen, die weiter unten beschrieben werden.

Java: Java ([Jas99], [Fla98]) wird in erster Linie für Applets[9] in Web-Browsern verwendet, kann aber auch als plattformübergreifende Sprache für Windows, Macintosh und Unix verwendet werden. Java besitzt mit dem Toolkit *awt* eine einheitliche grafische Benutzeroberfläche (graphical user interface, GUI).

Tcl/Tk: Tcl ([Scr99], [Rai98]) ist eine von John Ousterhout entwickelte Skriptsprache. Tk ist die dazugehörige grafische Oberfläche, die unter Windows, Macintosh und Unix lauffähig ist.

Perl/Tk: Perl ([Per99]) ist eine von Larry Wall entwickelte Skriptsprache, die vorwiegend als Ersatz für die Unix-Kommandos `grep`, `sed`, `cut`, `awk` u.a. gedacht war. Früher in erster Linie für die Systemadministration eingesetzt, hat sich die Sprache auch für CGI-Skripte im WWW etabliert. Seit der Portierung des Tk-Toolkits nach Perl ist auch die Erstellung von grafischen Benutzeroberflächen mit Perl unter Windows und Unix möglich. Eine Portierung des Tk-Toolkits auf den Macintosh ist noch nicht erfolgt. Weitere Literatur zu Perl findet sich in [Sri98], [PRK97] und [PMp99].

C++ und V-Toolkit: Das V-Toolkit ([Vtk99]) ist eine grafische Benutzeroberfläche für C++ und ist für Windows und Unix vorhanden.

C und Tk-Toolkit: Tk kann auch in Verbindung mit C verwendet werden. Damit kann die Einfachheit und Portabilität von Tk mit der Geschwindigkeit von C-Programmen kombiniert werden. Nachteilig ist, daß im Gegensatz zu Tcl/Tk-Skripten das Programm kompiliert werden muß und somit nicht „out of the box“ verfügbar ist.

[9] Applets sind Programme, für die besondere Sicherheitsbestimmungen gelten. So können Applets nicht auf Daten auf dem Rechner des Clients zugreifen. Weiterhin sind Applets für die Ausführung in Web-Browsern konzipiert, sie müssen demnach über eine grafische Benutzeroberfläche verfügen.

Kriterien für die verwendete Programmiersprache

Die Entscheidung, welche von diesen Programmiersprachen verwendet werden sollte, fällt selten objektiv. Entscheidend ist oftmals die Kenntnis einer Programmiersprache oder die Existenz von einigen wenigen Features, die andere Sprachen nicht bieten. Es ist aber trotzdem ein Versuch gemacht worden, die Sprachen nach folgenden Kriterien zu untersuchen:

Portierbarkeit: Alle oben aufgeführten Sprachen erfüllen das minimale Portierbarkeitskriterium, nämlich die Portierbarkeit nach Windows und Unix. Mit Java und Tcl/Tk ist außerdem die Portierung nach Macintosh möglich.

Verfügbarkeit: Ein Vorteil bei kompilierbaren Programmen liegt in der Unabhängigkeit von Interpretern. Bei interpretierten Programmen muß neben dem eigentlichen Programm auch der Interpreter installiert werden. Von Vorteil ist es, wenn der Interpreter bereits standardmäßig ausgeliefert wird, was meist bei Perl unter Unix der Fall ist, oder eine Integration recht leicht möglich ist, wie bei Visual Basic unter Windows.

Bei einigen interpretierten Sprachen ist die Erzeugung eines ausführbaren Programms möglich. Z.B. existiert für Perl das Tool `Perl2Exe` ([Pex99]), welches ein Perl-Skript mit dem Perl-Interpreter und allen abhängigen Modulen zu einem ausführbaren Programm zusammenbindet.

Wenn der Interpreter mit installiert werden muß, sollte die Installation einfach durchzuführen sein und möglichst mit der Installation der eigentlichen Anwendung gekoppelt sein.

Entwicklungszeit: Generell ist die Entwicklungszeit bei Compilersprachen wegen des Entwicklungsdurchlaufs Übersetzen–Binden–Ausführen/Debuggen höher als bei interpretierten Sprachen. Bei Java ist, auch wenn es eine interpretierte Sprache ist, das Übersetzen trotzdem notwendig. Perl/Tk und Tcl/Tk sind komplett interpretiert, das vollständige Parsen des Programmtexts erfolgt während des Programmstarts und ist dadurch langsamer als bei kompilierten Sprachen.

Ein weiteres Kriterium für die Entwicklungszeit ist das Bereitstellen von höheren Programmierkonstrukten. Bei interpretierten Sprachen wie Perl und Tcl sind Objekte wie Listen und assoziative Arrays enger Bestandteil der Sprache. Somit können komplizierte verschachtelte Objekte mit wenigen Zeichen dargestellt werden (im Gegensatz zu Java, wo jedes Objekt mit `new` erzeugt werden muß und die Sprache dadurch unhandlich wird).

Mit Tk existiert ein Toolkit, das die Erstellung von GUIs sehr vereinfacht. Ein minimales Perl/Tk-Programm benötigt für die Darstellung eines Hauptfensters drei Zeilen, ein minimales Tcl/Tk-Programm sogar gar keine.

Sprachmächtigkeit: Die Sprachmächtigkeit geht Hand in Hand mit der Entwicklungszeit. Je mächtigere Konstrukte eine Sprache bietet, desto weniger Zeit benötigt man für die Programmierung. Wie oben aufgeführt, ist Perl eine relativ mächtige Sprache.

Für die großen Datenmengen liegt die Verwendung einer Datenbank nahe. Das würde aber zu Lasten der Portabilität gehen, weil Datenbanksysteme in der Regel nicht portabel sind. Als Alternative stehen leichte, portable Datenbanksysteme wie Berkeley DB ([BDB99]), GDBM ([GDB99]) u.ä. oder einfache Textdateien zur Verfügung. Diese einfachen Datenbanksysteme werden durch Perl unterstützt. Die Wahl ist aber auf Textdateien gefallen, welche bei Bedarf in den Speicher geladen werden. Der Zugriff erfolgt dann über assoziative Felder (in Perl werden diese Felder „Hashes“ genannt).

Bei der Stand-Alone-Lösung soll das zentrale Element der Benutzeroberfläche eine Karte sein, die als Eingabeobjekt für die Wahl von Start- und Zielort und als Ausgabeobjekt für die Ausgabe der gefundenen Route und der zusätzlichen Information dient. Tk besitzt (als einziges der hier aufgeführten Toolkits) über ein mächtiges Canvas-Objekt. Es ist in der Lage, grafische Elemente zu zeichnen, das automatisch bei Refreshs gezeichnet wird und bei denen die einzelnen Elemente auf Events reagieren können.

Effizienz: Effizienz meint hier in erster Linie Speicherverbrauch und Ausführungsdauer. C-Programme sind natürlich bei der Ausführungszeit effizienter als interpretierte Programme. Kompilierte Programme verbrauchen meistens auch weniger Speicher als interpretierte Programme. Eine genauere Betrachtung zur Ausführungszeit und Speicherverbrauch findet man im nächsten Abschnitt.

Vergleich und Bewertung

Um die Anwendungen möglichst schnell programmieren zu können, ist die Entscheidung auf eine interpretierte Programmiersprache gefallen. Da der Umfang der Implementierung anfangs noch unklar war (das Informationssystem kann noch um beliebig viele Features erweitert werden), sollte der Zyklus vom Programmieren bis zum Austesten so klein wie möglich gehalten werden. Auch dies spricht für eine interpretierte Sprache. Weiterhin sind interpretierte Sprachen im allgemeinen portabler als kompilierte Sprachen.

Nichtdestotrotz sollte sich die Suche auch in interpretierten Sprachen in akzeptabler Zeit durchführen lassen. Deshalb wurde für einige Programmiersprachen der heuristische Suchalgorithmus A^* (für eine Beschreibung des Algorithmus siehe Seite 76 im Abschnitt 3.6.2) implementiert und verglichen.

Eine Implementierung des Algorithmus in Javascript[10] hat ergeben, daß Perl bei einer vergleichbaren Anwendung von Programmiermitteln (d.h. mit der Verwendung von assoziativen Feldern) ungefähr 30 Mal schneller ist [11]. Hinzu kommt, daß die Daten, die bei Javascript über das Internet geladen werden müßten, ungefähr eine Größe von 700K für ca. 2400 Straßendaten mit erzeugtem Straßennetz hätten.

Eine Beispielimplementierung in Java[12] hat im Gegensatz dazu eine vergleichbare Ausführgeschwindigkeit ergeben (ca. 2x bis 3x langsamer als die Perl-Variante). Die Zeit für den Aufbau des Netzes ist hierbei nicht enthalten, die bei einer Server-Variante mittels C durchgeführt werden kann. Außerdem ist auch hier das Problem mit den großen Datenmengen präsent.

Tcl ist, im Gegensatz zu der Version 7.x, seit der Version 8.0 in vielen Bereichen vergleichbar schnell. Allerdings gibt es Bereiche, die noch nicht so effizient wie bei Perl implementiert werden können (z.B. multidimensionale assoziative Felder). Bei einer Beispielimplementierung des Algorithmus A*[13] war Perl ca. 15 Mal schneller als Tcl. Der Grund hierfür ist die Verwendung von multidimensionalen assoziativen Feldern, die bei Tcl nur indirekt unterstützt werden, während bei Perl echte Referenzen und Objekte verwendet werden.

Ein weiteres Effizienzkriterium ist der Speicherverbrauch. Der Speicherverbrauch ist nicht ganz so kritisch wie die Ausführgeschwindigkeit, weil moderne Rechner heute bereits mit 64 MB Speicher oder mehr ausgeliefert werden.

Perl/Tk-Programme sind, was den grundlegenden Speicherverbrauch angeht, speicherhungriger als Tcl/Tk-Programme. Ein leeres Perl/Tk-Programm, das nur ein Fenster darstellt, verbraucht ca. 3500 kB, während ein Tcl/Tk-Programm nur 2300 kB und ein Java-Programm ca. 4000 kB verbraucht[14].

Ein wichtiges Kriterium bei der Entscheidung war die Existenz eines Canvas-Objektes. Java verfügt über ein solches Objekt nicht; eine Implementierung wäre zu zeitaufwendig gewesen. Außerdem ist bei Tcl/Tk und bei Perl/Tk das Canvas-Objekt in C implementiert; eine Implementierung in Java wäre wahrscheinlich um ein Vielfaches langsamer gewesen.

Die Entscheidung ist schließlich — der vielen Vorteile wegen — auf Perl/Tk gefallen.

---

[10] Der Javascript-Code für die Implementierung wird vom Skript `miscsrc/jscode.pl` erzeugt

[11] Es kam die Javascript-Version in den WWW-Browsern Microsoft Explorer 5.0 und Netscape 4.0 im Einsatz. Netscape 4.0 ist bei den großen Datenmengen sofort abgestürzt.

[12] Implementierung in `java/TestBBBike.java`

[13] Tcl-Implementierung in `tcl/bbbike.tcl` und `tcl/strassen.tcl`

[14] Die Werte wurden unter Linux 2.2.10 mit Perl5.00503 + Tk800.015, Tcl/Tk 8.0 und Java 1.1.7B ermittelt.

Kapitel 3

Das Informationssystem

Als Name für das Informationssystem wurde „BBBike“ gewählt. Die beiden „B“s stehen für Berlin und Brandenburg. Zwar wurde das Programm für diese beiden Bundesländer entwickelt, allerdings ist es flexibel genug, um mit Daten aus anderen Gebieten umzugehen. Die Bezeichnung „BBBike“ mit den drei aufeinanderfolgenden „B“s ist auch eine bewußte Assoziation mit der Abkkürzung WWW.

3.1 Architektur

3.1.1 Datenbasis

Die Datenbasis baut auf den Koordinaten einer Untermenge des Straßennetzes von Berlin und der Umgebung Berlins auf. Weitere Daten halten zusätzliche Attribute für die Straßen (Qualität, Vorhandensein von Radwegen). Zusätzlich zu den Koordinaten des Straßennetzes existieren Koordinaten für die Darstellung von Bahnstrecken, Gewässern, sonstigen Flächen usw.

Die Windinformation wird aus dem WWW von den Wettermeldungs-Seiten des Meteorologischen Instituts der Freien Universität (FU) Berlin ([Met99]) sowie des Deutschen Wetterdienstes (DWD, [DWD99]) geladen und ausgewertet.

Im Abschnitt zur Datenbasis (Abschnitt 3.2, Seite 15) wird genauer auf die Datenbasis und auf die Datenerfassung eingegangen.

3.1.2 Anwendungen

Es wurden drei Anwendungen implementiert:

- eine Stand-Alone-Version als Perl/Tk-Programm (`bbbike`)
- eine CGI-Version für das WWW als Perl-Programm (`bbbike.cgi`)
- eine einfache Konsolenversion (`cbbbike`)

3.1.3 Gemeinsam genutzte Module

Der Zugriff auf die Datenbasis und die Anwendung der Daten für die Suche wird in allen Anwendungen genutzt. Für diese Aufgaben wurde ein Perl-Modul entwickelt. Weiterhin gibt es Module für die Berechnung der Fahrleistung, zum Abrufen der Wetterinformation aus dem WWW sowie weitere Module, die im Abschnitt 3.6 auf Seite 71 beschrieben sind.

3.1.4 Hilfsprogramme

Für die Konsistenzerhaltung der Datenbasis sowie als Erleichterung für die Datenerfassung existieren einige kleinere Perl-Skripte.

3.2 Datenbasis

3.2.1 Struktur

Das Informationssystem basiert auf den Koordinaten für das Berliner Straßennetz. Es ist nicht das komplette Straßennetz erfaßt, weil die Suchzeit beim implementierten Suchalgorithmus exponentiell zu der Anzahl der existierenden Knoten wächst und weil auch nicht die Daten für das gesamte Straßennetz vorhanden waren.

Die Koordinaten des Straßennetzes sind zu Streckenzügen für die jeweiligen Straßen zusammengefaßt. Punkte, an denen sich Kurven oder Kreuzungen befinden, begrenzen die einzelnen Strecken. Durch diese Struktur läßt sich intern ein Straßennetz als gerichteter Graph aufbauen, wobei zwischen zwei Knoten (also Kurvenpunkte oder Kreuzungen) in der Regel zwei entgegenlaufende Kanten (für jede Fahrtrichtung eine Kante) existieren. Die Kanten enthalten als Kosten die Entfernung zwischen den beiden Knoten. Dieser Graph kann für den später beschriebenen Suchalgorithmus (siehe Seite 76) verwendet werden.

Das Straßennetz hätte auch als ungerichteter Graph aufgefaßt werden können; dies wäre aber weniger flexibel gewesen, weil es durchaus vorkommen kann, daß für die beiden Richtungen eines Straßenstücks unterschiedliche Kosten zugewiesen werden[1]. Bei Einbahnstraßen z.B. ist nur eine Richtung erlaubt, somit muß der Graph gerichtet sein.

Die Koordinaten können außerdem für die Visualisierung in einer Karte verwendet werden, sowohl für das Zeichnen des Straßennetzes als auch für das Zeichnen von Routen. Die Streckenzüge sind in der Datenbasis durch den Straßennamen gekennzeichnet; dieser kann dann für die Ausgabe (z.B. bei einer Fahrtbeschreibung) und für die Eingabe (z.B. für die Anzeige aller erfaßten Straßen) verwendet werden.

[1] Z.B. gibt es Radwege, die sich nur auf einer Straßenseite befinden.

Aus Performancegründen sind die Koordinatendateien der Straßen in verschiedene Entfernungsbereiche aufgesplittet: Berlin, nähere Umgebung (bis zu ungefähr 50 km von den Berliner Stadtgrenzen entfernt) und weitere Umgebung (mehr als 50 km von den Grenzen entfernt). Die Straßendateien können bei Bedarf zusammengefaßt und dann als ein einzelner Graph betrachtet werden; somit kann sich die Suche auf alle Entfernungsbereiche beziehen.

Zusätzlich zu den Koordinatendateien sind weitere Dateien für die Attributisierung der Straßen vorhanden. Attribute enthalten die Kategorie der Straße (Hauptstraße, Nebenstraße …), Straßenqualität (meist die Beschaffenheit der Straßenoberfläche) und das Vorhandensein von Radwegen. Diese zusätzlichen Dateien müssen strikt eine Untermenge des Straßennetzgraphen darstellen, damit eine korrekte Abbildung der Attributkanten auf die Straßenkanten erfolgen kann.

Die Daten für die Lage der Ampeln sowie die Höhenangaben enthalten nur einzelne Punkte, die mit den Knoten im Straßennetz zusammenfallen müssen.

Wegen der Forderung nach der Untermenge bei den Attributen sowie bei den Ampeln müssen neben den Kurvenpunkten und den Kreuzungen auch alle Punkte, an denen eine Attributänderung erfolgt oder eine Ampel existiert, im Straßennetzgraph als Knoten existieren.

Nicht für das Straßennetz erforderlich, aber für die Darstellung von weiteren Informationen auf der Karte interessant, sind zusätzliche Vektordateien für die Darstellung von Bahnstrecken, Gewässern, Parks etc. Da diese Dateien für den Suchalgorithmus nicht notwendig sind, gibt es auch keine weiteren Anforderungen für die Knoten und Kanten dieser Dateien.

Um sicherzugehen, daß die Attributdateien Untermengen des Straßennetzes darstellen, sind Skripte erforderlich, die dies überprüfen. Diese Skripte sind auf Seite 91 beschrieben.

Als Koordinatensystem wurde ein rechtwinkliges System mit Nullpunkt in der Nähe von Kleinmachnow gewählt[2]. Die Schrittweite beträgt einen Meter. Beim Zeichnen der Daten wird eine Umwandlungsfunktion verwendet, um die Koordinaten der Datenbasis in die Koordinaten des Tk-Canvas' umzuwandeln. Durch Änderung dieser Umwandlungsfunktion kann leicht auf andere, bereits bestehende Koordinaten umgeschaltet werden — sei es für die Portierung des Informationssystems auf andere Städte oder für die Verwendung von „offiziellen" Daten. Für den Suchalgorithmus ist die Wahl des Nullpunktes nicht entscheidend. Hier wird nur gefordert, daß die Schrittweite einen Meter betragen muß. Auf eine Korrektur der rechtwinkligen Koordinaten in polare Koordinaten wurde verzichtet; ohnehin ist der Unterschied im begrenzten Berliner Gebiet vernachlässigbar.

Die unvollständige Datenbasis der Straßen ist problematisch, wenn die Eingabe von Start- und Zielpunkt nicht durch Klick auf die Karte, sondern durch

[2]Kleinmachnow liegt südwestlich von Berlin. Die Wahl des Nullpunktes auf diese Stelle fiel nach Untersuchungen der Fahrinfo-Daten von Hafas, die in einem binären Format vorliegen. Die Fahrinfo-Daten wurden allerdings *nicht* für dieses Informationssystem benutzt.

textuelle Eingabe geschieht. Als provisorische Lösung wird die STRADA-Datei[3] der Deutschen Post verwendet, die eine Liste aller Straßennamen Berlins sowie die dazugehörige Postleitzahl enthält. Für nicht in der Datenbasis enthaltene Straßen kann anhand dieser Datei eine Liste der sich im Nahbereich befindlichen Straßen erzeugen, die zugleich Teil der Datenbasis sind. Der Nahbereich wird anhand der Postleitzahl festgestellt.

Die bislang beschriebenen Dateien sind statisch und werden mit der Anwendung mitgeliefert. Es besteht aber die Möglichkeit, geänderte Dateien über das Internet zu aktualisieren.

Die Wetterinformation ist dynamisch und wird aus dem WWW bei Bedarf geladen. Auf Seite 85 wird die Prozedur dazu genauer beschrieben.

3.2.2 Aspekte zur Implementierung

Bei der Implementierung der Datenbasis müssen zwei Aspekte beachtet werden:

- Wie werden die Daten persistent gehalten?
- Wie werden die Daten in der Anwendung benutzt?

Die Anwendung sollte auf die Daten möglichst schnell zugreifen können, da der Suchalgorithmus unter Umständen mehrere tausend Knoten expandieren muß. Die Daten können innerhalb der Anwendung im Speicher als Perl-Datenstrukturen (Hashes und Arrays) oder außerhalb der Anwendung in einem Datenbanksystem gehalten werden.

Für die Entscheidung sollten Effizienzkriterien wie Geschwindigkeit und Speichernutzung, aber auch Kriterien wie die notwendige Verwendung von externen Programmen oder zusätzlichen Perl-Modulen und die Anwendung des darunterliegenden Systems bei der Dateierfassung einbezogen werden.

Bei den Daten muß zwischen den Quelldaten (extensionale Datenbasis) und den zur Laufzeit generierten Daten (intensionale Datenbasis) unterschieden werden. Die Quelldaten sind die Koordinatendateien der Straßen sowie die dazugehörigen Attributdateien. Die Quelldaten dienen zur Ausgabe der Straßen in der GUI und zur Erzeugung einer gültigen Auswahl bei der Eingabe der Straßennamen. Außerdem wird aus diesen Daten der Graph des Straßennetzes erzeugt, auf dem der Suchalgorithmus basiert.

Die Wahl der Datenrepräsentation ist für die Anwendung nicht essentiell. Wie aus den Beispielen für die folgenden Benchmarks ersichtlich, kann mit kleinen Änderungen am Programmcode eine andere Repräsentation verwendet werden[4].

[3] Diese Datei (zu finden in [PLZ99]) wurde bei der Umstellung der Postleitzahlen in Deutschland von vier Ziffern auf fünf Ziffern vertrieben. Die Datei wurde leicht modifiziert, wobei das Format von Datensätzen mit festen Längen in ein Format mit Trennzeichen überführt wurde. Nicht relevante Datenfelder wurden bei der Überführung gelöscht.

[4] Grundsätzlich können beliebige Datenrepräsentation in Perl über den TIEHASH-Mechanismus in einen Perl-Hash überführt werden.

Bei den Benchmark-Tests wurde der Suchalgorithmus mit gewöhnlichen Perl-Hashes, mit einer Kombination aus `DB_File` und `MLDBM`[5] und mit einer relationalen Datenbank (`MySQL 3.22.25` mit dem Perl-Interface `DBI 1.13` sowie `DBD::mysql 2.04`) untersucht[6]. Dabei wurde festgestellt, daß die Lösung mit Perl-Hashes etwa 3x schneller sowohl als die `MLDBM`-Lösung als auch die `MySQL`-Lösung war.

Ein Vorteil bei der Verwendung von Datenbanksystemen ist, daß die Daten sofort zur Verfügung stehen. Bei der Verwendung von Perl-Hashes müssen sich die Daten im Hauptspeicher befinden. Es ist also ein Laden der Daten notwendig. Bei Datenbanksystemen ist weiterhin vorteilhaft, daß die Daten für den Graphen nicht permanent im Speicher gehalten werden müssen. Bei der Verwendung von Perl-Hashes für den Graphen des Straßennetzes mit ca. 2400 Straßen werden ungefähr 4 MB Speicher verbraucht.

Der Graph des Straßennetzes wird aus den Koordinatendateien erzeugt. Dieses Erzeugen kann zur Laufzeit geschehen oder vor Auslieferung des Programms vorgenommen werden. Bei Datenbanksystemen wie `MySQL` ist das Einfügen mit größeren Overhead verbunden, so daß sich hier letzteres anbietet. Bei der Verwendung von Perl-Hashes sind beide Lösungen möglich. Um die Größe der auslieferbaren Distribution gering zu halten, wird derzeit das Erzeugen zur Laufzeit durchgeführt. Im Abschnitt 3.8.1 auf Seite 93 wird beschrieben, wie die Zeit für diese Generierung minimiert werden kann.

Für die persistente Speicherung der Koordinatendaten können einfache Textdateien oder Datenbanksysteme verwendet werden. Der Geschwindigkeitsvorteil zusammen mit der Tatsache, daß keine zusätzlichen Perl-Module bzw. externe Programme notwendig sind, haben zum Entschluß für die Verwendung von einfachen Textdateien für die Koordinatendateien sowie die Konvertierung in Perl-Hashes während der Laufzeit für den Straßennetz-Graphen geführt.

3.2.3 Implementierung

Die Koordinatendateien enthalten den Namen eines Kartenobjekts (Straße, Bahnhof ...) und die zugehörigen Koordinaten. Das gewählte Dateiformat erlaubt es nicht, Attribute direkt den Koordinaten der Strecken zuzuordnen; vielmehr gibt es weitere Dateien, die ein Attribut und eine eigene Koordinatenliste enthalten[7].

[5] Mit dieser Kombination können Perl-Objekte in eine Flat-File-Datenbank geschrieben werden.

[6] Das Testen des Geschwindigkeitsunterschieds zwischen Perl-Hashes und den Zugriff auf `MySQL` wurde in `diplom/code/BBBikeDiplom.pm` durchgeführt; zwischen Perl-Hashes und `MLDBM` in `Strassen.pm`.

[7] Attribute in die Streckendaten einzubinden ist problematisch, weil Attribute meist nur für eine Teilstrecke gelten. So ist es z.B. möglich, daß nur ein Teil einer Straße eine Kopfsteinpflasterstrecke enthält. Um das Datenformat einfach zu halten, wurde der Weg der getrennten Dateien mit dem gleichen Format gegangen. Dadurch ist aber die Konsistenzerhaltung schwieri-

Alle Programmdaten befinden sich im Unterverzeichnis `data`. Die Koordinaten für das Berliner Straßennetz befinden sich in der Datei `strassen`. Das Format dieser Datei (wie auch der meisten anderen Dateien in diesem Verzeichnis) sieht wie folgt aus:

```
Name TAB Kategorie SPACE X1,Y1 SPACE X2,Y2 ...
```

Dabei wird in der Datei `strassen` das Feld „Name" für den Straßennamen verwendet, das Feld „Kategorie" für die Straßenkategorie:

`B`: Bundesstraßen außerhalb von Berlin

`HH`: wichtige Hauptstraßen

`H`: Hauptstraßen

`N`: Nebenstraßen

`NN`: Nebenstraßen, die für der Kfz-Verkehr gesperrt sind

Die Kategorie „Bundesstraßen" wird nur für Straßen außerhalb der Berliner Stadtgrenzen verwendet. Hauptstraßen sind Straßen, die in gängigen Stadtplänen so bezeichnet sind. Wichtige Hauptstraßen sind subjektiv ausgewählte Hauptstraßen mit hohem Verkehrsaufkommen (meistens Bundesstraßen innerhalb der Berliner Stadtgrenzen und andere wichtige Ausfallstraßen).

Es befinden sich noch weitere Koordinatendateien im selben Verzeichnis:

`ampeln`: Koordinaten für Ampeln im Berliner Stadtgebiet. Dabei bleibt das Namensfeld unbenutzt; als Kategorien wird „X" (unbestimmt), „F" (reine Fußgängerampel) und „B" (beschrankter Bahnübergang) verwendet.

`berlin`: Koordinaten der Berliner Stadtgrenze. Die Grenze wird nur gebraucht, um sie für die bessere Orientierung zeichnen zu können.

`faehren`: Fähren in Berlin und Brandenburg.

`flaechen`: Flächen, z.B. Parks, Wälder und Flughäfen. Derzeit werden Parks und Wälder mit der Kategorie „F:P" gekennzeichnet und sonstige Flächen mit „F:farbe". Das „F:" steht dabei für Fläche.

`gesperrt`: Diese Datei enthält Angaben zu temporären und permanenten Sperrungen. Im Namens-Feld befindet sich eine textuelle Beschreibung zur Art der Sperrung bzw. der Name des Straßenzuges, für den die Sperrung gilt. Im Kategorie-Feld können die Werte „0", „1" oder „2" stehen. Dabei steht „2" für eine totale Sperrung in beide Fahrtrichtungen, „1" für Einbahnstraßen (der Koordinatenzug dahinter gibt die Richtung der Sperrung an) und

---

ger, da Änderungen in der Stammdatei meist Änderungen in der Attributdatei erfordern. Dafür sind kleine Konsistenzprüfprogramme geschrieben worden.

„0“ steht für einzelne Punkte, die genaugenommen keine Sperrung darstellen, sondern wo der Radfahrer gezwungen ist, sein Fahrrad zu tragen (etwa eine Treppe). Wenn es sich bei einer Straße um eine unechte Einbahnstraße handelt, d.h. Radfahrer dürfen die Straße in Gegenrichtung benutzen, Autofahrer aber nicht, dann wird die Straße nicht als Einbahnstraße in der Datenbank gekennzeichnet. Anderseits sind Straßen, die für Radfahrer gesperrt sind, entweder nicht eingezeichnet (z.B. die Stadtautobahnen und Kraftfahrzeugstraßen Berlins) oder als völlig gesperrte Straße gekennzeichnet. Gesperrte Straßen sind auch dort markiert, wo Straßen durch Baustellen zeitweise nicht passierbar sind (wie zur Zeit am Potsdamer Platz).

Noch nicht in dieser Datei implementiert ist die Darstellung verbotener Abbiegevorgänge. Diese Sperrart könnte durch drei Koordinatenpunkte dargestellt werden, wobei mit dem mittleren der Punkt der Sperrung und mit den beiden anderen die Fahrtrichtung angegeben wird.

Durch diese Datei wird der Graph des Straßennetzes so geändert, daß die gesperrten Straßen völlig gelöscht werden (beide Kanten werden entfernt), bei Einbahnstraßen wird die Kante der gesperrten Richtung entfernt. Bei den Punkten, an denen getragen werden muß, kann der Benutzer im Programm entscheiden, ob diese Stellen auch aus dem Graphen entfernt werden sollen. In diesem Fall werden alle Kanten, die von diesem Knoten ausgehen, gelöscht.

`hoehe`: Höhenpunkte in Berlin und Brandenburg. Die meisten dieser Höhenangaben stammen aus topographischen Übersichtskarten von Berlin im Maßstab 1:5000 sowie topographischen Karten im Maßstab 1:1000. In dieser Datei besteht das Namensfeld aus der Höhe über Normalnull (NN). Die Höhenpunkte werden zur Berechnung von Steigungen und Gefällen verwendet.

`landstrassen`: Straßen außerhalb Berlins. Genaugenommen sind es nicht nur Landstraßen, da auch Stadtstraßen, wie z.B. diejenigen in Potsdam, auch in dieser Datei enthalten sind.

`orte`: Koordinaten von Orten außerhalb Berlins. Das Namensfeld enthält den Namen des Ortes. Die Kategorisierung erfolgt mit den Ziffern „0“ bis „5“ und stellt eine subjektive Einteilung der Wichtigkeit oder Größe der Ortschaft dar. „0“ sind meistens Teile von Ortschaften oder Bezirke von Städten, während „5“ für größere Städte in der Region wie Potsdam oder Frankfurt/Oder vorbehalten ist.

`plaetze`: Diese Datei ist als Zusatz für die Datei `strassen` für die WWW-Version gedacht. Somit können auch Namen von Plätzen als Start- und

Zielort eingegeben werden. Diese Datei wurde nach der Analyse der WWW-Logdateien angelegt, als klar wurde, daß neben Straßennamen häufig auch Plätzenamen eingegeben wurden.

`qualitaet_s`, `qualitaet_l`: Diese Dateien beinhalten Daten über die Straßenbeschaffenheit. Um die Problematik einer exakten Kategorisierung zu umgehen, wurde eine Kombination aus wenigen Kategorien und einer zusätzlichen textuellen Beschreibung gewählt. So beinhaltet das Namensfeld neben dem Straßennamen eine Beschreibung zu der Straßenqualität (z.B. „mäßiges Kopfsteinpflaster“). Das Kategoriefeld besteht aus Werten von „Q0“ bis „Q3“, wobei „Q0“ für gutes Pflaster steht, auf dem ohne Einschränkungen schnell gefahren werden kann, ohne daß Komforteinbußen entstehen. „Q1“ ist meist gutes Kopfsteinpflaster oder mäßiger bis schlechter Asphalt, auf dem ca. 22–28 km/h schnell gefahren werden kann. „Q2“ steht für mäßiges Kopfsteinpflaster und „Q3“ für sehr schlechten Belag. Die maximalen Geschwindigkeiten liegen bei etwa 15 bis 22 km/h bei „Q2“ und maximal 15 km/h bei „Q3“. Im Programm können die maximalen Geschwindigkeiten individuell geändert werden. In der Datenbank wurde auf die Erfassung von Straßen der Qualitätsstufe „Q0“ weitgehend verzichtet, weil es sich hierbei um die meisten Straßen in Berlin und im Umland handelt. Nur bei ungewöhnlichen Kombinationen (z.B. sehr gutes Kopfsteinpflaster der Stufe „Q0“) sind solche Qualitätsstufen aufgeführt.

`radwege`: Hier sind Radwege, Busspuren und verkehrsberuhigte Bereiche dargestellt. Da Radwege und Busspuren oft nur in einer Fahrtrichtung anzutreffen sind, gelten die Koordinatenzüge nur für die jeweilige Fahrtrichtung. Im Kategorienfeld werden folgende Werte verwendet:

- `RW0`: Radweg
- `RW1`: benutzungspflichtiger Radweg, d.h. ein Radweg mit StVO[8]-Zeichen Nr. 237
- `RW3`: Radpur, d.h. ein für Radfahrer abgetrennter Streifen auf der Fahrbahn
- `RW4`: Busspur, meist für Radfahrer freigegeben
- `RW5`: verkehrsberuhigte Zone (Schrittgeschwindigkeit)

`rbahn`, `sbahn`, `ubahn`, `rbahnhof`, `sbahnhof`, `ubahnhof`: Zur besseren Orientierung auf der Karte, aber auch als Entscheidungshilfe für den eventuellen Einsatz von Bike+Ride, können Regional-, S- und U-Bahnlinien mit Bahnhöfen gezeichnet werden. Das Namensfeld bezeichnet dabei die Linienbezeichnung bzw. den Bahnhofsnamen. Das Kategoriefeld besteht aus zwei

[8] Straßenverkehrsordnung

Buchstaben, wobei der erste für die Bahnart („R“ für R-Bahn, „S“ für S-Bahn und „U“ für U-Bahn) steht und der zweite für die Berliner VBB-Zone (A, B, C oder Gebiete außerhalb des Berliner Verkehrsgebiets), in der sich der Linienabschnitt bzw. Bahnhof befindet.

`wasserstrassen, wasserumland`: Diese beiden Dateien enthalten Gewässer in Berlin (erste Datei) und außerhalb Berlins (zweite Datei). Als Kategorien werden „F:W“ für Seen, „W“ für Bäche, „W1“ für Flüsse und „W2“ für Ströme verwendet. Daneben gibt es noch die Kategorie „F:I“ für Inseln innerhalb von Seen.

3.2.4 Datenerfassung

In dieser Arbeit wurde auf die Datenerfassung kein besonderes Augenmerk gegeben. Idealerweise sollten die Koordinaten der Straßen von einem Vermessungsamt o.ä. geliefert werden. Da diese Daten nicht zur Verfügung standen, wurden die Koordinaten manuell aus Stadtplänen erfaßt. Da die Koordinatendateien Textdateien sind, erfolgt die Eingabe der Koordinaten einfach über einen Editor.

Beim Editieren von Attributen kann auf die vorhandenen Straßenkoordinaten zurückgegriffen werden. Dafür kann in der Perl/Tk-Anwendung in den Editiermodus umgeschaltet werden. Ein Klick auf einen Punkt auf der Karte führt dazu, daß die Koordinaten in die X11-Selection[9] geschrieben werden. Mit der nächsten Einfügeoperation im Editor können diese Koordinaten in die Koordinatendatei eingefügt werden.

Da bei dieser Vorgehensweise Fehler entstehen können und somit zu einer inkonsistenten Datenbasis führen, sollte nach dem Editieren die Konsistenz mit den auf Seite 91 beschriebenen Hilfsprogrammen überprüft werden.

3.3 Perl/Tk-Anwendung

Die Kernanwendung des Informationssystems bildet die Perl/Tk-Schnittstelle. Ein Vorteil gegenüber der später beschriebenen WWW-Schnittstelle liegt in der höheren Interaktivität, die durch die fehlende Client-Server-Verbindung implementiert werden konnte. Durch das Darstellen einer Karte können Routenendpunkte sowohl grafisch als auch textuell eingegeben werden. Das Ergebnis der Route kann sofort als Strecke auf der Karte oder als Fahrtenbuch dargestellt werden.

In den folgenden Abschnitten wird zunächst die grafische Oberfläche aus Benutzersicht und dann die Implementierung des Programms beschrieben.

[9] Die X11-Selection ist ein Puffer des Fenstersystems, der allen Anwendungen zur Verfügung steht ([Jon91]).

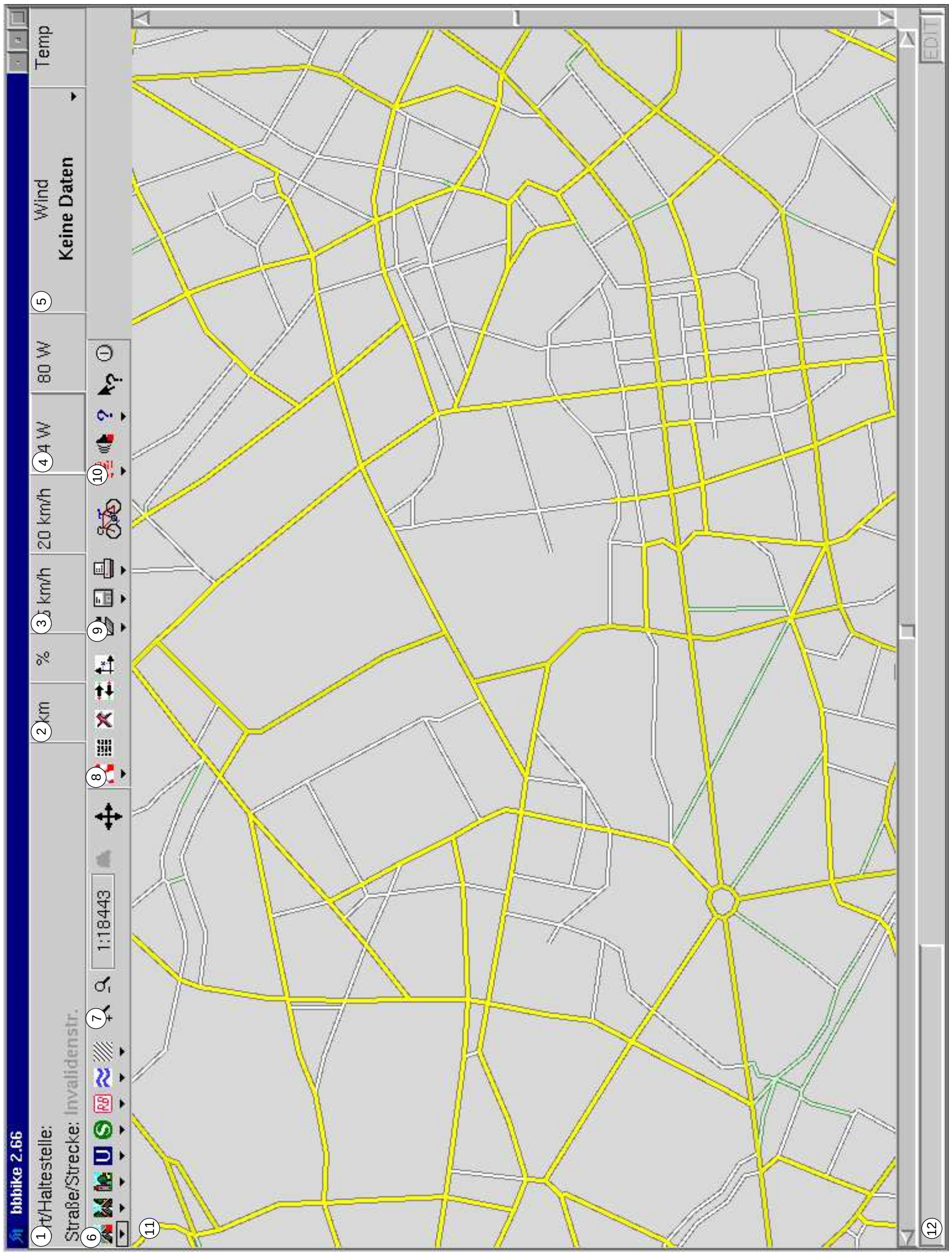

Abbildung 3.1: BBBike

3.3.1 Beschreibung der Benutzeroberfläche

Die grafische Anwendung kann auf der Abbildung 3.1 eingesehen werden. Um im folgenden Text besser auf die Bereiche des Screenshots verweisen zu können, sind

Markierungen (z.B. ①) eingetragen.

Der größte Teil des Fensters wird durch den Platz für die Darstellung der Karte eingenommen ⑪. Unter der Karte ist eine Leiste für Statusinformationen ⑫: eine Progressbar[10] und Platz für Statusmeldungen, die nicht in gesonderten Dialogfenstern angezeigt werden. Über der Karte befindet sich eine Iconleiste ⑥, die die Funktion von Menüs übernimmt. Ganz oben ist weiterer Platz für Statusanzeigen ①, die mit der Karten- und Routeninformation verbunden sind. Im folgenden Text wird die Funktionalität der grafischen Objekte genauer beschrieben.

Im oberen Teil des Fensters befinden sich die verschiedenen Statusangaben. Links wird neben den Beschriftungen „Ort/Haltestelle" und „Straße/Strecke" ① der Name des sich unter dem Mauscursor befindlichen Kartenobjekts angezeigt. Kartenobjekte, die durch einen Punkt dargestellt werden, z.B. Ortsnamen, U- und S-Bahnhöfe, werden dabei im oberen der beiden Felder, Strecken und Flächen wie Straßen, Bahnlinien und Gewässer im unteren dargestellt.

Verläßt die Maus ein Kartenobjekt, so wird der Name aus dem Feld nicht gelöscht, sondern in einem hellen Grau dargestellt. So wird der Benutzer nicht gezwungen, genau auf die manchmal sehr kleinen Straßenobjekte positionieren zu müssen.

Mit Klick auf die Beschriftungen wird ein Dialogfenster (siehe Abbildung 3.2 auf Seite 27) geöffnet, in dem eine vorhandene Straße bzw. ein Ort direkt eingegeben werden kann. Die Karte wird daraufhin so verschoben, daß sich diese Straße oder dieser Ort in der Mitte des Kartenausschnitts befindet.

Daneben ist das Feld zur Anzeige der Länge der aktuellen Route ②. Voreingestellt ist die Ausgabe in Kilometern; mit Klick auf die rechte Maustaste über der „km"-Beschriftung kann zur Ausgabe in Metern umgeschaltet werden.

Das „%"-Feld gibt an, um wieviel Prozent die gefundene Route von der Luftlinie abweicht. Bei offensichtlichen Rundreisen (wenn die Abweichung mehr als 1000% betragen würde) wird diese Darstellung ausgeblendet.

Die nächsten Felder ③ zeigen die Fahrzeit für eine angegebene Durchschnittsgeschwindigkeit an. Voreingestellt werden die Geschwindigkeiten „15 km/h" und „20 km/h" angezeigt. Mit Klick auf die linke Maustaste kann eine andere Geschwindigkeit gewählt werden. Mit der rechten Maustaste kann eine Geschwindigkeit als „Referenzgeschwindigkeit" für den Optimierungsparameter (siehe Seite 33) ausgewählt werden. Die Referenzgeschwindigkeit ist mit einem erhöhten Rand dargestellt. Die Anzahl der Geschwindigkeitsfelder ist variabel und hängt von der Größe des Applikationsfenster und von Kommandozeilenoptionen ab.

Daneben befinden sich die Felder für die Anzeige der Fahrzeit anhand von Durchschnittsleistungen ④. Beim Starten der Anwendung wird über das BikePower-Modul für die Geschwindigkeitsfelder die entsprechende Leistung bei Windstille und ebener Strecke berechnet. Die Leistung kann mit Klick auf die

[10] Grafisches Element, das den Fortschritt einer länger andauernden Aktion anzeigt.

linke Maustaste geändert werden. Auch hier kann mit der rechten Maustaste eine „Referenzleistung“ eingestellt werden. Diese Leistung wird zu einer Geschwindigkeit umgerechnet und dann genau wie die Referenzgeschwindigkeit verwendet.

Im nächstes Feld erfolgt die Anzeige der Winddaten ⑤. Im oberen Teil wird das Datum und die Uhrzeit der Winddaten angegeben, falls sie aus dem Internet entstammen. Im unteren Teil wird die Windrichtung und die Windgeschwindigkeit in m/s ausgegeben. Wenn keine Verbindung zum Internet hergestellt werden kann, erscheint der Text „Keine Daten“. Mit einem Klick auf den oberen Teil versucht das Programm, die Angaben zu aktualisieren. Mit einem Klick auf den unteren Teil wird ein Dialogfenster geöffnet, in dem man selbständig die Winddaten eingeben kann.

Die rechte Maustaste öffnet ein Menü mit weiteren Optionen. So kann angegeben werden, von welcher Quelle (in diesem Fall meint „Quelle“ die Wetterstation und den Web-Server) die Winddaten kommen sollen. In der Voreinstellung werden intern die Daten aller Quellen eingeholt und die aktuellste ausgegeben. Es ist aber auch möglich, nur die Daten der FU Berlin oder des DWD auszuwerten.

Weiterhin kann die Quelle der Daten (hier ist Quelle als „Medium“ gemeint) angegeben werden. Dies wird normalerweise das WWW sein, aber für den Offline-Betrieb kann — falls vorhanden — die HTML-Datei aus einem Cache verwenden werden (Einstellung „lokaler Cache“) oder aus einer Wetterdatenbank. Im Fachbereichsnetz befindet sich diese Wetterdatenbank im Verzeichnis `/home/e/eserte/doc/met`.

Bei der Existenz dieser Wetterdatenbank können die Wetterdaten eines beliebigen Datums unter „Auswahl aus Datenbank“ ausgewählt werden. Auch hier kann die Wahl aus verschiedenen Wetterstationen getroffen werden.

Die weiteren Optionen im Menü sind:

Wind ignorieren: Hiermit wird das Färben von Routen anhand der Windrichtung unterdrückt. Dies ist sinnvoll, wenn z.B. eine Karte mit einer Route als Bilddatei erzeugt werden soll, bei der eine Färbung eher störend wäre.

Aktualisierung: Hiermit werden die Wetterdaten aus der externen Datenquelle geholt.

Automatische Aktualisierung: Wenn die Wetterstation geändert wurde, werden bei aktiviertem Knopf die Wetterdaten automatisch aktualisiert. Dieser Knopf sollte deaktiviert bleiben, falls man nicht permanent online ist oder eine langsame Verbindung ins Internet hat.

Automatische Routenaktualisierung: Bei Änderung der Wetterdaten wird die Route entsprechend der neuen Windgeschwindigkeit neu eingefärbt.

Im letzten Feld der oberen Leiste wird die Temperatur ausgegeben. Diese Angabe ist rein informativ und wird ansonsten in der Applikation nicht verwendet[11].

[11] Genau genommen geht die Temperatur in die Leistungsberechnung von BikePower ein, aber

Wenn das Applikationsfenster verkleinert wird, werden nacheinander Felder aus der Anzeige herausgenommen. Zunächst wird die Anzahl der Geschwindigkeits- und Leistungsfelder reduziert, dann das Temperaturfeld und nacheinander alle anderen Felder von rechts entfernt.

In der nächsten Reihe befinden sich die Icons, mit denen die Applikation gesteuert wird 6. Auf eine klassische Menüstruktur (eine Menüleiste mit den Einträgen „Datei" und „Bearbeiten") wurde verzichtet, weil diese sich primär an Dokumentenbearbeitung orientiert. In BBBike können zwar Routen auch abgespeichert und geladen werden, aber diese Funktionalität ist eher unwichtig. Standardmenüeinträge unter „Bearbeiten" wie „Ausschneiden", „Kopieren" und „Einfügen" machen ebenfalls nicht viel Sinn.

Normalerweise befinden sich die Icons in einer Reihe; wenn das Applikationsfenster zu klein ist, werden die Icons in zwei Reihen verteilt.

Zeichnen der Kartenobjekte Mit dem Icon links außen 6 kann die Darstellung der Straßen im Berliner Stadtgebiet ein- und ausgeschaltet werden. In der Voreinstellung werden die Straßen beim Start immer gezeichnet. Mit Klick auf die rechte Maustaste oder auf das Dreieck unter dem Icon wird ein Menü aufgeklappt. Der Aufbau dieses Menüs entspricht ungefähr auch dem Aufbau der folgenden Menüs.

Allgemein befindet sich in den Menüs ganz oben ein Auswahlknopf, mit dem die Anzeige des jeweiligen Kartenobjekts an- und ausgeschaltet werden kann (im ersten Menü „Straßen"). Dieser Knopf hat also die gleiche Funktionalität wie das entsprechende Icon. Mit „Straße auswählen" wird ein Dialogfenster (siehe Abbildung 3.2) geöffnet, in dem eine Straße aus der Liste aller Straßen ausgewählt werden kann. In allen Menüs befinden sich unten drei Menüeinträge „oben zeichnen", „normal" und „unten zeichnen". Damit kann die Schichtung der gezeichneten Kartenobjekte geändert werden. Wenn „oben zeichnen" ausgewählt ist, werden die Straßen über allen anderen Kartenobjekten gezeichnet. Bei „unten zeichnen" werden sie unter allen anderen Kartenobjekten gezeichnet. Bei „normal" werden die Straßen wieder in der normalen Schichtung gezeichnet. Es gibt auch einen Schichteneditor[12], mit dem die Schichtung ähnlich wie bei den Programmen Gimp ([Gim99]) und Photoshop geändert werden kann.

die Auswirkungen auf die Leistung sind minimal.

[12]Der Schichteneditor kann mit Druck auf die Taste `X` erreicht werden.

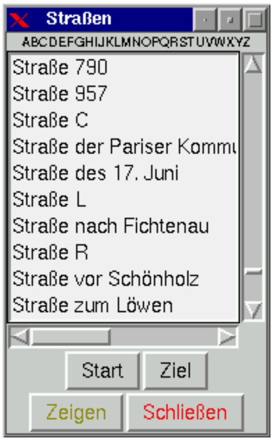

Abbildung 3.2: Dialogfenster für die Straßenauswahl

Im „Straßen"-Menü befinden sich weitere Menüpunkte mit folgende Funktionen:

Auswahl aus...: Hiermit wird ein Untermenü aufgeklappt, mit dem man weitere Möglichkeiten zur Auswahl einer Straße hat. Die PLZ-Datenbank enthält eine Liste aller Berliner Straßen, allerdings hat diese Datenbank keine Koordinaten, sondern enthält nur eine Zuordnung von Straßen zu Postleitzahlen. Diese Information kann verwendet werden, um den ungefähren Bereich der Straße zu markieren[13].

Mit den beiden Menüpunkten Telefonbuch-Datenbank (Straße und Name) kann bei einer eingelegten Telefonbuch-CD-ROM von 1998 oder 1999 auch eine Straße mit Hausnummer bzw. ein Name ausgewählt werden. Da die Datenbank exakte Koordinaten enthält, kann die gefundene Straße auch markiert werden.

Zusätzlich zur Markierung kann bei allen Auswahlmöglichkeiten festgelegt werden, ob die Straße als Start- oder Zielort festgelegt werden soll. Mit dieser Methode kann man eine Suche starten, ohne Punkte auf der Karte mit der Maus festgelegt zu haben.

Straßenkategorien: Im Untermenü hat man die Möglichkeit, einzelne Straßenkategorien von der Darstellung und auch von der Suche auszuschließen. Die verwendeten Straßenkategorien sind: wichtige Hauptstraßen, Hauptstraßen, Nebenstraßen und für den motorisierten Verkehr gesperrte Straßen. Zum Beispiel kann letzterer Punkt deselektiert werden, wenn eine Auto-Route gesucht werden soll.

[13] Dabei werden alle in der BBBike-Datenbank enthaltenen Straßen, die die gleiche Postleitzahl haben, markiert.

Höhenpunkte: Mit diesem Punkt kann das Zeichnen von Höhenpunkten samt der Höhenangabe in Metern ein- und ausgeschaltet werden. Höhenpunkte werden zur Berechnung von Steigungen und Gefällen verwendet.

Radwege: Hier wird das Zeichnen von Radwegen, Busspuren und verkehrsberuhigten Bereichen eingeschaltet. Da die Art der Radwege in beiden Richtungen unterschiedlich sein können, werden die Radwege an den Rändern der Straßen gezeichnet. Bei den farblichen Markierungen bedeutet dunkelblau, daß es sich um einen benutzungspflichtigen Radweg handelt; blau, daß es ein nicht benutzungspflichtiger Radweg ist; hellgrün sind Radstreifen und Suggestivstreifen auf der Fahrbahn; orange für Busspuren, die auch für Radfahrer zugelassen sind und olivgrün für verkehrsberuhigte Bereiche.

Einbahn-/gesperrte Straßen: Einbahnstraßen werden mit blauen Pfeilen dargestellt, die die erlaubte Richtung anzeigen. Gesperrte Straßen werden mit roten Kreuzen dargestellt. Stellen, an denen Radfahrer das Fahrrad tragen müssen, werden mit einem roten Punkt markiert.

Fähren: Hiermit werden Fähren als gepunktete weiße Linien eingezeichnet.

Ampeln: Schaltet das Zeichnen von Ampeln ein. Ampeln werden durch ein kleines Ampelsymbol dargestellt. Zusätzlich werden beschrankte Bahnübergänge durch Andreaskreuze markiert.

Outline zeichnen: Normalerweise werden Straßen als Linien ohne Rand dargestellt, wohingegen Stadtpläne Straßen mit schwarzen Umrandungslinien darstellen. Mit dieser Option kann diese optisch ansprechendere Darstellungsweise gewählt werden, die dafür langsamer beim initialen Zeichnen ist und sich auch etwas langsamer beim Scrolling verhält.

Straßennamen: Mit dieser (experimentellen) Option werden Hauptstraßen mit Straßennamen versehen. Das Zeichnen der Straßennamen ist recht langsam und ressourcenverschwenderisch und deshalb in der Voreinstellung ausgeschaltet. Außerdem funktioniert das Zeichen mit gedrehten Zeichensätzen nur auf X11R6-Servern, aber nicht auf älteren X-Servern oder auf Windows-Rechnern.

Straßenqualität: Zeichnet die Straßenqualität in den Stadtplan ein. Die Qualität wird mit gestrichelten Linien dargestellt, und zwar in folgenden Farben:

dunkelgrün: Sehr gute Qualität, die meisten Asphaltstraßen

grün: Gute Qualität, z.B. mäßige Asphaltstraßen oder gutes Kopfsteinpflaster

orange: Mäßige Qualität, z.B. gewöhnliches Kopfsteinpflaster rot: Schlechte Qualität, z.B. schlechtes Kopfsteinpflaster

Qualitätskategorien: In diesem Untermenü wird angegeben, welche Qualitätsstufen beim selektierten Qualitätskategorien-Knopf gezeichnet werden sollen. Bei der Voreinstellung werden alle Stufen bis auf „Q0“ (sehr gut) gezeichnet.

Mit dem nächsten Icon kann das Zeichnen von Landstraßen ein- und ausgeschaltet werden. Mit Landstraßen sind hier alle Straßen außerhalb von Berlin gemeint, auch wenn z.B. die Potsdamer Straßen Stadt- und keine Landstraßen sind.

Mit der rechten Maustaste bzw. mit dem Menüpfeil unter dem Icon kann wieder ein Menü aufgeklappt werden. Hier befindet sich ein neuer Menüpunkt „Landstraßen jwd zeichnen“. Normalerweise werden nur Landstraßen in der näheren Umgebung von Berlin (der Bereich reicht ca. von Brandenburg im Westen, Fürstenberg im Norden, Frankfurt (Oder) im Osten und Cottbus im Süden). Wenn dieser Menüpunkt aktiviert ist, werden alle in der Datenbank befindlichen Landstraßen gezeichnet.

Das nächste Icon schaltet das Zeichnen von Ortschaften ein und aus. Standardmäßig werden Orte mit Ortsnamen gezeichnet, außer wenn der entsprechende Menüpunkt ausgeschaltet ist. Im Untermenü „Kategorie“ kann das Zeichnen der Ortschaften auf bestimmte Kategorien eingeschränkt werden (siehe Seite 20). Mit Schriftgröße wird die Größe der Ortsnamen eingestellt. Die Auswahl reicht von „klein“ bis „sehr groß“. Für die Kategorien werden unterschiedliche Größen verwendet, Ortschaften der Kategorie 5 sind eben wichtiger als Orte der Kategorie 0 und bekommen eine entsprechend große Beschriftung. Mit dem Menüpunkt „Überlappungen vermeiden“ versucht die Applikation, beim Zeichnen der Namen Überlappungen zu vermeiden. Da diese Option das Zeichnen besonders bei großen Vergrößerungsstufen verlangsamt, ist sie in der Voreinstellung ausgeschaltet.

Die nächsten drei Icons dienen zum Zeichnen von U-, S- und Regionalbahnlinien. Für die Suche hat dies keine direkte Funktion. Die Darstellung hilft zum einen bei der Orientierung auf der Karte, da anhand der Liniennetzpläne in U- und S-Bahnen besser erkannt werden kann, welchen Teil Berlins die Karte gerade anzeigt. Zum anderen kann damit die Planung für Bike+Ride erleichtert werden. Bei Ausflügen ins Umland kann festgestellt werden, welche Bahnhöfe als Start- und Zielpunkte dienen können.

Beim Klick auf die Icons werden die Linien mit Bahnhöfen gezeichnet. Die entsprechenden Menüs sind für die drei Icons ähnlich aufgebaut und werden hier nur einmal beschrieben. Mit „U-Bahnlinien“ und „U-Bahnhöfe“ können die Linien und Bahnhöfe getrennt voneinander gezeichnet werden. Es gibt sowohl Auswahlmöglichkeiten für Linien als auch für einzelne Bahnhöfe. Mit dem Menüpunkt „U-Bhf-Namen“ werden neben den Bahnhofssymbolen die Namen der entsprechenden Bahnhöfe gezeichnet. Mit „Überlappungen vermeiden“ wird — genau wie bei Ortsnamen — versucht, Überlappungen zwischen den Schriftzügen zu vermeiden. Mit den Knöpfen für die VBB-Zonen A, B und C, außerhalb ABC und „in Bau/stillgelegt“ wird das Zeichnen des Netzes entsprechend eingeschränkt. Besonders das Ausschalten der VBB-Zone C kann für Zeitkartenbesitzer, die sich nur im Besitz von Monatskarten der Zonen AB befinden, interessant sein. Das Zeichnen von in Bau befindlichen Strecken ist rein informativ. So kann bei der Eröffnung schnell durch das Ändern der Kategoriebezeichnung in der Datenbasis die neue Strecke dargestellt werden.

Das nächste Icon dient zum Zeichnen von Gewässern (Flüssen, Kanälen und Seen). Auch hier gibt es eine Unterscheidung zwischen Gewässern innerhalb Berlins, im nahen Umland und im weiteren Umland. Flüsse werden anhand ihrer Bedeutung mehr oder weniger breit gezeichnet.

Mit dem letzten Icon in der ersten Gruppe werden sonstige Flächen (grün: Parks und Wälder, weiß: Flughäfen) gezeichnet. Das zugehörige Menü bietet zusätzlich das Zeichnen der Grenzen von Postleitzahlgebieten (die Datenbasis ist hier noch unvollständig) an, weiterhin die Grenzen von Berlin, Sehenswürdigkeiten, besonderen Gebäuden und Obstbaumalleen. Postleitzahlgrenzen und die Grenze von Berlin sind mit schwarzen Linien gekennzeichnet. Die Gebäude (Sehenswürdigkeiten) werden rot eingezeichnet. Für Obstbaumalleen wird für jede Obstsorte ein entsprechendes Symbol verwendet.

Maßstab Mit den nächsten beiden Icons ⑦ kann der Ausschnitt vergrößert oder verkleinert werden. Daneben ist der aktuelle Maßstab dargestellt. Der Maßstab bezieht sich auf den Bildschirm und ist nur korrekt, wenn die dpi-Auflösung des X11-Servers richtig angegeben wurde[14]. Beim Klick auf die Maßstabszahl erscheint ein Dialogfeld, in dem man den Maßstab verändern kann.

Übersichtskarte Mit dem letzten Icon in dieser Gruppe wird ein neues Fenster mit einer Übersichtskarte geöffnet. Die Übersichtskarte zeigt die Grenzen von Berlin und die wichtigsten Orte im Umland. Gleichzeitig wird der aktuell sichtbare Kartenausschnitt angezeigt. In der Übersichtskarte kann der Ausschnitt bewegt werden, um auch im Hauptfenster zu scrollen; ebenso wird auch der Ausschnitt aktualisiert, wenn im Hauptfenster gescrollt wird.

[14]Bei XFree86 muß beim Start des X-Servers die Option `-dpi` mit dem richtigen Wert angegeben werden.

Abbildung 3.3: Übersichtskarte

Scrolling Mit der Windrose kann das Hauptfenster gescrollt werden. Desweiteren kann für das Scrolling folgendes verwendet werden:

- die normalen Scrollbars rechts und unten
- die Cursortasten
- die Tasten „Pos1“ und „Ende“ zum schnellen Scrollen nach links und rechts, sowie die Tasten „Bild oben“ und „Bild unten“ zum schnellen vertikalen Scrollen
- Wenn die Option `Autoscroll` aktiviert ist, kann eine besondere Variante des Scrollings eingeschaltet werden: mit Klick auf die mittlere Taste wird der Scrollmodus eingeschaltet. Dieser Modus wird durch einen roten Punkt auf den geklickten Stellen angezeigt. Beim Bewegen der Maus wird nun der Ausschnitt mitbewegt. Mit einem weiteren Druck auf die mittlere Maustaste wird der Scrollmodus wieder ausgeschaltet.
- Wenn die Option `Followmouse` eingeschaltet ist, wird automatisch gescrollt, wenn sich die Maus den Rändern des Ausschnitts nähert. Dies ist einerseits bequem, weil keine weiteren Tasten gedrückt werden müssen, anderseits ist es nachteilig, daß gescrollt wird, auch wenn es vielleicht nicht erwünscht ist (z.B. wenn man die Maus nach oben in Richtung Icons bewegt).
- Die Scrolling-Methode im Übersichtsfenster wurde bereits auf Seite 30 beschrieben.

Im weiteren Sinne können auch andere Bewegungen des Kartenausschnitts als Scrolling aufgefaßt werden, z.B. die Auswahl von Straßen in den Straßenauswahldialogen. Mit der Option `smoothscroll` wird bei solchen Kartenausschnittsänderungen wirklich Scrolling durchgeführt, d.h. von der gegenwärtigen Stelle der Ausschnitt schrittweise zur neuen Stelle bewegt. Dadurch geht die Orientierung für den Benutzer nicht verloren.

Suche Die Icons in der nächsten Gruppe 8 steuern die Routensuche. Das erste Icon schaltet in den Suchmodus um. Der Suchmodus ist in der Voreinstellung eingeschaltet und kann normalerweise auch nicht geändert werden. Die folgenden Menüpunkte sind im angehängten Menü vorhanden:

Route löschen: Eine eingezeichnete Route wird gelöscht.

Route wiederherstellen: Die zuletzt gelöschte Route wird wieder dargestellt („Undo").

Route aktualisieren: Wenn Parameter wie die Windrichtung geändert wurden und keine automatische Aktualisierung eingestellt war, kann hiermit die Route wieder auf den aktuellen Stand gebracht werden. Diese Option bezieht sich nur auf die Anzeige der Route.

Suche wiederholen: Aus der aktuell dargestellten Route werden Start- und Zielort für eine neue Suche verwendet. Diese Option kann aufgerufen werden, um eine aktualisierte Route zu erhalten, wenn Optimierungen oder andere Suchkriterien geändert wurden. Im Gegensatz zu „Route aktualisieren" wird hiermit eine (in den meisten Fällen) geänderte Routenführung erzeugt.

Register: Im Register können bis zu neun Routen für einen schnellen Vergleich gespeichert werden. Beim Klick auf den Menüpunkt erscheint ein neues Fenster, in dem man in der „Abspeichern"-Reihe die aktuelle Route speichern kann und in der „Anzeigen"-Reihe eine gespeicherte wieder abrufen kann. Register, die bereits eine Route enthalten, sind rot eingefärbt. Zusätzlich zu den neun Registern kann auch das Undo-Register abgerufen werden[15]. Die Registerspeicherung ist nicht persistent. Für eine persistente Speicherung muß die Speicherfunktion (siehe Seite 36) aufgerufen werden.

Ausschnitt an Route anpassen: Hiermit wird der Ausschnitt durch Zoomen und Scrollen so geändert, daß die aktuelle Route komplett sichtbar ist.

Wenn beim Starten der Applikation `-advanced` eingegeben wurde, erscheinen neben weiteren Menüpunkten auch die folgenden zwei Menüpunkte: „Such-Statistik" und „Visual Search". Bei der Suchstatistik werden statistische Daten

[15] Technisch sind beide Registerarten gleich implementiert.

über die Suche — Anzahl der besuchten Knoten, Länge des Pfades, Suchdauer — in die Standardausgabe geschrieben.

Mit „Visual Search“ wird grafisch der Verlauf der Suche angezeigt. Für die verschiedenen verwendeten Suchalgorithmen (A*, Breitensuche, Tiefensuche) werden die expandierten Knoten angezeigt, somit kann mit dieser Funktion optisch verfolgt werden, wie die Suche abläuft. Im Abschnitt zum Suchalgorithmus (Seite 51) sind hierzu mehr Hinweise.

Im nächsten Menü-Abschnitt können einige Suchparameter eingestellt werden:

Einbahn-/gesperrte Straßen beachten: Diese Option ist in der Voreinstellung aktiviert. Hiermit wird bei der Suche verhindert, daß Routen durch Einbahnstraßen entgegen der Fahrtrichtung und durch gesperrte Straßen gefunden werden.

Tragen vermeiden: Wenn diese Option aktiviert ist, wird bei der Suche vermieden, daß Routen gefunden werden, an denen das Rad getragen werden muß.

Schlechte Wege vermeiden: Schlechte Wege, also Wege der Kategorie „Q3“, werden bei aktivierten Menüpunkt vermieden. Eine ähnliche Funktionalität kann auch in der Straßenqualitätsoptimierung (siehe unten) erreicht werden.

Fähren verwenden: Fähren werden bei der Suche berücksichtigt. Dabei wird die Zeit, die in der Fähre verbracht wird, nicht berücksichtigt.

Weitere Suchparameter kann man in den Optimierungen einstellen:

Ampel-Optimierung: Bei der Ampeloptimierung wird angenommen, daß an jeder Ampel eine durchschnittliche Zeit für das Warten verloren wird.

Straßenqualität-Optimierung, Straßenkategorie-Optimierung: Hier kann eingestellt werden, ob die Straßenqualität bzw. die Straßenkategorie in die Suche einbezogen werden soll.

Radwege-Optimierung: Optimierung der Suche nach Kriterien, ob Radwege vorhanden sind oder nicht.

Steigungsoptimierung: Mit dieser Option wird die Suche nach Steigungen optimiert. Dabei wird angenommen, daß die Referenzleistung auch für das Befahren der Steigung verwendet wird und somit eine wesentlich geringere Geschwindigkeit gefahren werden kann.

Optimierungsparameter einstellen: Dieser Menüpunkt öffnet ein Fenster (siehe Abbildung 3.4), in dem die Optimierungsparameter eingestellt werden können. Es stehen vier Ansichten zur Verfügung: Straßenqualität, Straßenkategorien, Radwege und Ampeloptimierung. In jeder dieser Ansichten wird ein Knopf „Verwenden“ angezeigt, welcher die gleiche Funktionalität wie der entsprechende Menüeintrag hat. Bei der Straßenqualität kann die maximale Geschwindigkeit eingestellt werden, die bei diesem Belag gefahren werden kann. 100 km/h ist eine theoretische Geschwindigkeit, die auch gleichbedeutend mit „keine Einschränkung“ ist.

Bei den Straßenkategorien erhält man eine ähnliche Ansicht. Hier sind alle Kategorien standardmäßig mit 100 km/h eingestellt, das heißt alle Straßenkategorien werden bei der Suche gleichberechtigt behandelt. Die Änderungsmöglichkeit hier hat folgenden Hintergrund: wenn ein Radfahrer ruhige Nebenstraßen unbedingt vorzieht, kann er hiermit die Suche optimieren. Nehmen wir an, der Radfahrer fährt mit einer Durchschnittsgeschwindigkeit von 15 km/h. Dann kann er bei den Hauptstraßen z.B. 7 km/h eintragen und alle anderen Felder unverändert lassen. In diesem Fall würde der Suchalgorithmus Nebenstraßen den Hauptstraßen vorziehen, es sei denn der Weg würde sich dadurch um mehr als das Doppelte verlängern. Anderseits können Radfahrer, die schnell vorankommen wollen und durch Nebenstraßen eher gebremst werden (kreuzende Vorfahrtstraßen, enge Straßen) für Nebenstraßen einen geringeren Wert eingeben, z.B. bei einer Durchschnittsgeschwindigkeit von 30 km/h den geringeren Wert 20 km/h. Um diese Eingabe zu vereinfachen, kann man in einem Auswahlfeld zwischen „Alle Straßen berücksichtigen“, „Nur Hauptstraßen“, „Hauptstraßen bevorzugen“, „Nebenstraßen bevorzugen“ und „Nur Nebenstraßen“ wählen.

Bei der Radwegeoptimierung ergibt sich ein ähnliches Bild. Auch hier kann der Benutzer zwischen der Benutzung von Radwegen oder Benutzung der Fahrbahn durch geeignete Wahl der Parameter wählen.

Bei der Ampeloptimierung kann angegeben werden, mit welcher Geschwindigkeit und Beschleunigung durchschnittlich gefahren wird. Anhand dieser Werte wird die durchschnittlich verlorene Strecke pro Ampel berechnet. Alternativ kann die Geschwindigkeit automatisch anhand der Referenzgeschwindigkeit eingestellt werden oder die verlorene Strecke auch manuell eingegeben werden.

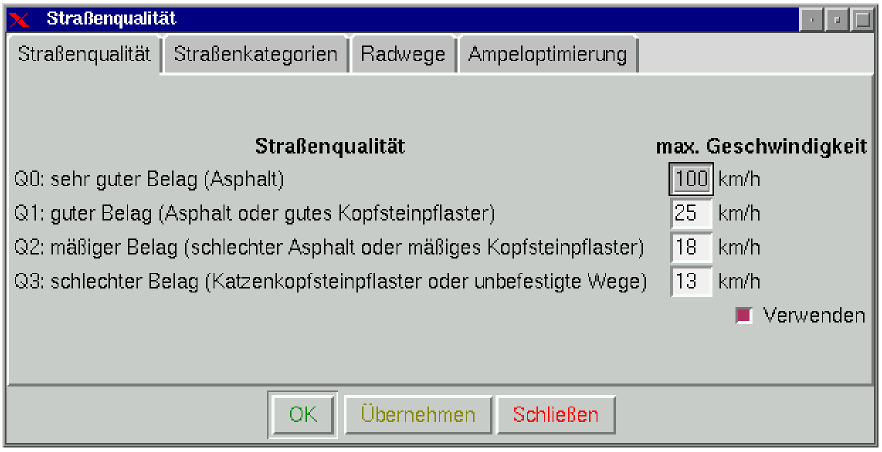

Abbildung 3.4: Einstellen von Optimierungsparametern

Mit den weiteren Optionen kann die Ausgabe des Suchergebnisses geändert werden. Mit der Option „Steigungen/Gefälle zeigen“ kann eingestellt werden, ob Steigungen (in Prozent angegeben) an der Route dargestellt werden sollen. Gefälle erhalten ein Minuszeichen vor der Zahl. Mit „Einfärben der Route“ kann angegeben werden, ob das Einfärben der Route über die Leistung oder die Windrichtung erfolgt, bzw. einfach rot oder blau gezeichnet wird. Die Option „Streckenprofil“ öffnet ein Fenster, in dem das Streckenprofil der aktuellen Route angezeigt wird. Mit „Startflagge“ und „Zielflagge“ wird die Ausgabe der Flaggen am Start- und Zielort ermöglicht. Zuletzt kann mit „Kalorienverbrauch anzeigen“ ein neues Fenster geöffnet werden, in dem der Kalorienverbrauch angezeigt wird.

Fahrtenbuch Neben der Anzeige der Route in der Karte kann auch eine textuelle Beschreibung der Route mit Hilfe des nächsten Icons angezeigt werden (Abbildung 3.5). Für jede Teilstrecke wird die Etappenlänge, die bisherige Gesamtlänge und der Straßenname ausgegeben. Während der Ausgabe wird vom Programm anhand von Winkelberechnungen festgestellt, ob es sich um einen Abbiegevorgang nach rechts oder links handelt. In diesem Fall wird auch diese Richtung mit einer Winkelangabe ausgegeben.

Mit Doppelklick auf eine Straße im Beschreibungsfenster wird der Kartenausschnitt so verschoben, daß die Straße angezeigt wird; außerdem wird die Straße markiert. Im Beschreibungsfenster gibt es noch zwei selektierbare Knöpfe: „Straßennamen an der Route“ und „Mit km-Angaben“. Mit dem ersten Knopf wird angegeben, daß entlang der Route die Straßennamen eingezeichnet werden sollen. Ist der zweite Knopf aktiv, wird die Teilstrecke in Kilometern in die Karte eingezeichnet. Weiterhin können folgende Aktionen mit der Routenbeschreibung durchgeführt werden:

Sichern: Die textuelle Beschreibung kann in eine Datei abgespeichert werden,

Drucken: Die Beschreibung wird hiermit ausgedruckt.

Mail: Der Beschreibungstext kann als E-Mail verschickt werden. Diese Aktion erfordert eine installierte `Mail::Send`-Installation und funktioniert nur mit Unix.

Fax: Versendet den Beschreibungstext als Fax. Hierfür muß die `sendfax`-Distribution installiert sein.

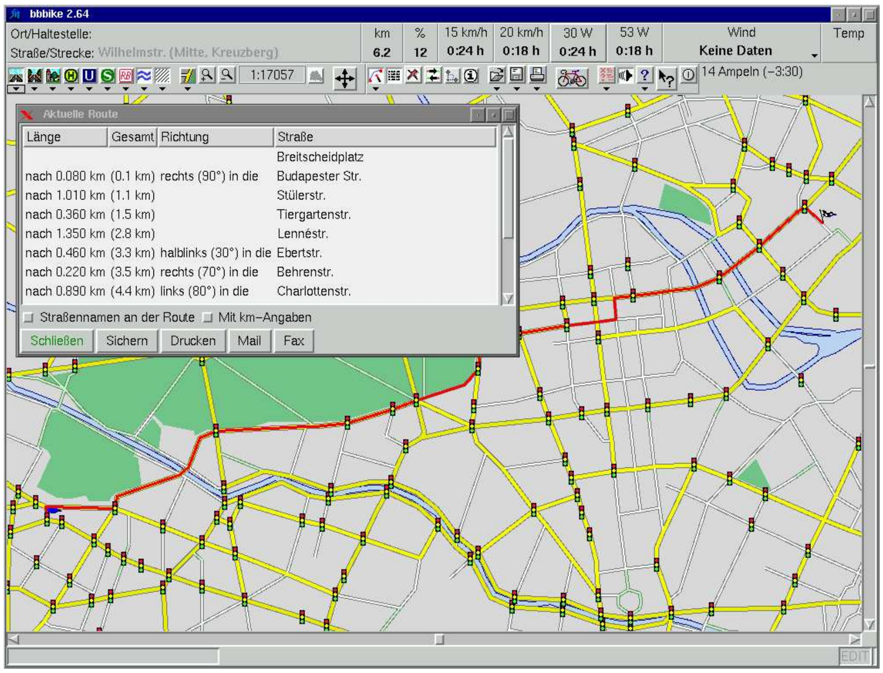

Abbildung 3.5: Textuelle Beschreibung einer Route

Route löschen Das nächste Icon dient zum Löschen einer Route. Mit der rechten Maustaste kann eine Route ebenfalls gelöscht werden. Mit der Backspace-Taste wird das letzte Stück einer Route gelöscht und mit der Delete-Taste das Stück bis zum letzten Via-Punkt.

Rückweg Daneben befindet sich das Icon, mit dem der Rückweg der Route berechnet wird. Ein einfaches Austauschen von Start- und Zielpunkt reicht hierbei nicht aus, weil durch das Beachten von Einbahnstraßen und Steigungen eine andere Route als Ergebnis gefunden werden kann.

Koordinaten Das nächste Icon ist nur aktiviert, wenn die Option `-advanced` eingegeben wurde. Es dient dazu, Koordinaten von ausgewählten Punkten auszugeben.

Ein- und Ausgabe Die nächste Gruppe von Icons ⑨ dient der Dateieingabe und -ausgabe sowie Druckoperationen. Das erste Icon dient dem Laden einer

Route. Routen werden als `bbr`-Dateien (Beschreibung siehe Seite 53) abgespeichert. Nach dem Laden einer Route wird diese in der Karte dargestellt und der Kartenausschnitt auf den Startpunkt verschoben. Im Menü des Icons findet man eine Liste der vier zuletzt abgespeicherten Routen-Dateien.

Daneben befindet sich das Icon zum Abspeichern einer Route als `bbr`-Datei. Weitere Bildausgabeformate können im Menü dazu gewählt werden: Postscript, GIF, JPEG und PPM. Für die Postscript-Ausgabe wird eine Tk-interne Funktion verwendet und funktioniert auf allen Betriebssystem-Plattformen. Die anderen Bildformate werden im Zusammenspiel von Tk, Ghostscript und dem netpbm-Paket erstellt und funktionieren nur auf Plattformen, auf denen diese Programme installiert sind. Für die Ausgabe kann man auch die Auflösung wählen: 36 dpi, 72 dpi, 100 dpi (die Voreinstellung) und 150 dpi.

Das letzte Icon in der Gruppe dient zum Drucken des aktuellen Kartenausschnitts mit allen Kartenobjekten und der Route. Das Drucken erfolgt über den Umweg Ghostview oder Gv oder direkt über einen Postscript-fähigen Drucker, falls diese beiden Programme nicht installiert sind. Im Menü kann zwischen Farbausgabe, Graustufen und Schwarz-Weiß-Ausgabe wählen. Weiterhin kann zwischen Landscape- und Portrait-Modus gewählt werden. Standardmäßig wird der Druck auf A4 skaliert. Auf Wunsch kann auch eine Legende gedruckt werden, die die wichtigsten Kartenobjekte erklärt und einen Nordpfeil enthält. Normalerweise wird diese Legende in der linken oberen Ecke ausgedruckt, kann aber mit der letzten Option in diesem Menü nach rechts verschoben werden.

BikePower Das Fahrrad-Icon startet die Eingabemaske von BikePower. Die Funktionen von BikePower sind im Abschnitt 3.6.3 auf Seite 81 beschrieben.

Optionen Weitere Optionen können im Menü des nächsten Icons ⑩ eingestellt werden:

Auf Anfrage aktualisieren: Einige Optionsänderungen wie Einschränkung der gezeichneten Straßenkategorien oder Zeichnen von Umrandungslinien (Outlines) erfordert das Neuzeichnen der Karte. Wenn dies auf langsamen Rechnern passieren soll, ist es sinnvoll, das Aktualisieren so lange wie möglich zu verzögern, besonders wenn viele Optionsänderungen getätigt werden.

Ausgabe sofort aktualisieren: Das ist die Standardeinstellung. Optionsänderungen werden sofort gezeichnet.

Ausgabe verzögert aktualisieren: Hier erfolgt die Aktualisierung verzögert nach etwa einer Sekunde, wenn in dieser Zeit keine weitere Optionsänderung vorgenommen wurde.

Netz sofort aktualisieren: Beim Ändern des gezeichneten Straßennetzes, also wenn z.B. zusätzlich zu den Berliner Straßen auch Landstraßen gezeichnet werden, muß das interne Netz-Objekt neu berechnet werden. Standardmäßig erfolgt das automatisch, kann aber auch hier ausgeschaltet werden.

Alles aktualisieren: Hiermit kann die Aktualisierung vorgenommen werden, wenn „Auf Anfrage aktualisieren" eingestellt war.

gedrehte Zeichensätze: Bei eingeschalteter Option werden für Straßen- und Gewässernamen gedrehte Zeichensätze verwendet, wenn der X11-Server das zuläßt.

Landscape, Portrait: Normalerweise wird die Karte im Landscape-Modus gezeichnet. Dabei ist Norden nach oben gerichtet. Wenn eine Route in Nord-Süd-Richtung eine größere Ausdehnung hat als in Ost-West-Richtung und diese Route ausgedruckt werden soll, könnte es sinnvoll sein, die Karte im Portrait-Modus zu zeichnen.

Canvas balloon: Hiermit wird ein Untermenü geöffnet. Mit „kein" wird angegeben, daß über der Karte niemals ein „Balloon" [16] angezeigt wird. Mit „nur Route" erscheint der „Balloon" nur über einer Route. Bei „überall" erscheint der „Balloon" über allen Kartenobjekten. Bei normalen Kartenobjekten wird im „Balloon" der Name der Straße etc. angezeigt. Bei Routen wird der Name der Straße sowie die Routenlänge bis zu dieser Stelle und die Fahrzeit nach der Referenzgeschwindigkeit angezeigt.

Farben ändern: Mit diesem Punkt wird der Standard-Perl/Tk-Dialog zum Ändern der Farben aufgerufen. Dieser Dialog ändert die Darstellung nur für die aktuelle Sitzung, ein Speichern der Änderung ist (noch) nicht möglich.

Schriftart ändern: Wenn das Modul `Tk::FontDialog` installiert ist, wird mit diesem Punkt ein Dialog zum Ändern der Schriftarten aufgerufen. Die Änderung kann durch Speichern der Optionen permanent gemacht werden.

Ständige Markierung: Einige Aktionen erzeugen eine Markierung auf der Karte, z.B. die Auswahl einer Straße. Normalerweise blinkt die Markierung ungefähr 10 Sekunden lang und verschwindet dann. Mit dieser Option bleibt die Markierung permanent.

Markierung löschen: Eine permanente Markierung kann hiermit gelöscht werden.

---

[16] „Balloons" werden auch als „Tooltip"-Fenster bezeichnet. Das sind kleine Fenster, die einen kurzen Hilfstext enthalten und nach kurzer Verweildauer des Mauscursors über einem GUI-Objekt erscheinen.

Followmouse: Hiermit wird die `followmouse`-Option eingeschaltet.

Geladene Route zentrieren: Wenn diese Option eingeschaltet ist, wird beim Laden einer Route der Kartenausschnitt auf den Startpunkt der Route verschoben.

Status nach STDERR: Fehlermeldungen werden gewöhnlich in einem Dialogfenster angezeigt. Mit dieser Option kann die Ausgabe der Meldungen auf die Standardfehlerausgabe der Console umgelenkt werden.

Wortreich (verbose): Es werden zusätzliche Ausgaben auf der Standardausgabe zum Zweck der leichteren Fehlersuche erzeugt.

Update über das Internet: Mit diesem Menüpunkt können geänderte Daten zum Programm über das Internet aktualisiert werden.

Beim Klick auf das Options-Icon wird ein Optionseditor geöffnet, mit dem alle Angaben, die in der Kommandozeile gemacht werden können, auch hier editiert werden können.

Das nächste Icon dient zum Ein- und Ausschalten der Soundausgabe. Bei einem Mausklick auf der Karte wird hiermit ein zusätzlicher akustischer Klick erzeugt, sofern die Soundhardware und -software dies zuläßt.

Das nächste Icon dient zur Hilfe des Benutzers. Beim Klick auf das Icon wird die Online-Dokumentation geöffnet. Die Online-Dokumentation liegt in zwei Formaten vor: das Perl-eigene Dokumentationsformat POD[17] und HTML. Wenn das Modul `Tk::Pod` installiert ist, wird der POD-Viewer aufgerufen; ansonsten wird ein Web-Browser mit der HTML-Dokumentation gestartet.

Im Menü des Hilfe-Icons befinden sich folgende Punkte:

Kurzhilfe: Die Kurzhilfe zeigt an, welche Funktionen die drei Maustasten aktuell haben. Je nach Kontext (über welchem Teil der Anwendung sich der Mauszeiger befindet) ändert sich der Hilfetext.

Kontexthilfe: Hiermit wird die kontextsensitive Hilfe eingeschaltet. Nach dem Einschalten erhält der Mauszeiger ein zusätzliches Fragezeichen, wenn das darunterliegende Widget[18] einen Hilfetext in der POD-Datei enthält. Ein durchgestrichenes Fragezeichen erscheint, wenn keine kontextsensitive Hilfe zu diesem Widget existiert.

Dokumentation (lokal): Die lokale HTML-Dokumentation wird angezeigt.

Dokumentation (WWW): Die aktuell im WWW erhältliche Dokumentation wird angezeigt.

[17]POD steht für „Plain old documentation".

[18]Widget: Window Object. Damit werden Teile einer grafischen Benutzeroberfläche wie Knöpfe, Menüs und Auswahlfenster bezeichnet.

Über: Programmname, Autor und eine kurze Systeminformation wird angezeigt.

Busy-Zeiger zurücksetzen: Es kann unter ungünstigen Bedingungen passieren, daß eine Operation nicht korrekt beendet wird und der Busy-Zeiger (Uhr oder Sanduhr) nicht mehr in einen normalen Mauszeiger umgewandelt wird. Hiermit kann der Zeiger wieder zurückgesetzt werden.

Das nächste Icon dient zum Einschalten der kontextsensitiven Hilfe (siehe auch den Menüpunkt mit der gleichen Funktion). Beim Klick auf die rechte statt die linkte Maustaste bleibt die kontextsensitive Hilfe persistent bis wieder auf das Icon geklickt wird. Somit kann man sich über die grafische Benutzeroberfläche durch die Dokumentation schmöckern.

Das letzte Icon dient zum Beenden des Programms.

Wenn die Anwendung mit der Option `-advanced` gestartet wurde, gibt es zusätzliche Icons und Menüpunkte:

- „Liste der Straßen neu erstellen", „Update der Straßen" und „Schnelles Update der Straßen": Wenn die Datenbasis editiert wurde, arbeitet das Programm weiterhin mit den bereits geladenen Daten. Um nun bei der Straßenauswahl die aktualisierte Liste zu erhalten, muß der Punkt „Liste der Straßen neu erstellen" angeklickt werden. Mit „Update der Straßen" wird das Straßennetz mit den neuen Daten komplett neu gezeichnet. Der Punkt „Schnelles Update der Straßen" führt ein „diff" zwischen den alten und neuen Daten durch und zeichnet nur die geänderten Straßen neu. Diese Funktionen stehen auch den anderen Strecken- und Punkttypen zur Verfügung.

- „Fragezeichen": In den Entwicklungsdaten existiert eine weitere Datei, in der Strecken und Punkte mit unsicherer Datenbasis aufgezeichnet werden. Dies dient in erster Linie als Erleichterung für die Pflege der Daten.

- „Such-Statistik" und „Visual Search" im Routen-Menü sind bereits auf Seite 32 beschrieben worden.

- „Straßennetz neu berechnen": Bei der Aktualisierung der Daten und bei der sonstigen Entwicklung kann die Neuberechnung des internen Straßennetzes nötig sein.

- „undef netz": Mit diesem Menüpunkt wird das interne Straßennetz gelöscht. Damit kann belegter Speicherplatz wiedergewonnen werden.

- „Ampeln an der aktuellen Route" im Routenlisten-Menü zeigt die Ampeln, die auf der Route liegen, in einer ausdruckbaren Liste an. Auch diese Daten sind als Hilfe für den Pfleger der Daten gemeint.

Der Koordinatenmodus und das zugehörige Menü sind gänzlich ausgeschaltet, wenn `-advanced` nicht aktiv ist. Im Koordinatenmodus werden bei jedem Klick die Koordinaten des aktuellen Punktes nach `STDERR` und in die X11-Selection geschrieben. So können Punkte in die Datenbasis übertragen werden, indem sie mittels eines normalen Texteditors übernommen werden.

Im Menü existieren die folgenden Punkte:

Koordinatenliste zeigen: Zusätzlich zu den beiden genannten Ausgabearten können angeklickte Koordinaten in einem Tk-Fenster angezeigt werden. Dies ist die beste Methode unter Windows, um die Koordinatendaten auszulesen.

Ausgabe: BBBike kann mit mehreren Koordinatensystemen arbeiten. Im Untermenü des Punktes „Ausgabe“ sind die unterstützten Koordinatensysteme zu finden.

Koordinatensystem: Das verwendete Koordinatensystem für das Zeichnen der Kartenobjekte wird hier ausgewählt. Auch hier erfolgt die Auswahl über ein Untermenü wie bei „Ausgabe“.

Kreuzungen/Kurvenpunkte zeichnen: Mit dieser Option werden alle Zwischenpunkte der vektorisierten Strecken gezeichnet.

Kurvenpunkte rot/grün: Hiermit kann die Farbe dieser Zwischenpunkte eingestellt werden.

Präfix-Ausgabe: Die Ausgabe der Koordinaten erfolgt normalerweise im Format „x“ Komma „y“. Mit dieser Option kann angegeben werden, daß ein zusätzlicher Buchstabe vor den Koordinatenangaben das Koordinatensystem näher bezeichnet.

Plätze zeichnen: Zusätzlich zu der Straßendatei gibt es auch eine Datei aller Plätze, die normalerweise nicht gezeichnet werden. Mit diesem Menüpunkt können auch die Plätze gezeichnet werden.

Neu laden: Beim Anwählen dieses Menüpunktes wird verglichen, ob sich die Datenbasis geändert hat. Ist das der Fall, werden die geänderten Daten aktualisiert (sofern möglich, wird das schnelle Update der Daten verwendet).

Punkt einfügen: Dieser Punkt wird ausgewählt, um einen neuen Punkt in einer bestehenden Strecke einzufügen. Dabei müssen zuerst der erste Nachbarpunkt, dann der neue Punkt und schließlich der zweite Nachbarpunkt angeklickt worden sein, um die Funktion auszuführen. Diese Funktion stellt sicher, daß in alle Strecken mit dem selben Streckenabschnitt die Einfügeoperation durchgeführt wird.

Punkt ändern: Hiermit wird ein ausgewählter Punkt an eine neue Position verschoben. Hier muß nur der zu verschiebende Punkt und die neue Position ausgewählt sein. Auch diese Funktion sichert die Konsistenz der Daten.

Standard-Modus: Dieser Modus ist beim Aufrufen der Anwendung aktiv.

Edit-Modus Berlin: Mit Klick auf diesen Punkt werden das Koordinatensystem und die Dateien zum Zeichnen der Kartenobjekte umgestellt, so daß die Originaldaten für das Berliner Stadtgebiet angezeigt werden.

Edit-Modus Brandenburg: Dieser Menüpunkt schaltet Koordinatensystem und Dateien für das Editieren im Brandenburger Gebiet um.

Die nächsten Menüpunkte enthalten gleich aufgebaute Untermenüs: Mit der Option „Modus“ wird in einen speziellen Editiermodus für dieses Kartenobjekt umgeschaltet. Mit „Undef all“ werden alle zusätzliche Variablen, die in diesem Modus erzeugt wurden, gelöscht. Mit „Speichern all“ werden die geänderten Daten unter einem anderen Dateinamen gelöscht. Von den Modi kann höchstens einer gleichzeitig aktiv sein.

Radwege: Im Radwege-Modus werden die Radwege an den Straßen in die Karte eingezeichnet. Beim Klick auf eine Teilstrecke einer Straße wird ein neues Fenster geöffnet. In diesem Fenster kann man für die beiden Richtungen der Straße angeben, ob ein Radweg existiert. Mit „Neu zeichnen“ kann man die Radwege neu einzeichnen. Mit „Sichern“ werden Änderungen abgespeichert. Wenn „Auto-Sichern“ aktiv ist, werden Änderungen sofort abgespeichert.

Ampelschaltung-Modus: Hier können Ampelschaltungsdaten entweder angezeigt oder auch editiert werden. Es gibt zwei Formen für die Ampelschaltungsdaten; dementsprechend gibt es auch zwei Kästen für die Ausgabe. Bei der Ausgabe werden die Zeit der Beobachtung („Wochentag“, „Zeit“) und die Richtung („von“ und „nach“) angezeigt. Die Spalten „grün“, „rot“ und „Zyklus“ zeigen die Länge der Grün- und Rotphasen bzw. die Kombination aus beiden (Zyklus) an.

Die zweite Art der Ampelschaltungsdaten kann auch editiert bzw. neue Daten eingegeben werden; bei der ersten Art muß dies im Texteditor erfolgen. Auch hier kann entweder per Button oder nach jeder Änderung gesichert werden.

Die Zyklus-Zeiten werden auf der Karte neben den Ampeln eingezeichnet. So kann schnell festgestellt werden, ob ein Straßenzug eine einheitliche Zyklus-Zeit hat („Grüne Welle“). Da sich die Zyklus-Zeiten im Tageslauf ändern, können mehrere Zyklus-Zeiten an dieser Stelle stehen. Mit den Option „Tages-“, „Berufs-“ und „Nachtverkehr“ kann diese Ausgabe für bestimmte Zeiten eingeschränkt werden.

Im Options-Menü befinden sich ebenfalls einige zusätzlichen Menüpunkte:

Konfigurations-Wizard: Hier befindet sich eine Studie zum Anlegen eines Konfigurations-Wizards für die Erstanmeldung eines Benutzers an. Der Wizard kann auch als Alternative für den Optionseditor verwendet werden.

Ptksh: Dieses Skript ist standardmäßig in der Perl/Tk-Distribution enthalten. Mit dem Skript kann eine Shell gestartet werden, die das Ausführen von beliebigen Perl- und Perl/Tk-Befehlen und die Ausgabe von nicht-lexikalischen Variablen[19] ermöglicht. Diese Funktion wird benötigt, um einen Fehler nachträglich zu finden, ohne den Perl-Debugger vorher gestartet zu haben.

Eval /tmp/add.pl und **Reload modules:** Diese beiden Punkte sind im Abschnitt 3.9 auf Seite 98 beschrieben.

Zusätzlich zeichnen: Im Untermenü kann man angeben, ob man eine beliebige Strecken- oder Punktdatei laden möchte. Die Syntax dieser Dateien entspricht der gleichen wie bei den Dateien in der Datenbasis. Mit den Auswählen-Funktionen wird ein Dialog zur Auswahl der Strecke oder des Punktes angezeigt.

Die Karte Im Hauptfenster wird die Karte dargestellt. In den vorher beschriebenen Menüs kann angegeben werden, welche Kartenelemente gezeichnet werden sollen, d.h. wie weit die Ausbreitung reichen soll (nur die Stadt oder weiter ins Umland) sowie welche Typen von Kartenelementen gezeichnet werden sollen. Die Eingrenzung hat den Vorteil, daß die Startzeit verringert werden kann und der Speicherverbrauch minimiert wird. Ebenso dient die Einschränkung der Kartenelementtypen diesen Vorteilen; außerdem geht durch zu viele Kartenelemente und zu großem Maßstab die Übersichtlichkeit verloren.

Das Suchen von Routen geht nur über Stadt- und Landstraßen, aber nicht mit den anderen Streckentypen wie Gewässern und Bahnstrecken. Um auf der Karte eine Route zu suchen, wird mit einem Klick auf eine Straße zunächst der Startpunkt definiert. Eine Startflagge markiert diesen Punkt. Mit einem weiteren Klick wird der Zielpunkt definiert und mit einer Zielflagge markiert. Unter Berücksichtigung der aktuell eingestellten Suchparameter (Optimierungen, Einschränkungen) wird eine Route gesucht und in die Karte eingezeichnet. Für das Einfärben der Route gibt es mehrere Möglichkeiten: „Leistung“ oder „Wind“. Bei „Leistung“ wird die benötigte Leistung im Vergleich zu Normalbedingungen

[19] Lexikalische Variablen in Perl sind Variablen, die mit dem Schlüsselwort `my` deklariert wurden. Diese Variablen sind im Block sichtbar, in dem sie deklariert wurden, und entsprechen somit lokalen Variablen in anderen Programmiersprachen. Dagegen sind lokale Variablen (mit `local` deklariert) im deklarierten Block und in allen aus diesem Block aufgerufenen Funktionen sichtbar.

(Windstille, ebene Strecke) verglichen. Wenn eine höhere Leistung benötigt wird, wird die Route rot eingefärbt, bei einer niedrigeren Leistung grün, bei gleicher Leistung orange. Wenn „Wind“ ausgewählt ist, Winddaten vorhanden und nicht ignoriert werden sollen, wird die Route anhand der Windrichtung eingefärbt: für Rückenwind grün, für Seitenwind orange und für Gegenwind rot, jeweils mit einigen Abstufungen. Weiterhin besteht die Möglichkeit, die Route nur rot oder blau einzufärben.

Mit einem weiteren Klick kann die Route von ihrem Zielpunkt erweitert werden; dies ist sinnvoll, wenn nicht die kürzeste Route zwischen zwei Punkten, sondern eine Rundfahrt gesucht werden soll oder eine bereits gefahrene Route eingezeichnet werden soll. Die früheren Zielpunkte werden dabei zu Via-Punkten.

Mit Klick auf die rechte Maustaste wird die Route gelöscht. Eine gelöschte Route kann mittels der Undo-Funktion oder der Tastenkombination Ctrl-Z wiederhergestellt werden. Mit der mittleren Maustaste[20] oder der Backspace-Taste wird die letzte Teilstrecke (bis zum letzten Kurvenpunkt oder zur letzten Kreuzung) gelöscht. Mit der Delete-Taste wird die gesamte Route bis zum letzten Via-Punkt gelöscht.

Um die freie Eingabe von Teilstrecken zu ermöglichen, kann mit einem Klick bei gedrückter Alt- oder Shift-Taste (je nachdem, welche Tastenkombination nicht durch den Window-Manager belegt ist) ein beliebiger Punkt auf der Karte gewählt werden. Die Route wird dann vom Zielpunkt ohne Suche linear bis zum neuen Zielpunkt verlängert. Um wieder die Suche verwenden zu können, muß sich der letzte frei eingegebene Punkt auf einer Straße befinden.

Der aktuelle Kartenausschnitt kann durch die oben beschriebenen Scrolling-Methoden bewegt werden. Mit den Zoom-Funktionen und der direkten Eingabe des Maßstabs kann die Größe des Kartenausschnitts geändert werden.

3.3.2 Implementierung

Die Funktionalität der Anwendung kann in folgende Gruppen aufgeteilt werden, die in den nachfolgenden Abschnitten genauer beschrieben werden:

Basis: In diese Gruppe gehört das Grundgerüst der Anwendung. Dazu gehört das Laden von Modulen, Variablendeklarationen, Einlesen von Konfigurationsparametern über die Kommandozeile oder Konfigurationsdateien und die Lösung von Problemen der Kompatibilität bei Anwendung verschiedener Hardware, Betriebssysteme, Perl- und Perl/Tk-Versionen.

Da Teile des Programmcodes auch von anderen Teilen des Informationssystems mitverwendet werden, sind Module ausgelagert worden. Die Beschreibung dieser Module befindet sich im Abschnitt 3.6 ab Seite 71.

[20] Sofern die Maustaste nicht durch das Autoscrolling belegt ist.

Benutzeroberfläche: Hier wird die Erstellung der grafischen Benutzeroberfläche erläutert.

Ausgabe der Kartenobjekte: In diesem Abschnitt wird das Zeichnen der Kartenobjekte beschrieben.

Interaktivität mit dem Benutzer: Der Benutzer verwendet die dargestellte Karte, um Aktionen auszulösen, die hier beschrieben werden.

Suche: Beschreibung der Funktionen, die die Suche aufrufen und die Parametrisierung der Suche steuern. Die eigentliche Suche wird in `Strassen.pm` durchgeführt und im Abschnitt 3.3.2 auf Seite 51 genauer beschrieben.

Routen: Erläuterung der Funktionen zur Manipulation von Routen.

I/O[21] mit der Außenwelt: Hier wird die Interaktion des Programms mit dem Betriebssystem und anderen Programmen behandelt.

Wetterdaten: Hier werden die Funktionen, die für die Analyse, das Laden und Eingeben von Wetterdaten zuständig sind, behandelt.

Sonstiges: Zum Schluß werden sonstige Funktionen, die nicht anderweitig kategorisiert werden können, beschrieben.

Basis

Die Anwendung besteht aus einem Hauptprogramm (`bbbike`) und mehreren Modulen, die entweder aus Gründen der Wiederverwendbarkeit oder wegen der Minimierung des Speicherverbrauchs ausgelagert wurden.

Module Module, die für die Anwendung unbedingt notwendig sind, werden mit dem Perl-Kommando `use` beim Start geladen. Dazu gehört natürlich das Modul `Tk`, das die Funktionalität für die grafische Oberfläche zur Verfügung stellt und das später beschriebene Modul `Strassen.pm`, das die Verbindung der Anwendung zur Datenbasis darstellt. Weitere Module werden bei Bedarf zur Laufzeit geladen. Dies hat den Vorteil, daß der Speicherverbrauch minimal gehalten werden kann, wenn nicht alle Features der Anwendung verwendet werden.

Bei der Verwendung von Perl/Tk gilt es einige Punkte bei der Kompatibilität zu beachten. Grundsätzlich sind Perl/Tk-Programme sowohl unter Windows als auch unter Unix ohne Änderungen lauffähig. Kompatibilitätsprobleme entstehen z.B. dadurch, daß Features der neueren Perl/Tk-Version 800.xxx verwendet werden, die Anwendung aber auch unter der Version 402.xxx laufen soll. Das Modul `TkCompat.pm` hält Funktionen bereit, die diese neuen Features emuliert oder leere Platzhalter-Funktionen bereitstellt. In dem Modul werden auch einige bekannte

[21] Input/Output: Ein-/Ausgabe

Bugs der Perl/Tk-Version 402.xxx behoben. Weitere Probleme durch unimplementierte Funktionalität im Windows-Port von Perl/Tk werden in `WinCompat.pm` behoben. Auch hier werden Bugs behoben oder Platzhalter-Funktionen bereitgestellt.

Auch bei der Interaktion der Anwendung mit dem Betriebssystem gibt es zwischen Unix und Windows Unterschiede, die von Perl nicht transparent behandelt werden, z.B. das Starten von externen Programmen oder die Benutzerverwaltung. Die Funktionalität für Windows für diese Bereiche des Betriebssystem wird in `Win32Util.pm` bereitgestellt. Ebenso gibt es ein `UnixUtil.pm`, welches aber — wegen der Unix-Orientierung von Perl — wesentlich weniger Funktionen enthält.

Um das Programm mit relativen Pfaden betreiben zu können, werden die Module `FindBin` und `lib` verwendet. Mit diesen beiden Modulen kann einmal der Pfad des Programms selbst ermittelt werden und dann relativ zu diesem Pfad die Include-Verzeichnisse für die weiteren Module gesetzt werden.

Weitere Module werden im Abschnitt 3.6 ab Seite 71 beschrieben.

Variablendeklarationen und -initialisierung Im nächsten Abschnitt im Programmcode werden Variablen deklariert und initialisiert. Bei Perl hat man die Möglichkeit der Deklaration von globalen Variablen mit „use vars“ oder die datei- und blocklokale Deklaration mit „my“. Die erste Methode hat sich als vorteilhafter erwiesen, wenn Programmcode nachträglich in Module verschoben wird und dabei auf globale Variablen zugegriffen werden soll. Die Deklaration mit „use vars“ ist auch während der Entwicklung vorteilhafter, da nach dem Nachladen von Programmcode während der Laufzeit weiterhin auf globale Variablen zugegriffen werden kann, dateilokale Variablen jedoch verloren gehen.

Auf eine genaue Auflistung der Variablen wird verzichtet. Mit den Variablen werden Konstanten für die Erscheinung der Kartenobjekte (Farben, Strichbreiten etc.), voreingestellte Dateinamen für die Datenbasis und Konfigurationsparameter zugewiesen.

Konfiguration Für die Konfiguration der Anwendung wird das Perl/Tk-Modul `Tk::Getopt` verwendet. Dieses Modul ermöglicht die Konfiguration aus den folgenden Quellen:

- Laden einer Konfigurationsdatei
- Kommandozeilen-Parameter
- GUI-Interface

Kommandozeilen-Parameter haben eine höhere Priorität als die Werte der Konfigurationsdatei. Im GUI-Interface geänderte Werte gehen sofort in die Anwendung ein und können in der Konfigurationsdatei gespeichert werden. In der Anwendung wird eine Tabelle mit den Optionsnamen und zugehörigen Variablen bereitgestellt, aus der das GUI-Interface automatisch erstellt werden kann.

Benutzeroberfläche

Widgets Die Widgets der Anwendung werden so dargestellt, daß der vorhandene Platz optimal ausgenutzt wird. Beim Start der Anwendung wird festgestellt, wie groß das Hauptfenster ist; entsprechend wird die Anzahl der gezeichneten Felder für die Geschwindigkeits- und Leistungsanzeige angepaßt[22]. Bei einer Änderung der Größe des Hauptfensters wird die Anzahl der gezeichneten Felder entsprechend geändert. Diese Funktionalität wird in `arrange_topframe` und `arrange_symframe` bereitgestellt.

Im unteren Teil des Hauptfensters wird die Progressbar dargestellt, der bei längeren Operationen den Fortschritt in Prozent anzeigt. Für dieses Widget wird das Modul `SRTProgress.pm` verwendet.

Iconleiste und Menüs Die Iconleiste wird mit der Funktion `do_iconframe` erzeugt. Diese Funktion ist so aufgebaut, daß — je nach Größe des Hauptfensters — die Iconleiste in einer oder zwei Reihen angelegt werden kann.

Die Icons sind so aufgebaut, daß mit Klick auf die rechte Maustaste oder mit Klick auf den darunterliegenden Pfeil ein Menü geöffnet wird. Einige Menüeinträge kommen in den meisten Menüs vor; die Erzeugung ist deshalb in den Funktionen `menu_entry_up_down` (für die Änderung der Schichtreihenfolge der Kartenobjekte) und `menu_entry_choose_ort` (für die Anzeige des Dialogs für die Straßen- oder Punktauswahl). Das Zeichnen des Pfeils und das Binden der rechten Maustaste wird in den Funktionen `menupfeil` und `menuright` durchgeführt.

Tastatur- und Mausbindung Die Tastatur- und Mausbindungen[23] werden mit der Tk-Methode `bind` durchgeführt. Es existieren Tastaturbindungen für das Zeichnen von Kartenobjekten sowie zum Ein- und Ausschalten von einigen Attributen.

Die Mausbindungen speziell für die Kartenobjekte werden weiter unten erläutert.

Sonstige GUI-Erweiterungen In der Anwendung werden für die GUI weitere Module verwendet:

Tk::ContextHelp: Hiermit wird die kontextsensitive Hilfe aktiviert. Für GUI-Objekte, für die ein Hilfetext existiert, kann dieser in einem Text-Fenster oder, falls installiert, in einem POD-Fenster angezeigt werden.

Tk::CanvasBalloon: In der Perl/Tk-Distribution ist bereits das Modul `Tk::Balloon` für die Anzeige von „Tooltips“ enthalten. Mit dem Modul

[22] Es ist auch möglich, für Rechner mit wenig Hauptspeicher die Anzahl der gezeichneten Felder zu reduzieren.

[23] Bindungen sind Aktionen, die beim Betätigen einer Taste oder einer Mausbewegung ausgelöst werden.

`Tk::CanvasBalloon` erhält man diese Funktionalität auch für Canvas-Objekte. So kann beim Führen des Mauscursors über Kartenobjekte der entsprechende Name angezeigt werden.

Kartenobjekte

Die `plot*`-Funktionen sind dafür verantwortlich, die Datenbasis auszulesen und das Zeichnen der Kartenobjekte durchzuführen. Die Plot-Funktionen sind prinzipiell gleich aufgebaut: In einer Schleife werden die Daten (die Koordinaten) ausgelesen. Anhand der zugeordneten Kategorie und den eingestellten Attributen für das Zeichnen werden die Kartenobjekte auf dem Canvas gezeichnet.

Die verschiedenen Plot-Funktionen bearbeiten die verschiedenen Typen der Datenbasis. Mit `plotstr` werden alle strecken- und flächenbasierten Daten gezeichnet, also Straßen, Bahnstrecken, Gewässer und sonstige Flächen. `plot_sperre` zeichnet Straßen und Einbahnstraßen. `plotp` ist eine allgemeine Verteilerfunktion, die für das Zeichnen von punktbasierten Daten verwendet wird. Diese Funktion ruft `plotorte` (Zeichnen von Ortschaften), `plotobst` (Zeichnen von Vorkommen von Obstbaumalleen), `plot_hoehe` (Höhenpunkte) und `plot_point` (sonstige Punkte wie Bahnhöfe und Ampeln). Für die letzte Funktion wird `plot_symbol` verwendet, um das richtige Symbol für den Punkt zu zeichnen (z.B. ein S-Bahn-Zeichen oder ein Ampelsymbol).

Bei den Funktionen `plotstr`, `plot_point` und `plot_orte` wird der Programmcode für das eigentliche Zeichnen dynamisch vor Ausführung der Schleife erzeugt. Der Grund hierfür ist die Vermeidung der Doppelung von Programmcode, der auch bei der alternativen Methode des dynamischen Zeichnens des Straßennetzes (Seite 96) verwendet wird.

Um das Zeichnen schneller durchführen zu können, kann in `plotstr` und `plot_point` optional die XS-Entsprechung aufgerufen werden (siehe Seite 87).

Beim Zeichnen können optional zu den Kartenobjekten zugehörige Texte (Labels) erzeugt werden. Bei Strecken werden diese Labels an den Verlauf der Strecke angepaßt, bei Punkten wird die Bezeichnung normalerweise neben den Punkt gesetzt. Für das angepaßte Zeichnen von Labels ist es notwendig, daß Texte rotiert auf dem Canvas dargestellt werden können. Diese Funktionalität ist jedoch in Perl/Tk nicht enthalten. Moderne X11-Server (ab der Version X11R6) können auf Zeichensätze Matrixmanipulationen durchführen (siehe [Flo94]), so daß bei dieser Systemumgebung das Zeichnen von rotierten Zeichensätzen mit der Funktion `rot_text_old` emuliert werden kann. Bei Flächen wird das Label in die Mitte des Polygons gezeichnet. Für das Feststellen des Schwerpunktes wird die Funktion `get_polygon_center` verwendet. Diese Methode ist nicht perfekt, weil sich der Schwerpunkt eines Polygons durchaus außerhalb der Polygonfläche befinden kann.

Bei der Entwicklung ist es vorteilhaft, geänderte Dateien der Datenbasis einfach nachladen zu können. Mit `reload_all` wird anhand von Timestamps festgestellt, welche Koordinatendateien nachgeladen und neu gezeichnet werden müssen.

Bei der Anzeige der Kartenobjekte kann die Schichtung verändert werden. Das Anwenden der aktuellen Schichtung wird in der Funktion `restack` durchgeführt. Die aktuelle Schichtung kann mit `real_stack_order` abgefragt werden. Die Schichtungsreihenfolge von Kartenobjekten wird mit den Funktionen `special_raise` (Kartenobjekte werden über allen anderen Kartenobjekten gezeichnet), `special_lower` (Kartenobjekte werden unter allen anderen gezeichnet) und `special_normal` (die normale Schichtung für dieses Kartenobjekt wird angewendet) geändert. Diese Funktionen müssen immer auf zusammengehörige Gruppen von Kartenobjekten angewendet werden (so müssen z.B. Bahnstrecken und die dazugehörigen Bahnhöfe gemeinsam gehoben oder gesenkt werden). Als weitere Alternative für das Ändern der Schichtung kann das Modul `Tk::LayerEditor` verwendet werden.

Als zusätzliche Operationen existieren die Änderung des Kartenausschnitts, also das Verschieben (Scrolling und Zentrieren) und das Vergrößern und Verkleinern (Zooming/Skalieren).

Für das Zooming kann die Canvas-Methode `scale` verwendet werden. Allerdings ist diese Methode nicht perfekt, weil nur die Koordinaten der Kartenobjekte angepaßt werden, nicht aber Breite der Linien, Größe der Schrift oder Symbole. Deshalb wurde für BBBike eine neue Funktion (`scalecanvas`) geschrieben, die einige dieser Probleme behebt. In dieser Funktion werden weitere Funktionen aufgerufen: mit `show_zoomrect` wird nach dem Skalieren der alte Bereich der Karte optisch gekennzeichnet. `scale_width` paßt die Linienbreite und die Symbolgröße der Kartenobjekte an den neuen Maßstab an. Für die entsprechenden Werte werden die Funktionen `get_line_width` (Linienbreite), `get_line_length` (Länge der Pfeile und Kreuze für gesperrte Straßen) und `get_symbol_scale` (Symbole — einerseits Bilder wie für das Ampelsymbol oder zusammengesetzte Canvas-Objekte wie für die Bahnhofs-Symbole).

Für das manuelle Scrolling können die auf Seite 31 aufgeführten Methoden verwendet werden. Daneben kann der Kartenausschnitt auch automatisch verschoben werden, so z.B. bei der Auswahl einer Straße. Mit `is_visible` wird zurückgegeben, ob der übergebene Punkt im aktuellen Kartenausschnitt sichtbar ist. Mit den Funktionen `center_view` und `center_view2` wird der Kartenausschnitt auf den angegebenen Punkt zentriert. Die zweite Variante verschiebt den Kartenausschnitt zusätzlich so, daß möglichst viel in Richtung der zweiten übergebenen Koordinate zu sehen ist (dies kann beim Anzeigen einer Route angewandt werden). `zoom_view` verschiebt und skaliert den Kartenausschnitt so, daß die angegebenen Punkte vollständig sichtbar sind. Somit kann eine Route komplett angezeigt werden.

Interaktivität

Beim Bewegen der Maus über Kartenobjekte wird in der obersten Leiste und evtl. auch als Tooltip angezeigt, welche Straße oder Stelle das Kartenobjekt repräsentiert. Wenn die Maus in das Objekt hereinbewegt wird, werden die Funktionen `enterpoint` (für Punkte) oder `enterstr` (für Strecken) aufgerufen. Beim Verlassen von Objekten werden die entsprechenden Funktionen `leavepoint` und `leavestr` aufgerufen. Bei den `enter`-Funktionen wird festgestellt, welcher Name dem Objekt zugeordnet ist und dieser dargestellt. Bei gestapelten Kartenobjekten wird der Name des obersten Objekts angezeigt. Bei den `leave`-Funktionen wird der Name nur angegraut, so daß sichtbar wird, daß der Name nicht mehr gültig ist. Beim Bewegen der Maus über eine Route werden die Funktionen `enterroute` und `leaveroute` aufgerufen. Diese zeigen außer der darunterliegenden Straße auch die Fahrzeit und Streckenlänge bis zu diesem Routenpunkt an. Um bei gestapelten Canvas-Objekten in Tk auch die darunterliegenden Objekte feststellen zu können, existieren die Funktionen `find_below`, `show_below_str` und `show_below_route_str`.

Mit Klick auf eine Straße wird entweder der Start- oder der Zielpunkt einer Route festgelegt. Wenn noch kein Punkt ausgewählt wurde, wird `search_route_mouse` aufgerufen. Wenn eine Route über einen bestehenden Zielpunkt hinaus verlängert werden soll, wird `search_route_mouse_cont` aufgerufen. In beiden Funktionen wird bei Bedarf der Straßennetz-Graph neu berechnet und dann die Canvas-Koordinaten in die intern verwendeten Koordinaten des Straßennetzes umgewandelt. Wenn ein Zielpunkt gesetzt wird, wird die Suche mittels `search_route` gestartet. Wenn der Klick zu einer linearen Verlängerung der Route führt, wird nur `addpoint_inter` aufgerufen.

Start- und Zielpunkt müssen einem Knoten im Straßennetz-Graphen entsprechen, damit die Suche funktionieren kann. Normalerweise wird der Benutzer aber einen Punkt anwählen, der zwar auf der Straße liegt, aber kein Knoten des Graphen ist. Deshalb muß der Punkt nachträglich in den Graphen eingefügt werden. Erst einmal wird per `nearest_line_points` die exakte Position des Punktes festgestellt, dann wird die Methode `add_net` mit dem neuen Punkt aufgerufen. Die Umrechnung der Koordinaten wird mit den Funktionen `transpose` (interne Koordinaten auf Canvas-Koordinaten) und `anti_transpose` (Canvas-Koordinaten auf interne Koordinaten) durchgeführt. Diese Funktionen werden im Module `BBBikeTrans` dynamisch erzeugt.

Die Funktion `nearest_line_points` wurde eingeführt, weil die Zuordnung der Canvas- auf interne Koordinaten ungenau sein kann. Bei Tk-Canvas-Koordinaten werden Ganzzahlen verwendet, die bei einer skalierten Karte ungenau sein können. Um dieses Problem zu umgehen, wurde als erste Lösung für jedes einzelne Straßenstück ein eigenes `createLine`-Kommando abgesetzt, das über die *Tags* die internen (genauen) Koordinaten enthält. Wie aber auf Seite 94 beschrieben, war diese Methode zu langsam. Deshalb wird jetzt nur *ein* `createLine`-Kommando für einen Straßenzug verwendet. Für das Feststellen der exakten Position innerhalb des Straßenzuges mußte eine neue Funktion geschrieben werden, nämlich `nearest_line_points`.

Für das Suchen von Straßen und Orten über Namen wird die Dialog-erzeugende Funktion `choose_ort` aufgerufen. Ausgewählte Straßen und Orte können von dort aus markiert werden, d.h. das gesuchte Objekt wird auf der Karte zentriert und mit den Funktionen `mark_street` und `mark_point` mit einer temporären Markierung versehen. Das Objekt kann auch als Start- oder Zielpunkt einer Route definiert werden. Diese Funktionalität wird in `set_route_start` und `set_route_ziel` vorgenommen.

Bei der Auswahl einer Straße aus der STRADA-Datenbank oder eines Ortes kann es vorkommen, daß der ausgewählte Punkt nicht auf einer Straße der Datenbasis liegt und die oben genannte Funktion nicht angewendet werden kann. In diesem Fall wird mittels der Methode `fix_coords` der nächstgelegene Punkt gesucht und verwendet.

Wenn die `followmouse`-Option eingeschaltet ist, wird festgestellt, ob sich die Maus an den Rändern der Karte befindet. In diesem Fall wird ein automatisches Scrolling durchgeführt. Mit der Funktion `start_followmouse` wird die Ausführung der Option eingeschaltet, mit `stop_followmouse` wieder ausgeschaltet. Bei der Aktivierung wird eine sich alle 0,1 Sekunden wiederholende Funktion ausgeführt, die überprüft, ob sich die Maus im Bereich zum Scrollen befindet. Ist das der Fall, wird das Scrollen durchgeführt. Wenn die Maus nach oben verschoben wird, gibt es noch eine 0,3 Sekunden währende Verzögerung, weil der Benutzer evtl. nur das Menü erreichen wollte. Eine alternative Scrollmöglichkeit besteht mit der Verwendung des `Tk::Autoscroll`-Moduls.

Suche

Die eigentliche Suche wird vom später beschriebenen Modul `Strassen.pm` (Seite 71) bewerkstelligt. In der Anwendung selbst wird die Suche vorbereitet, also die Parametrisierung der Suche in der Funktion `search_route` zusammengetragen, und dann die Suchfunktion angestoßen.

Vor Starten der Suche muß der Straßennetz-Graph erzeugt werden. Die Funktion `make_netz` übernimmt diese Aufgabe. Dabei wird festgestellt, welche Kartenobjekte in den Graphen aufgenommen werden sollen und schließlich wird `make_net` in `Strassen.pm` aufgerufen.

Die Funktion `calc_ampel_optimierung` berechnet anhand der eingestellten Geschwindigkeit und statistisch ermittelten Werten für die Länge der Ampelphasendauer die durchschnittlich verlorene Zeit und die daraus resultierende Strecke. Zur Zeit ist diese Berechnung wenig differenziert. In einem ersten Verfeinerungsschritt könnte zwischen den Ampelphasen an Haupt- und Nebenstraßen sowie zwischen den Ampelphasen im Tages-, Berufs- und Nachtverkehr unterschieden werden. In einem weiteren Schritt könnten individuelle Ampelphasen berücksichtigt werden. Der berechnete Wert kann bei der Suche verwendet werden.

Mit `re_search` wird eine Suche vom aktuellen Start- zum Endpunkt wiederholt. Falls sich Optimierungsattribute geändert haben, kann die neue Suche zu einer geänderten Route führen. `way_back` berechnet den Rückweg der aktuellen Route. Für diese beiden Funktionen werden die Start-, Via- und Endpunkte im Feld `@search_route_points` gehalten.

Routen

Das Ergebnis einer Suche muß auf der Karte dargestellt werden. Mit `addpoint_xy` werden nacheinander die Koordinaten des Suchergebnisses für das Zeichnen der Route verwendet. Dabei wird festgestellt, welche Windrichtung auf dem Routenteilstück herrscht und dieses Teilstück entsprechend eingefärbt. Wenn das Einfärben nach der erforderlichen Leistung gewünscht wird, wird stattdessen das Verhältnis zwischen gewünschter Geschwindigkeit und leistungsbezogener Geschwindigkeit als Grundlage für das Einfärben verwendet. Weiterhin werden in dieser Funktion Gefälle und Steigungen festgestellt und bei Bedarf neben dem Teilstück dargestellt. Das Berechnen der Steigung ist etwas umfangreicher: da nicht jedem Knoten des Graphes ein Höhenpunkt zugeordnet ist, muß für das Feststellen einer Steigung die bisherige Route bis zum vorherigen Höhenpunkt untersucht werden. Beispiel: Es existiert eine Route A-B-C, wobei Höhenpunkte in A und C definiert sind. Die Funktion muß hier die durchschnittliche Steigung zwischen den Punkten A und C feststellen.

Die Funktion `addpoint_inter` wird aufgerufen, wenn der Benutzer einen Punkt zur Route hinzufügt, der die Route linear vom letzten Punkt fortsetzen soll (d.h. es soll keine Suche angestoßen werden). In diesem Fall wird `addpoint_xy` und dann die Aktualisierungsfunktion `updatekm` aufgerufen. `updatekm` aktualisiert die Anzeige für die Streckenlänge, die Fahrzeit und die Anzahl der Ampeln an der Route.

`redraw_path` wiederholt das Zeichnen einer Route. Mit `resetroute` wird eine Route von der Karte gelöscht. `reverse_route` dreht eine Route um. Im Gegensatz zu `way_back` wird dabei keine neue Suche aufgerufen; die Route kann somit (wegen Einbahnstraßen) nicht legal oder (wegen unterschiedlicher Attributierung in beiden Fahrtrichtungen) nicht die optimale Route sein.

Teilstücke der Route können mit `deltovia` und `dellast` gelöscht werden. Dabei löscht `dellast` nur den letzten Punkt der Route, während `deltovia` die Route bis zum letzten Via-Punkt löscht.

Die Umwandlung der Routenkoordinaten in eine textuelle Beschreibung wird in `show_route_strname` vorgenommen. In dieser Funktion wird das Fenster für die Ausgabe erzeugt und dann die Methode `route_to_name` aufgerufen, die die entsprechenden Straßennamen mit Etappenlängen für das Teilstück und dem Winkel für den nächsten Abbiegevorgang zurückgibt. In `show_route_strname` werden diese Informationen aufbereitet, indem der Winkel in einen „rechts/links“-

Ausdruck übersetzt wird und versucht wird, mittels `de_artikel` die richtige grammatikalische Form für „in die“ zu finden. Bei geöffneten Beschreibungsfenster wird beim Erweitern oder Ändern einer Route die textuelle Beschreibung sofort angepaßt.

Gefundene Routen können zwischenzeitlich in ein Register geschrieben werden. Die Anzeige des Registerinhalts wird in `show_register` vorgenommen. Das Einfügen in das Register und das Auslesen aus dem Register wird in den Funktionen `save_route_to_register` und `get_route_from_register` durchgeführt.

Die Funktion `bikepwr_get_v` stellt das Bindeglied zum Modul `BikePower` her, wobei die aktuelle Leistung, Windstärke und Steigung übergeben wird und die Geschwindigkeit zurückgeliefert wird.

I/O

Für das Laden und Speichern von Routen wird einfach das Modul `Data::Dumper` verwendet. Dieses Modul enthält Funktionen, um Perl-Daten persistent zu halten. Beim Abspeichern wird das Feld mit den Routenkoordinaten als Perl-Array gespeichert. Für das Laden kann der `do`-Operator verwendet werden oder, um möglichst sicher zu gehen, die `rdo`-Methode des `Safe`-Moduls[24]. Als weiteres Format für das Laden kann auch eine Track-Datei von GPS-Geräten verwendet werden. Somit können bereits gefahrene Routen auf der Karte dargestellt werden.

Die Liste mit den zuletzt geladenen und gespeicherten Routen-Dateien wird mit den Funktionen `add_last_loaded`, `load_last_loaded` und `save_last_loaded` verwaltet.

Die textuelle Beschreibung einer Route kann auch gespeichert, gedruckt oder als E-Mail verschickt werden. Für die gedruckte Form wird erst einmal mit dem Modul `Tk::Enscript` Postscript erzeugt, der dann an das externe Programm Ghostview, Ghostscript oder GV weitergeschickt wird, mit denen gedruckt werden kann. Die Umwandlungsfunktionen befinden sich im Modul `BBBikePrint`. Das Verschicken von E-Mails wird im Modul `BBBikeMail` vorgenommen, das wiederum das CPAN-Modul `Mail::Send`[25] aufruft.

Wetterdaten

Das Laden und Parsen der Daten aus dem Internet wird mit dem später beschriebenen Modul `wettermeldung2` (Seite 85) vorgenommen. In der Anwendung wird die Auswahl der gewünschten Quelle (Tempelhof oder Dahlem) und das Anstoßen der Funktionen in `wettermeldung2` vorgenommen. Mit `show_weather_db` kann eine lokal vorhandene Wetterdatenbank, falls vorhanden, angezeigt werden. Der

[24] Mit dem `Safe`-Modul kann die Ausführung von bestimmten, möglicherweise unsicheren Perl-Opcodes verhindert werden.

[25] `Mail::Send` unterstützt unter Windows nicht das Versenden von E-Mails. Das Modul kann unter `<URL:http://www.cpan.org/authors/id/GBARR>` gefunden werden.

Benutzer kann dann einen Eintrag eines bestimmten Datums für die aktuellen Wetterdaten auswählen.

In der Funktion `enter_wind` kann die Eingabe der Windgeschwindigkeit und -stärke manuell erfolgen. Die Eingabe der Windstärke erfolgt entweder über ein Scale-Widget oder über das Modul `Met::Wind`, das eine Tabelle mit der Zuordnung von Windstärken, -geschwindigkeiten und einer textuellen Beschreibung anzeigt.

Sonstiges

Mit `delayed_sub` können Aktionen verzögert ausgeführt werden. Der Vorteil hierbei ist, daß eine abschließende Aktion, die evtl. zeitaufwendig ist, erst ausgeführt wird, wenn alle vorangegangen Aktionen fertig sind. Die Verzögerung in `delayed_sub` wird verwendet um festzustellen, ob keine weiteren vorangehenden Aktionen durchgeführt werden müssen. Diese Funktion wird beim Verändern von Attributen von Kartenobjekten aufgerufen.

3.3.3 Installation

Im Gegensatz zu gewöhnlichen Programmpaketen ist eine Installation der Anwendung nicht notwendig. Nach dem Auspacken des BBBike-Archivs kann — soweit sich bereits eine komplette Perl5-Installation[26] sowie eine Perl/Tk-Distribution[27] auf dem Zielsystem befinden — das Programm sofort benutzt werden, weil alle Modul- und Datenabhängigkeiten auf relativen Pfaden zum Hauptprogramm beruhen.

Trotzdem gibt es eine Installationsroutine, die folgende Dinge erledigt:

- Überprüfen, ob die nötigen Perl-Module installiert sind (d.h. die Existenz von `Tk.pm` wird getestet).
- Eintrag der Anwendung in das Start-Menü des Window-Systems
- Erzeugen eines Icons für die Anwendung auf dem Desktop
- Sonstige Einstellungen wie das Anmelden eines MIME-Typs für die Anwendung
- Kompilierung der XS-Erweiterungen

Bei den windowsystemspezifischen Features wird MS Windows sowie bei Unix das KDE[28]-System unterstützt.

[26] mindestens Version 5.004
[27] mindestens Version 402.004
[28] `<URL:http://www.kde.org>`

Bei Windows wird das Modul `Win32Util` verwendet, um die nötigen Einträge zu erzeugen. Mit `Win32Util::install_extension` wird die Dateinamenerweiterung `.bbr` als BBBike-Routen-Datei gekennzeichnet. Damit wird beim Klick auf eine `bbr`-Datei automatisch die Anwendung gestartet und die `bbr`-Datei als Route geladen. Mit `Win32Util::create_program_group` wird eine neue Programmgruppe im Start-Menü mit dem Namen „BBBike" erstellt. Die Programmgruppe enthält Einträge für den Start der Anwendung, einen Link auf die WWW-Version im Internet, die Dokumentation (die HTML-Version), die Konsolenversion von BBBike und BikePower. Zum Schluß wird mit `Win32Util::create_shortcut` ein Desktop-Icon für den Start von BBBike erzeugt.

Für die KDE-Installation kann zwischen einer „Root"- und einer „Benutzer"-Installation (Typ „user") gewählt werden. Bei der Root-Installation muß zuerst nach den offiziellen KDE-Verzeichnissen gesucht werden. Dies erfolgt in der Funktion `find_kde_dirs`. Bei einer Benutzerinstallation werden die KDE-Verzeichnisse im Heimatverzeichnis verwendet. Beim Installationstyp „best" wird wenn möglich eine „root"-, ansonsten eine „user"-Installation vorgenommen. Im `applnk`-Verzeichnis wird ein KDE-Link auf das BBBike-Programm angelegt. Im `mimelnk`-Verzeichnis werden die Pseudo-Mime-Typen ([Fre96]) `x-bbbike-route` und `x-gistrack` angelegt. Die `magic`-Datei von KDE wird um diese beiden Mime-Typen erweitert. Weiterhin werden die BBBike-Icons in das `icons`-Verzeichnis installiert. Zum Schluß wird noch ein Desktop-Icon erzeugt.

Nachdem die windowsystemspezifische Installation vorgenommen wurde, wird versucht, die XS-Erweiterungen[29] zu kompilieren. Bei Nicht-Existenz eines C-Compilers oder bei sonstigen Fehlern wird der Benutzer davon unterrichtet; die Installation wird dadurch aber nicht abgebrochen, weil die XS-Erweiterungen optional sind.

3.4 WWW-Schnittstelle

Die WWW-Schnittstelle wurde als CGI-Programm (`bbbike.cgi`) realisiert. Das hat den Vorteil, daß als Programmiersprache auch Perl verwendet werden kann und somit die Module für die Datenzugriffe und die Suche mitverwendet werden können. Nachteilig ist, daß mit CGI-Programmen keine hohe Interaktivität wie z.B. mit Java-Applets erreicht werden kann. Anderseits ist der daraus resultierende Ansatz, nämlich ein weitgehend textbasiertes System, grafisch zwar nicht sehr ansprechend, dafür aber durch die schrittweise Führung durch das Programm besonders leicht verständlich. Die Perl/Tk-Anwendung hingegen ist für den Benutzer, der sich im Stadtplan Berlins schnell zurechtfindet, idealer, da er wesentlich schneller Start- und Zielpunkt eintragen kann.

[29] Mit XS ist es möglich, C-Programme für den Aufruf als Perl-Funktionen zu schreiben. XS steht für „eXternal Subroutines"

Nichtdestotrotz ist auch die grafische Auswahl in der WWW-Schnittstelle möglich. Die Anzeige des Stadtplans erfolgt hier nicht vektororientiert, sondern anhand von vorgebauten GIF- oder PNG-Bildern, die mittels des `GD.pm`-Moduls aus der Datenbasis erstellt wurden[30]. Eine gefundene Route läßt sich mit dem gleichen Modul dynamisch in den Stadtplan einbetten.

Ein Nachteil bei der Verwendung von reinen CGI-Programmen ist der Overhead beim Start, da das CGI-Programm für jede HTTP[31]-Verbindung gestartet werden muß. Da bei BBBike alle Straßendaten in den Hauptspeicher geladen werden, sind die Antwortzeiten dementsprechend schlecht[32]. Dieser Nachteil kann durch die Verwendung von „persistenten" CGI-Programmen mittels Modulen wie `FCGI` oder `CGI::MiniSvr` entschärft werden.

3.4.1 Ablauf des CGI-Programms aus Benutzersicht

Beim Starten sieht der Benutzer einen kurzen Erklärungstext und darunter drei Bereiche für die Eingabe der Start-, Via- und Zielstraße (Abbildung 3.6). In jedem dieser drei Bereiche kann man zwischen mehreren Eingabealternativen wählen: Eingabe der Straße im Textfeld, Anklicken eines Buchstabens für die Anzeige einer Liste der bekannten Straßen oder Anklicken der kleinen Berlin-Karte für das grafische Auswählen einer Straße.

In der unteren Leiste befindet sich ein Link „Einstellungen". Hier können fahrradspezifische Parameter permanent eingestellt werden. Die Parameter werden für die Leistungsberechnung verwendet.

---

[30] Die neuesten Versionen der `libgd`-Bibliothek und des `GD.pm`-Moduls können wegen lizenzrechtlichen Problemen nur das Format PNG erzeugen (siehe [lgd99] und [Uni99]).

[31] HTTP: Hypertext Transfer Protocol ([HTT96], [HTT99])

[32] Bei einem PC mit AMD-K5-Prozessor (166 MHz) als WWW-Server beträgt die Antwortzeit zwischen 4 und 10 Sekunden pro Seitenaufbau

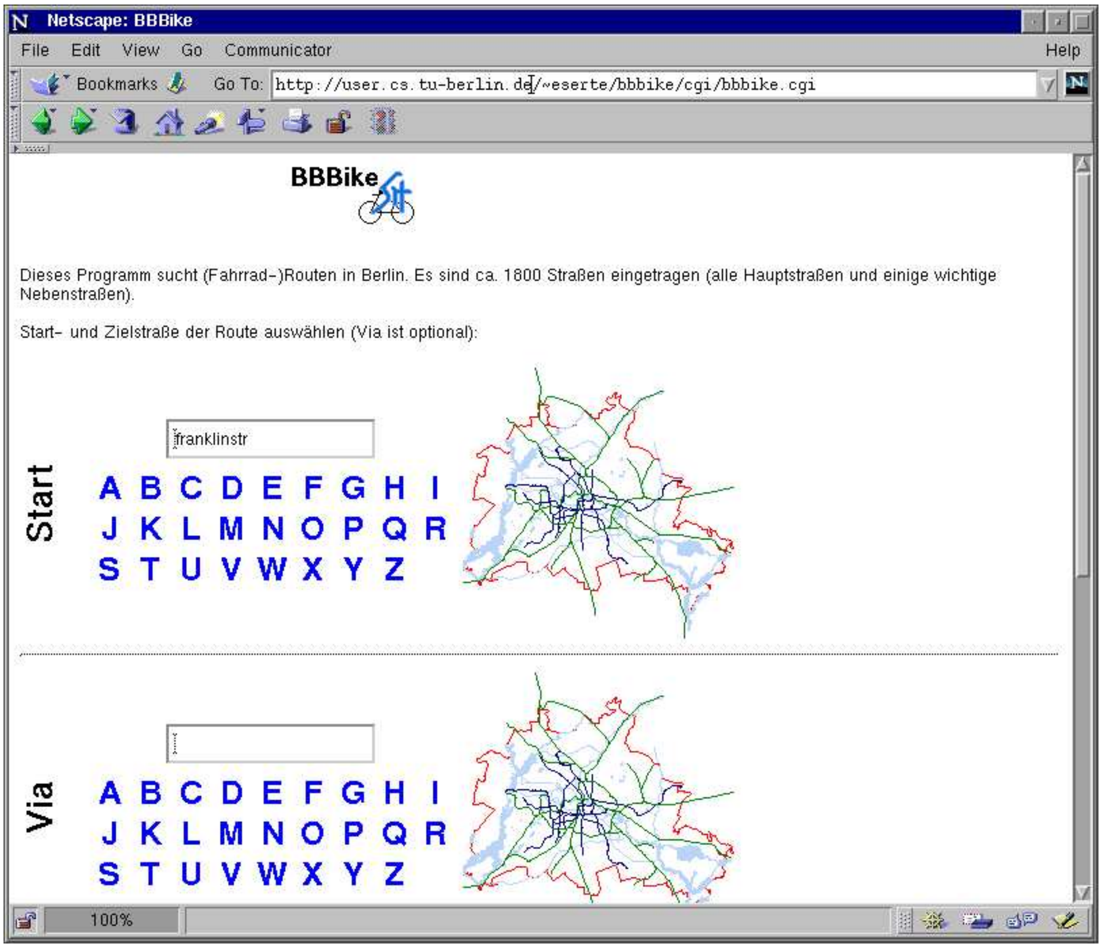

Abbildung 3.6: Startbildschirm von bbbike.cgi

Nachdem Start- und Zielstraße eingegeben wurden (die Viastraße ist optional) kann mit Klick auf „Weiter“ die Bearbeitung fortgesetzt werden.

Wenn der Benutzer die Straße im Textfeld eingegeben hat, ist es möglich, daß die Straße fehlerhaft eingegeben wurde oder die Straße in der Datenbasis nicht vorhanden ist[33]. In diesem Fall wird im nächsten Schritt noch einmal nachgefragt. Dabei wird bei einer fehlerhaften Eingabe eine Liste von ähnlich klingenden Straßennamen vorgelegt und bei gültigen, aber nicht in der Datenbasis vorhandenen Straßennamen eine Liste der Straßen in der näheren Umgebung erzeugt.

[33]In der Datenbasis sind nur ca. 2000 der über 11000 Berliner Straßen aufgeführt.

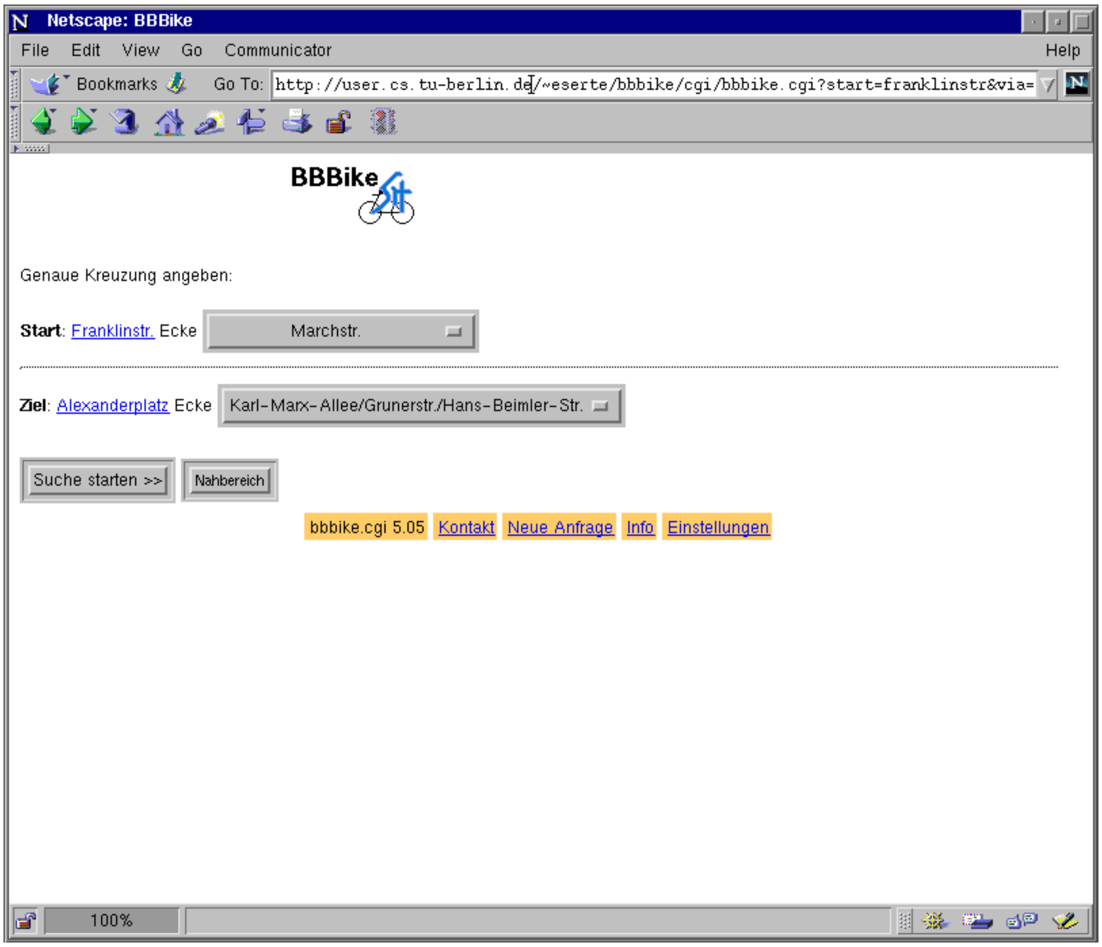

Abbildung 3.7: Kreuzung auswählen

Als letzter Schritt ist die Angabe der genauen Kreuzung erforderlich (siehe Abbildung 3.7). Eine Eingabe der Hausnummer wäre wünschenswert, ist aber mit der derzeitigen Datenbasis nicht machbar.

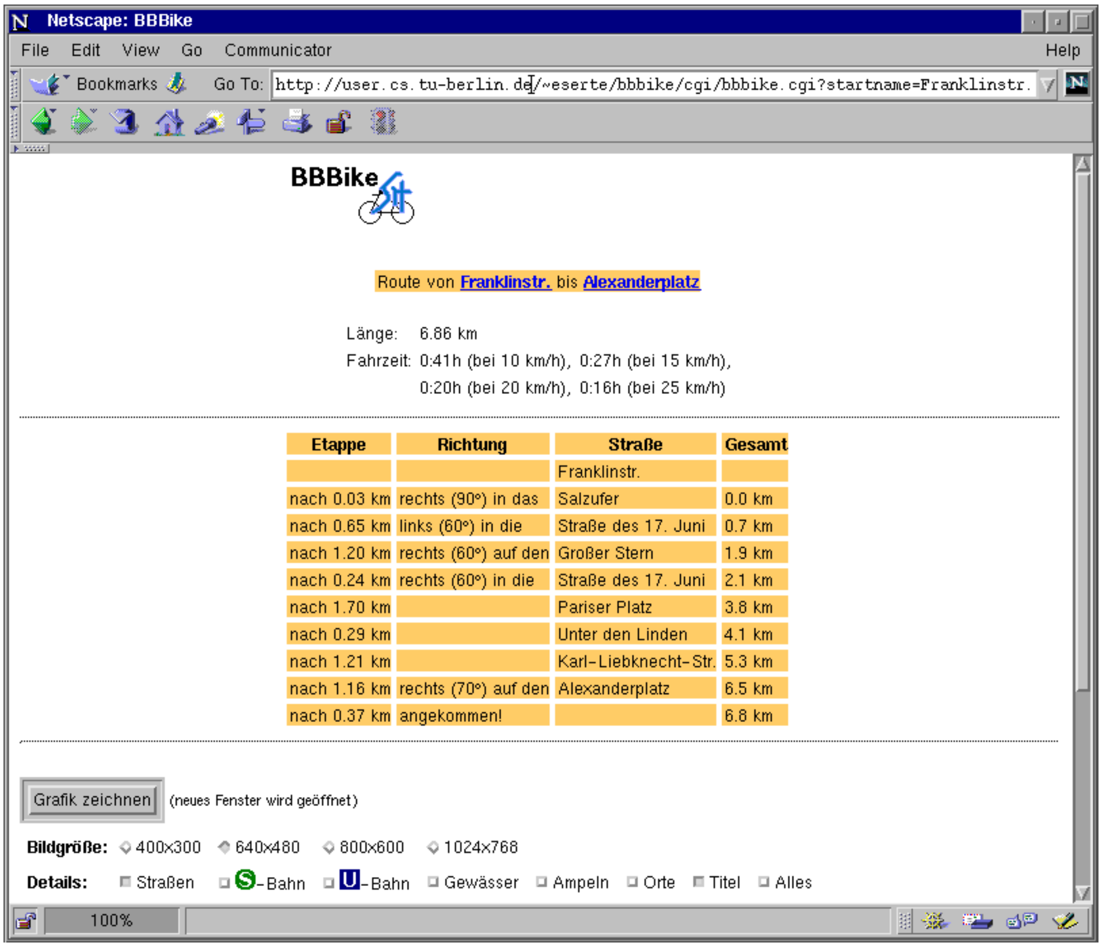

Abbildung 3.8: Anzeige der Route als Text

Das Ergebnis der Suche wird als Liste der zu befahrenden Straßen mit Etappenstrecke, Gesamtstrecke und Hinweisen für das Abbiegen dargestellt (Abbildung 3.4.1). Zusätzlich wird die veranschlagte Fahrzeit bei einigen Durchschnittsgeschwindigkeiten (10 km/h, 15 km/h, 20 km/h und 25 km/h) angezeigt. Wenn die Parameter in den Einstellungen eingegeben wurden und Windparameter aus der Wettermeldung vorliegen, wird auch die Fahrzeit für einige Durchschnittsleistungen (50 Watt, 100 Watt und 200 Watt) angezeigt.

Das Suchergebnis kann mit Klick auf den Knopf „Grafik zeichnen“ als Route auf einem dynamisch generierten Stadtplan angezeigt werden (siehe Abbildung 3.9). Für den Stadtplan kann die gewünschte Größe von 400x300 bis 1024x768 Pixeln und die gezeichneten Details (Straßen, U- und S-Bahnen, Gewässer, Ampeln…) angegeben werden.

Als weitere Option kann der Rückweg berechnet werden. Falls vom CGI-Programm eine Verbindung zum Meteorologischen Institut der FU Berlin geschaffen wurde, wird die aktuelle Temperatur, Windrichtung und -geschwindigkeit angezeigt.

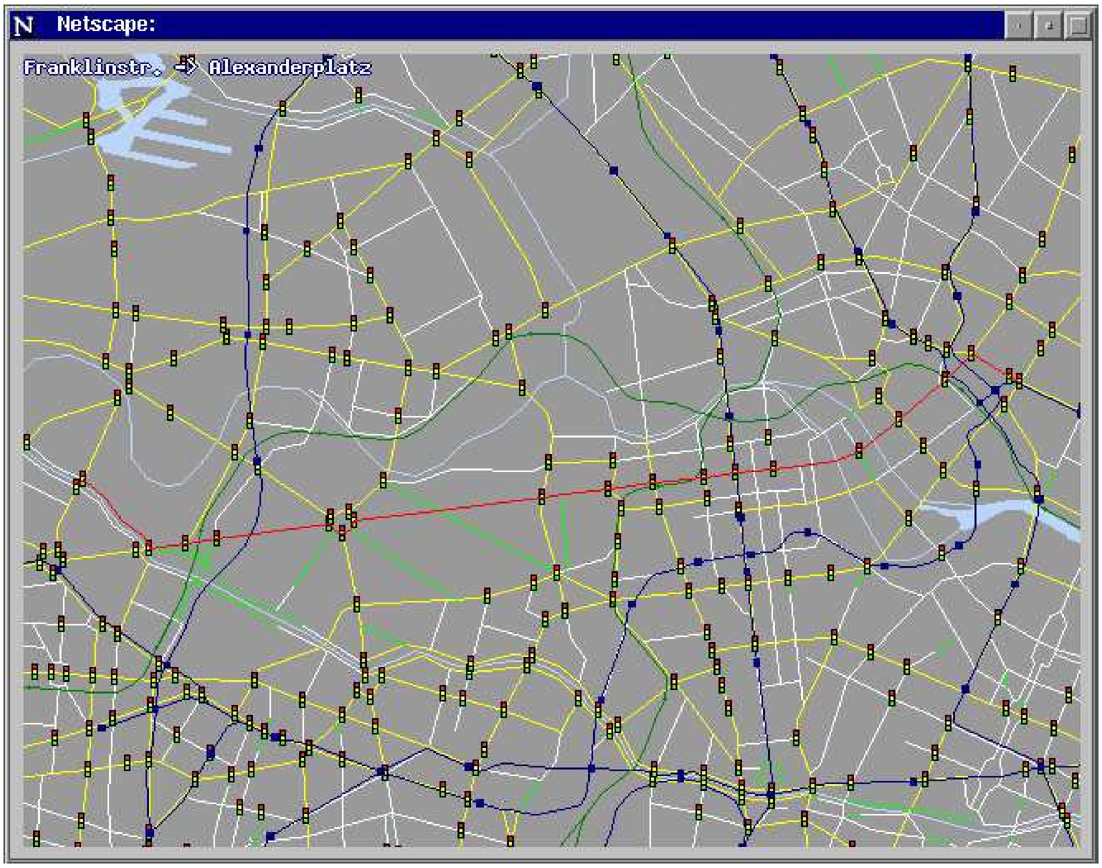

Abbildung 3.9: Anzeige der Route als Grafik

Als Hilfe beim Eingeben der Straßen steht der Link „Alle Straßen anzeigen“ zur Verfügung. Beim Klick auf diesen Link wird ein Fenster mit der Liste aller Straßen angezeigt. Straßen können dann per „Cut-and-paste“ in das Eingabefeld überführt werden.

Beim Anklicken eines Buchstabens im Buchstabenfeld des Startformulars wird eine Liste der Straßen, die auf diesen Anfangsbuchstaben beschränkt sind, angezeigt (siehe Abbildung 3.10). Mit Klick auf die nebenstehende Checkbox wird die Straße als Start- oder Zielstraße übernommen.

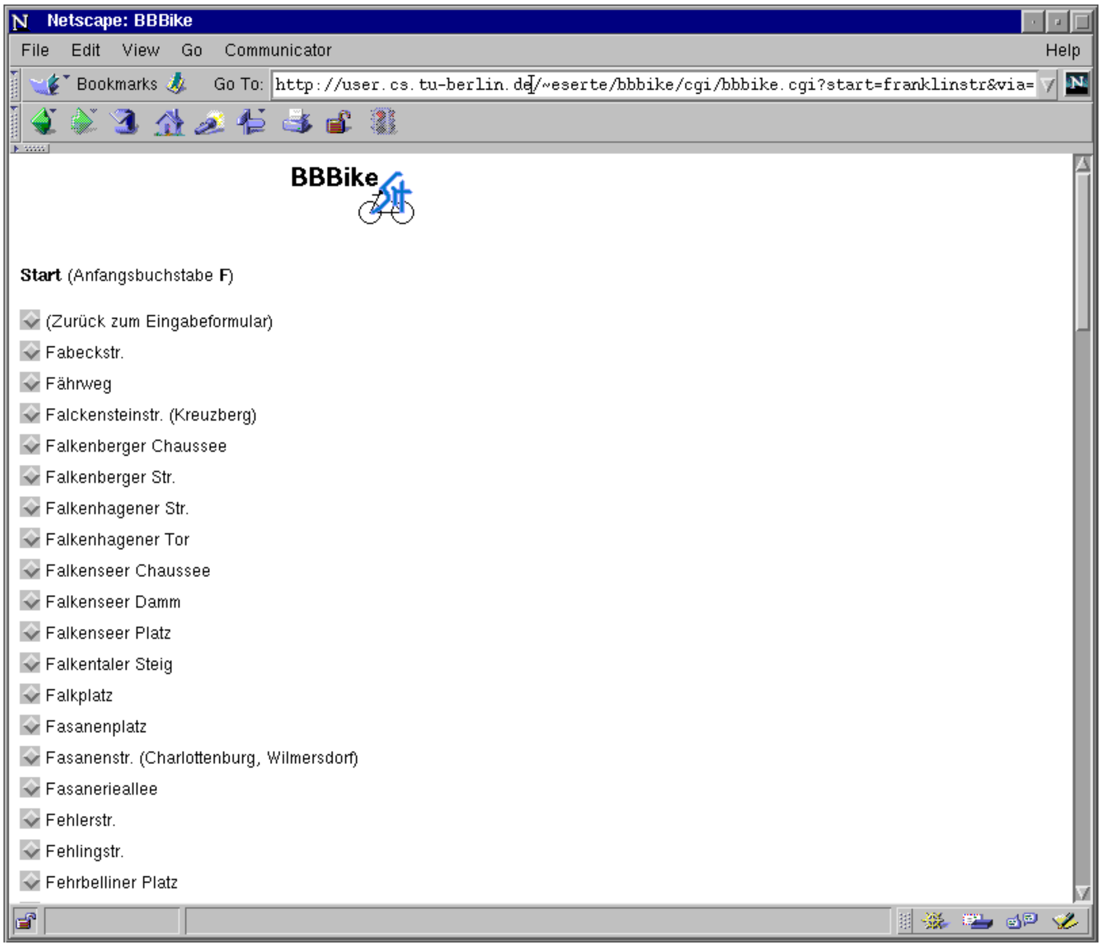

Abbildung 3.10: Straßenliste nach Buchstaben geordnet

Beim Klick auf die Berlin-Karte im Startformular wird der entsprechende Quadrant als einfache Karte angezeigt (Abbildung 3.11). Mit Klick auf eine Straße oder Kreuzung werden die jeweiligen Koordinaten übernommen. Nachteilig ist bei dieser Form, daß auf der Karte kein Text, also insbesondere keine Straßennamen, vorkommen. Somit ist diese Variante nur für Benutzer sinnvoll, die gute Kenntnisse der Karte von Berlin aufweisen. Bei einigen Browsern ist es allerdings möglich, sich die Straße unter dem Mauszeiger anzeigen zu lassen. Vorteilhaft ist bei dieser Methode, daß mit der Maus sofort die Kreuzung ausgewählt wird und somit der Schritt der Kreuzungswahl wie bei der konventionellen Eingabemethode entfällt.

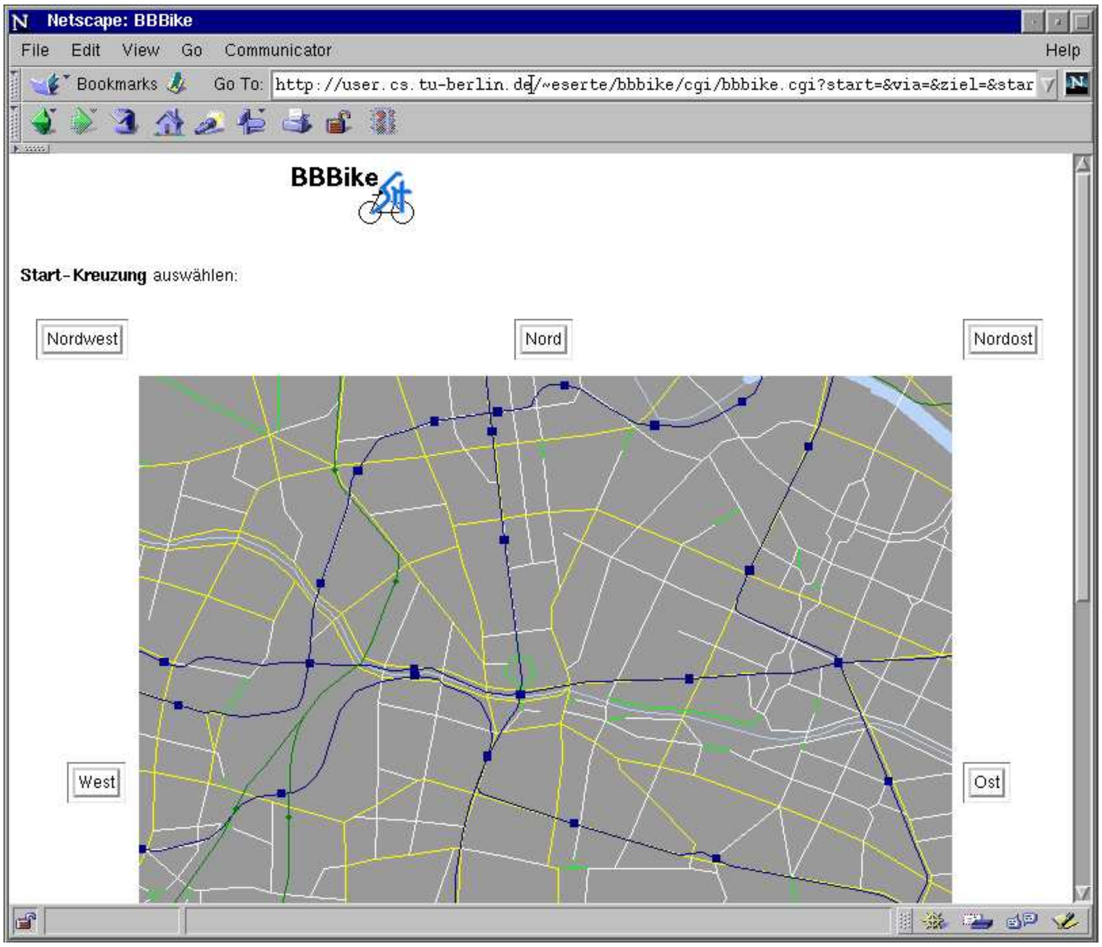

Abbildung 3.11: Eingabe über die Karte

3.4.2 Implementierung

Verschiedene CGI-Interface-Formen

Wie bereits erwähnt, hat CGI den Nachteil, daß bei jeder HTTP-Verbindung das CGI-Programm neu gestartet werden muß. Bei in Perl geschriebenen CGI-Programmen muß bei jedem Start das Programm neu geparst werden[34]. Weiterhin muß bei Bedarf die Datenbasis geladen werden und vor der Suche der Straßennetz-Graph erzeugt werden. Um diese Probleme zu umgehen, gibt es Möglichkeiten, um CGI-Programme persistent zu halten.

Folgende CGI-Interface-Formen wurden im Perl-Programm realisiert:

CGI.pm: Das `CGI.pm`-Modul wird in der Perl-Distribution mitgeliefert und ist die gewöhnliche Form der CGI-Programmierung mit Perl. Dieses Modul

[34] Das Parsen dauert auf einem AMD-K6 mit 166 MHz ca. 1,5 Sekunden

bietet keine Persistenz des CGI-Programms.

FCGI: Mit dem `FCGI`-Interface ([Fcg99]) wird für jedes CGI-Skript ein Prozeß gestartet und persistent gehalten. Im Gegensatz zum `CGI.pm`-Interface existiert hier eine `while`-Schleife, die für jeden HTTP-Aufruf durchlaufen werden muß. Dadurch entfällt das Parsen des CGI-Skriptes (außer einmalig zum Start); außerdem müssen alle externen Daten, also die Datenbasis der Straßen sowie das generierte Straßennetz, nur einmal eingelesen bzw. erzeugt werden und sind für den Rest der Lebensdauer des WWW-Servers sofort verfügbar. Anderseits ist es nachteilig, daß über die ganze Zeit hinweg der Speicher belegt bleibt. Außerdem blockiert ein laufender Prozeß alle anderen, so daß zeitlich aufwendige oder zahlreiche Suchanfragen zu einer Verlangsamung des Systems führen.

Apache::Registry: Eine spezielle Lösung für das weitverbreitete WWW-Serverprogramm Apache ([Apa99]) existiert mit der `modperl`-Erweiterung ([Mpl99]). Diese Erweiterung ermöglicht die enge Kooperation zwischen Perl-Programmen und dem WWW-Server. Bei der Anwendung der `modperl`-Erweiterung sind keine weiteren Änderungen an bestehenden Perl-CGI-Skripten erforderlich. Die `CGI.pm`-Lösung kann also weiterverwendet werden. Die Geschwindigkeitssteigerung kann entweder durch das Einbinden des Perl-Interpreters in den Apache-Web-Server erreicht werden (dadurch wird der Overhead durch das Starten des Perl-Interpreters minimiert) oder durch das Einbinden eines Perl-Moduls (dadurch wird der Overhead des Parsens des Perl-Moduls minimiert). Im Fall von BBBike wird der Overhead hauptsächlich durch das Laden der Daten verursacht, so daß sich dieses Modul hier nicht lohnt.

CGI::MiniSvr: ist eine interessante Alternative zu den beiden anderen Varianten, weil diese keine Neuübersetzung des WWW-Serverprogramms erfordert. Der Ansatz von `CGI::MiniSvr` ist folgender: bei einer neuen Anfrage wird ein neuer Prozeß erzeugt, der einen kleinen HTTP-Server („MiniSvr“) sowie das CGI-Programm enthält. Der HTTP-Server wird über eine neue Port-Nummer ereicht. Wie bei `FCGI` läuft das Programm in einer `while`-Schleife, bis der Timeout erreicht wird. Dann wird der Prozeß terminiert. Vorteilhaft ist hierbei, daß jede Verbindung einen eigenen Prozeß besitzt, der andere Verbindungen nicht blockiert. Dieser Vorteil kann allerdings auch zu einem Nachteil werden, da mehrere Prozesse den virtuellen Speicher des Servers auffüllen können. Deshalb kann die Anzahl der BBBike-Prozesse in diesem Modus begrenzt werden. Außerdem muß nach einiger Zeit der Prozeß terminiert werden, so daß Benutzer, die eine längere Pause beim Arbeiten mit dem CGI-Skript machen, eine Fehlermeldung erhalten, daß der Server nicht mehr erreichbar ist. Nachteilig ist auch, daß die Einstellungen in diesem Modus nicht permanent gespeichert werden können, weil bei

Cookies unterschiedliche Portnummern als unterschiedliche Datenquellen interpretiert werden.

In der Voreinstellung wird das `CGI.pm`-Interface verwendet. Wenn die Dateiendung des CGI-Programms auf .fcgi endet, wird das `FCGI`-Interface benutzt. Mit Setzen der Perl-Variablen `$use_miniserver`wird das `CGI::MiniSvr`-Interface verwendet[35]. `Apache::Registry` wird automatisch verwendet, wenn `modperl` verwendet wird.

Ermittlung der Eingabeparameter

Um die Suchfunktion aus `Strassen.pm` verwenden zu könnnen, müssen die Koordinaten der Start-, Ziel- und evtl. auch der Via-Straße ermittelt werden. Der Benutzer gibt gewöhnlich die Straßennamen ein, die möglicherweise fehlerhaft sein können. Es erfordert einen iterativen Prozeß, um zunächst den richtigen Straßennamen zu ermitteln, daraus einen in der Datenbasis bekannten Straßennamen herauszufiltern und schließlich über die Kreuzungsermittlung die genauen Koordinaten zu berechnen.

Bei diesem iterativen Prozeß werden die bereits ermittelten Daten als Formularparameter (bei CGI-Programmen werden diese als „Querystring“ bezeichnet) an das nächste Formular übergeben. Alle Formulare sind in einer Programmdatei implementiert; allein anhand der übergebenen Formularparameter wird erkannt, welches Formular angezeigt werden muß.

In der Funktion `choose_form` wird dieses schrittweise Ermitteln bewerkstelligt. Im Anfangszustand wurden noch keine Eingaben vorgenommen und es wird das Startformular angezeigt. Nach dem Eintippen der Straßennamen werden zunächst die Querystring-Parameter `start`, `via` und `ziel` gesetzt und das CGI-Skript neu geladen. Diesmal wird in `choose_form` erkannt, daß Eingaben vorgenommen wurden. Für diese wird erst einmal überprüft, ob die Straße in der STRADA-Liste der bekannten Straßen in Berlin vorhanden ist. Dabei wird die Methode `look_loop` im Modul `PLZ.pm` verwendet. Diese Methode geht bei der Suche inkrementell vor, d.h. zunächst wird nach der exakten Schreibweise gesucht, bei Mißerfolg wird ein Fehler erlaubt, und falls noch immer keine Straße gefunden wurde, wird mit approximativer Suche bis zu einer Maximalanzahl von drei Fehlern gesucht[36]. Als letzte Maßnahme wird noch in den Dateien `plaetze` und `strassen` überprüft, ob die Straße dort vorkommt, da einige Straßen und Plätze nicht in der STRADA-Datei vorkommen.

Wenn bei dieser Suche mehr als ein Straßenname gefunden wurde, wird dem Benutzer die Trefferliste vorgelegt und er muß sich eine Straße aus der Liste

[35] Perl-Variablen zum Konfigurieren des CGIs können in der Konfigurationsdatei `bbbike.cgi.config` eingestellt werden.

[36] Fehler meint hierbei die Anzahl von Einfüge-, Lösch-, Vertausch- und Ersetzungsoperationen, die im Damerau-Levenstein-Maß definiert sind. Der Ausdruck dafür in `agrep` ist *Fehler* (siehe [Wu91] und [Wu92])

aussuchen.

Wenn eine Straße in der STRADA-Datei, aber nicht in der Straßen-Datenbasis vorkommt, wird die Funktion `choose_street_html` aufgerufen. Diese Funktion stellt alle Straßen mit der selben Postleitzahl wie die gesuchte Straße im Auswahlformular dar. Somit kann der Benutzer eine bekannte Straße in der Nähe der gesuchten Straße auswählen.

Schließlich wird eine in der Datenbasis bekannte Straße in den Querystring-Parametern `startname` und `zielname`[37] weitergegeben.

Nachdem der Straßenname vom Benutzer genau festgelegt wurde, muß nun die genaue Position des Start- oder Zielpunktes festgestellt. Idealerweise würde das über Hausnummern gehen, da aber die Daten dafür nicht vorhanden sind, wird einfach nach einer Kreuzung, die auf dieser Straße liegt, nachgefragt. Zunächst werden für die angegebenen Straßen die Indices im `Strassen`-Objekt festgestellt werden. Die Indices werden in den Feldern `@start_coords`, `@via_coords` und `@ziel_coords` gespeichert. Für den Fall, daß nur ein Index vorhanden ist, braucht keine Kreuzung ausgewählt zu werden und es kann gleich mit der Suche fortgesetzt werden. Ansonsten wird eine Liste mit den Kreuzungen ausgegeben. Der Benutzer hat nun die Möglichkeit, per Checkbox die nächste Kreuzung auszuwählen und die Suche anzustoßen.

Nachdem die Kreuzung ausgewählt wurde, werden die Querystring-Parameter `startc` und `zielc`[38] mit den exakten Koordinaten belegt. Mit diesen Koordinaten kann die Suche gestartet werden.

In `choose_form` wird auch der HTML- und Javascript-Code für die kleine Berlin-Karte ausgegeben. Bei textorientierten Browsern wird dieser Code nicht ausgegeben, bei grafisch orientierten wird die Karte als Imagemap ausgegeben und bei DHTML-fähigen Browsern als Imagemap mit Highlightning-Funktion.

Weitere Eingabemöglichkeiten

Mit der Funktion `choose_ch_form` wird für den angegebenen Anfangsbuchstaben eine Liste der Straßen ausgegeben. Um eine korrekte Sortierung zu erzwingen, wird mit `setlocale` aus dem `POSIX`-Modul auf deutsche Sortierung geschaltet[39].

Wenn der Browser Javascript unterstützt, wird der Submit schon beim Betätigen der Checkbox durchgeführt, ansonsten gibt es einen zusätzlichen Submit-Button.

Die komplette Liste der Straßennamen wird in der Funktion `choose_all_form` erzeugt. Um die Datei klein zu halten, wurde darauf verzichtet, wie bei

[37] Das Format für diese Parameter lautet: „Straßenname“ Delimiter „Bezirk“ Delimiter „PLZ“.

[38] Format: „x,y“

[39] Da die Locale-Benennungen auf unterschiedlichen Betriebssystemen nicht genormt sind, werden mehrere Werte (`de`, `de_DE` usw.) versucht.

`choose_ch_form` Checkboxen zum direkten Auswählen darzustellen[40]

Mit `nahbereich` wird der Nahbereich für die angegebenen Straßen ausgegeben. Unter Nahbereich versteht man hier die Liste der nächstliegenden Kreuzungen, die maximal einen Kilometer von der ausgewählten Kreuzung entfernt sind. Die nächsten Kreuzungen werden über das `Kreuzungen`-Objekt und die dazugehörige `nearest`-Methode festgestellt. Nach der Eingabe einer Kreuzung wird zur Suche übergegangen.

3.4.3 Suche und Ausgabe des Ergebnisses

Die Funktion `search_coord` ruft die Suchfunktion auf und sorgt für die Darstellung des Suchergebnisses. Die Suchparameter stammen aus dem Querystring, und zwar aus den Parametern `startc`, `viac` und `zielc`, und bezeichnen die Koordinaten des Start-, Via- und Zielpunktes. Evtl. sind auch die Klartextnamen der Punkte vorhanden; diese sind dann in den Parametern `start`, `via` und `ziel` codiert. Weiterhin kann noch der Parameter `printmode` gesetzt sein.

Falls das `StrassenNetz`-Objekt noch nicht existiert (dies ist der Fall, wenn das CGI im Normalmodus und nicht als FCGI läuft), wird dieses Objekt erst einmal erzeugt. Falls eine gecachte Version existiert, wird diese verwendet (durch die Angabe der Option `UseCache => 1`).

Die Suche wird mit Aufruf der Methode `new_search` durchgeführt. Als Extra-Argument wird der Via-Punkt (falls vorhanden) übergeben. Die Funktion gibt als Ergebnis die kürzeste Route zurück.

Falls Wetterdaten angezeigt werden sollen und können bzw. eine Leistungsberechnung möglich ist, wird über den Aufruf der Funktion `gather_weather_proc` das Ergebnis einer früher gestarteten `start_weather_proc`-Funktion eingesammelt.

Bei der Leistungsberechnung wird die Route in einzelne Teilstrecken zerlegt. Für jede Teilstrecke wird der Wind (Richtung und Stärke) ermittelt und mittels der Funktion `bikepwr_get_v` die Leistung in Watt für diese Teilstrecke berechnet.

In der HTML-Ausgabe werden zunächst die Länge der Route und die Fahrzeiten für die Durchschnittsgeschwindigkeiten sowie für die Leistungsaufnahmen angezeigt.

Das Ergebnis der Suche wird als Fahrtenbuch in Tabellenform angezeigt. Der Inhalt dieser Tabelle wurde bereits auf Seite 35 beschrieben. Bei der Ausgabe gibt es noch einige Besonderheiten:

- Bei Text-Browsern (z.B. lynx) wird die Ausgabe nicht als Tabelle, sondern als vorformatierter Text vorgenommen. Wegen der gewöhnlichen Be-

[40] Der dynamisch generierte HTML-Code hätte sich um ca. 2x bis 3x vergrößert, da für jede Straße nochmal der Code für die Checkbox (mindestens Start und Ziel) dazugekommen wäre. Alternativ hätte man zunächst ein Javascript-Array und dann die Liste dynamisch im Client erzeugen können. Bei Tests hat sich das als *sehr* langsam erwiesen.

schränkung der Zeilenbreite auf 80 Zeichen muß der Straßenname ggfs. mit der Funktion `string_kuerzen` gekürzt werden.

- Bei der Ausgabe für die Druckvorlage wird auf die Einfärbung der Tabellenzellen verzichtet sowie eine etwas kleinere Schrift verwendet.

Zusätzlich zur Routenbeschreibung kann eine Grafik mit der Streckenführung erzeugt werden. Im nächsten Teil der HTML-Ausgabe werden Radiobuttons für die Einstellung der Parameter für diese Grafik angezeigt: für die Größe der erzeugten Grafik sowie für die Details. Mit Klick auf den Button „Grafik erzeugen“ wird die Grafik mit Hilfe der Methoden aus der Klasse `BBBikeDraw` erzeugt. Die Ausgabe der Grafik erfolgt in einem separaten Fenster.

Im FCGI-Modus wird bei der Erzeugung der Grafik auf den normalen CGI-Modus zurückgefallen. Dafür gibt es zwei Gründe:

1. Das Skript wird für andere Zugriffe blockiert, wenn gerade die Bilderzeugung läuft. Dies gilt prinzipiell für alle Berechnungen, allerdings ist die Erzeugung der Grafik zweitaufwendiger als beispielsweise die Durchführung der Suche, die im Schnitt nur ein bis zwei Sekunden dauert.

2. Es gibt einen Bug entweder in `mod_fcgi` oder in `FCGI.pm`, so daß 0-Bytes aus dem Stream entfernt werden. Binäre Dateien und insbesondere GIF- und PNG-Bilder können somit nicht ausgegeben werden.

Im weiteren Velauf der Ausgabe werden weitere Buttons und Links für folgende Dienste bereitgestellt:

Druckausgabe: Diese Ausgabe ist gegenüber dem normalen Formular leicht optimiert, weil keine Farben dargestellt werden. In der Druckausgabe wird nur das Fahrtenbuch ausgegeben, insbesondere fehlen alle (für das Drucken überflüssigen) Buttons und Links.

Rückweg: Wegen Einbahnstraßen kann der Rückweg anders verlaufen als der Hinweg. Weiterhin gestalten sich Abbiegevorgänge in verschiedenen Richtungen auch unterschiedlich. Zuletzt ist das Ergebnis der Leistungsberechnung bei Rücken- und Gegenwind unterschiedlich.

Neue Anfrage mit gleichem Start- oder Zielpunkt: Bei Anwählen dieser Links wird entweder einer der beiden Punkte beibehalten.

Falls die vorhin erwähnte Funktion `gather_weather_proc` aufgerufen wurde, werden die aktuellen Wetterdaten (Temperatur, Windrichtung und -stärke) ausgegeben.

Weitere Funktionen

`draw_route` Das Zeichnen der Route wird in dieser Funktion vorgenommen. Hier wird zunächst versucht, das Modul `BBBikeDraw.pm` zu laden, dann wird ein neues `BBBikeDraw`-Objekt erstellt und die zugrundeliegende Karte und die Route gezeichnet. Zum Schluß wird `flush` ausgeführt und damit das erzeugte Bild an den Client geschickt.

`draw_map` Hiermit wird ein Stadtplanquadrant erzeugt. Als Eingabeparameter werden x- und y-Wert für den zu zeichnenden Quadranten übergeben. Falls ein Cache für die Kartenbilder existiert und die gewünschte Karte ebenfalls existiert, kann das Erzeugen übersprungen werden, es sei denn, die Option `-force` wurde angegeben oder es wurden Änderungen an der Datenbasis (der Zeitstempel der Straßen-Datenbasis ist jünger als der Zeitstempel der Karte) bzw. Änderungen am CGI-Skript selber durchgeführt. Wenn die Karte nicht existiert oder nicht aktuell ist, wird wie bei `draw_route` eine neue Karte erzeugt und im Cache abgespeichert. Zugleich wird eine korrespondierende Map-Datei erstellt, die eine HTML-Imagemap der erstellten Karte enthält. Beim Führen der Maus über eine Straße wird damit der Name der Straße in der Statuszeile angezeigt. Leider funktioniert dieses Feature nur beim Netscape-Browser unter Unix. Dieses Feature ist für zukünftige Entwicklungen vorbereitet, wenn Browser das HTML-4.0-Tag `TITLE` unterstützen werden.

Nach dem Erzeugen der Karte wird der HTML-Code ausgegeben. Die Seite ist tabellarisch aufgebaut, wobei in der Mitte die Karte und in den Himmelsrichtungen Buttons zum Bewegen des Kartenausschnitts erscheinen. Beim Klick auf die Karte werden die Koordinaten ermittelt und an das CGI-Skript zurückgegeben.

`limit_processes` Beim `CGI::MiniSvr`-Interface (siehe Beschreibung auf Seite 63) wird für jeden Benutzer ein neuer Prozeß gestartet. Da das CGI-Programm sehr viel virtuellen Speicher verbraucht (ca. 10 MB unter FreeBSD/i386), muß die Anzahl der erlaubten Prozesse eingeschränkt werden. In der Konfigurationsvariablen `$max_proc` wird diese Anzahl eingestellt. In der DBM-Datei[41] `bbbike-limit` im temporären Verzeichnis wird eine Liste mit den Prozeß-IDs der laufenden `bbbike.cgi`-Prozesse gehalten. Mittels der Perl-Funktion `kill` wird überprüft, ob der Prozeß noch existiert. Falls mehr `bbbike.cgi`-Prozesse als erlaubt existieren, wird auf die normale, langsame Version umgeschaltet. Ansonsten wird der aktuelle „Prozeß-Slot“ zurückgegeben, in dem später die richtige Prozeßnummer eingetragen wird. Das Aufrufen von `limit_processes` erfolgt nämlich im Serverprozeß. Erst nach dem `fork` des Kindprozesses ist dessen Prozeßnummer bekannt und kann dann mit der Funktion `set_process` eingetragen werden.

[41] DBM-Dateien sind leichte Datenbanksysteme, die bereits auf Seite 12 beschrieben wurden.

`start_weather_proc` Diese Funktion startet einen Prozeß, der die Wetterdaten mittels `wettermeldung2` aus dem WWW holt und in die Datei `wettermeldung` im temporären Verzeichnis schreibt. Dies geschieht nur, falls die Datei nicht älter als eine halbe Stunde ist. Beim Starten des Prozesses wird zwischen Windows und Unix als Serversystem unterschieden. Bei Unix wird konventionell `fork` und `exec` für das Erzeugen des neuen Prozesses verwendet. Bei Windows ist `fork` nicht innerhalb von Perl implementiert; dafür wird bei Vorhandensein des Moduls `Win32::Process` die Funktion `Create` verwendet. Das Aufrufen von `start_weather_proc` geschieht so früh wie möglich, damit bei einer möglichen langsamen Verbindung das Holen der Wetterdaten und das Berechnen der Route parallel ablaufen kann.

`gather_weather_proc` Das Ergebnis von `start_weather_proc` wird hier „eingesammelt". Auch hier wird überprüft, ob die Wetterdaten aus der Datei `wettermeldung` jünger als als 30 Minuten sind. Das Parsen geht einfach über die `split`-Funktion von Perl.

3.5 Konsolenanwendung

Die Konsolenanwendung `cbbbike` ist als eine einfache Schnittstelle für Benutzer gedacht, deren System keine grafische Benutzeroberfäche hat, Perl/Tk nicht unterstützt oder kein Zugang zum WWW besitzt. `cbbbike` ist wie das CGI-Interface so aufgebaut, daß der Benutzer schrittweise nach dem genauen Start- und Zielpunkt gefragt wird. Als weitere Vereinfachungen ist die Eingabe einer Via-Straße weggefallen, außerdem wird auf das Suchen von Straßennamen im STRADA-Verzeichnis verzichtet.

Folgende Schritte werden im Programm für die Konfiguration durchgeführt:

- Laden der Perl-Module wie z.B. `Strassen.pm`
- Setzen einer Standardkonfiguration
- Laden einer Konfiguration, falls eine Konfigurationsdatei mit dem Namen `cbbbike.config` existiert
- Setzen von betriebssystemabhängigen Einstellungen. Wenn das Programm auf der Konsole von Windows/MS-DOS/OS2-Systemen läuft, muß der Zeichensatz angepaßt werden. Die Datenbasis von `BBBike` verwendet den Zeichensatz `ISO-8859-1`, wohingegen MS-DOS-basierte Systeme `CP 437`, `CP 850` oder ähnliche Codepages benutzen ([Sim92]). Für diesen Fall wird die gesamte Ein- und Ausgabe über die Funktionen `cp850_iso` und `iso_cp850` abgewickelt.
- Versuch, die XS-Datei `BBBikeXS` zu laden.

- Ermitteln von Kommandozeilenoptionen. In der Kommandozeile kann `-v` für Debuggingmeldungen und `-bench` für Zeitmessungen eingegeben werden. Weiterhin kann bereits in der Kommandozeile eine Start- und Zielstraße eingegeben werden.

- Berechnung der intensionalen Datenbasis aus der Straßen-Datenbasis. Es muß die Liste aller Kreuzungen sowie das Straßennetz berechnet werden, es sei denn, ein Straßennetz-Server ist aktiv. Während der Berechnung läuft im Hintergrund ein Waitproc-Prozeß (gestartet mit `waitproc` und beendet mit `stop_waitproc`), der einen rotierenden Strich während der Berechnungszeit anzeigt.

- Einstellen des Terminals. Für die Eingabe wird das Modul `Term::ReadLine` mit Kommandozeilenhistory verwendet. Weiterhin wird der Zeichencode für das Löschen des Bildschirms festgestellt. Falls die installierte `Term::ReadLine`-Variante History unterstützt, wird die alte History-Datei mit `readhist` geladen.

Nach der Konfiguration wird die Schleife für die Eingabe der Start- und Zielstraße und die Berechnung der Strecke aufgerufen. Die Straßennamen befinden sich in den Variablen `$point[0]` und `$point[1]`, falls sie in der Kommandozeile eingegeben wurden, ansonsten werden diese Variablen mittels der `readline`-Methode eingelesen. Beim Einlesen wird sogleich die Konvertierung mit der `input`-Funktion vorgenommen.

Im nächsten Schritt wird überprüft, ob die Straßennamen gültig sind. Mit der `agrep`-Methode in `Strassen.pm` wird eine Liste der besten Treffer zurückgegeben. Wenn mehr als ein Treffer gefunden wurde, wird die Liste dem Benutzer zur Auswahl der Straße vorgelegt. Die „verifizierte" Straße wird in die Variable `$point_verified[n]` übernommen.

Danach wird die genaue Kreuzung abgefragt. Die gültigen Kreuzungen sind im Hash `%crossings` abgespeichert. Mit den beiden Kreuzungen für Start- und Zielort sind auch die Koordinaten für die Suche definiert (in den Variablen `$start_coord` und `$ziel_coord`). Diese Koordinaten werden als Eingabeparameter für die `StrassenNetz`-Methode `new_search` verwendet.

Das Suchergebnis, falls eins existiert, wird als Liste mit Straßennamen, Etappenlänge, Richtung und Winkel seitenweise ausgegeben. Am Ende hat man die Option, eine Grafik mittels `BBBikeDraw` zu erzeugen und die erzeugte GIF- oder PNG-Datei entweder in einem Tk-Fenster oder in einem externen Viewer[42] anzuzeigen. Weiterhin kann die Liste nochmals angezeigt werden, der Rückweg berechnet oder eine neue Anfrage gestartet werden.

Weitere Funktionen in `cbbbike`:

[42] z.B. `xv` unter X11 oder `Paint Shop Pro` unter Windows

exit_app: Beendet das Programm. Vor dem Beenden wird noch writehist aufgerufen, um die History zu speichern.

next_line: Gibt eine Zeile aus und überprüft, ob bereits mehr als 23 Zeilen dargestellt wurden. In diesem Fall wird die Ausgabe seitenweise dargestellt.

show_graphics: Diese Funktion ruft die Methoden zum Erzeugen eines GIF- oder PNG-Bildes der Route im Modul BBBikeDraw auf. Voreingestellt werden neben der Route auch die Straßen, U- und S-Bahnlinien und Gewässer mit eingezeichnet. Falls ein GIF- oder PNG-Viewer definiert ist, wird dieser aufgerufen.

writehist: Falls die Readline-Variante History unterstützt, wird das Array der History in die Datei $configdir/cbbbikehist geschrieben. Standardmäßig werden höchstens 20 Einträge in der Datei gehalten.

readhist,_readhist: Hier wird ebenfalls überprüft, ob History unterstützt wird. Bei einem positiven Ergebnis wird die History-Datei ausgelesen und per SetHistory gesetzt.

tk_viewer: Der Tk-Viewer kann anstelle des externen GIF- oder PNG-Viewers zum Anzeigen der Routengrafik verwendet werden.

3.6 Gemeinsam genutzte Module

3.6.1 Strassen.pm

Das Modul Strassen.pm stellt die Verbindung der Anwendungen an die Datenbasis her. Weiterhin sind in diesem Modul auf der Datenbasis arbeitende Algorithmen enthalten, z.B. die Routensuche. Das Modul enthält mehrere Perl-Packages: Strassen::Util (für gemeinsam genutzte Hilfsfunktionen und das Caching), Strassen (für das Lesen und Schreiben der Datenbasis), Strasse (Klasse zum Kapseln einer Straßendefinition), Kreuzungen (Klasse zum Kapseln von Kreuzungen), MultiStrassen (von Strassen abgeleitete Klasse zum Zusammenfassen von Strassen-Objekten) und StrassenNetz (Berechnung des Straßennetz-Graphen und Suche).

Strassen

Strassen enthält Methoden zum Auslesen der Daten im Verzeichnis data. Wenn dem Konstruktor new ein Dateiname übergeben wird, wird read_data zum Einlesen aufgerufen. Mit write werden die Daten wieder auf den Datenträger zurückgeschrieben. Mit den Methoden init und next können die Datensätze innerhalb einer Schleife ausgelesen werden. next gibt der Datensatz als Perl-Array zurück, während `next_obj` ein `Strasse`-Objekt zurückgibt. Mit der Methode `all_crossings` können alle Kreuzungen ausgegeben werden.

Weiterhin existieren die Methoden `agrep` und `agrep_file` zum approximativen Suchen nach Straßennamen. Diese Methoden untersuchen auf Existenz des Perl-Moduls `String::Approx` und des Programms `agrep`. Wenn keines dieser beiden Werkzeuge vorhanden ist, wird auf normale Suche umgestellt. Die approximative Suche erfolgt inkrementell: zuerst wird eine genaue Suche ausgeführt, und dann die Suche mit einem bis drei Fehlern. Die gefundenen Treffer werden als Liste zurückgegeben.

Mit `diff_orig` kann die Differenz zwischen zwei `Strassen`-Dateien festgestellt werden[43]. Die Differenz kann für das Neuzeichnen eines Straßennetzes verwendet werden, wobei nur die geänderten Teile gezeichnet werden, was die Geschwindigkeit erhöht.

Mit `make_grid`, `grid` und `get_new_grids` kann ein Straßennetz in ein Gitter eingeteilt werden. Dabei werden die Straßen in einer Relation zu Gittern der Länge 1000 m zugeordnet. Diese Zuordnung kann beim Zeichnen des Straßennetzes verwendet werden, um nur die Straßen im sichtbaren Kartenbereich zu zeichnen. Durch die Verwendung eines Gitters kann die Anzahl der zu untersuchenden Straßen begrenzt werden. Die Implementierung des Gittermodells wurde dadurch vereinfacht, daß nur festgestellt wird, ob sich wenigstens ein Punkt des Straßenzuges innerhalb eines Quadranten befindet. Streckenteile, die sich innerhalb des Quadranten befinden, die beiden zugehörigen Endpunkte jedoch außerhalb, werden somit nicht gezeichnet.

`Strasse`

In der Klasse `Strasse` ist außer der Kapselung eines `Strassen`-Datensatzes auch die statische Funktion `de_artikel` definiert, um die Präposition und den Artikel korrekt zu ermitteln: „**in den** Mehringdamm“, „**in die** Franklinstr.“ usw.

`MultiStrassen`

Mit der Klasse `MultiStrassen` können mehrere `Strassen`-Objekte zu einem Objekt zusammengefügt werden. Eine Schleife mit `init` und `next` geht damit über alle enthaltenen `Strassen`-Objekte.

`Kreuzungen`

`Kreuzungen` wird in erster Linie dazu benutzt, um anhand eines Punktes die nächstgelegene Kreuzung zu finden. Auch hier wird ein Gitter über Daten gelegt. Mit der Methode `nearest` werden die nächsten Kreuzungen zurückgegeben.

---

[43] Die Differenz wird mit dem externen Programm `diff` festgestellt.

StrassenNetz

Die Methode make_net in der Klasse StrassenNetz erstellt aus einem Strassen-Objekt einen Graphen, der für die Suche verwendet werden kann. Es gibt drei Versionen dieser Methode: make_net_slow_1, make_net_slow_2 und make_net_fast. Letztere Methode ist in der Datei BBBikeXS.xs enthalten und in C geschrieben, so daß Systeme mit einem C-Compiler diese schnellere Version verwenden können. Die beiden langsameren Methoden unterscheiden sich in der Art des Datenformats, das für die Speicherung des Netzes verwendet wird: die Version 1 benutzt gewöhnliche multidimensionale Perl-Hashes, während die Version 2 binäre Arrays verwendet. Der Geschwindigkeitsunterschied zwischen beiden Versionen ist eher marginal, was anscheinend einer effizienten Implementierung von assoziativen Arrays in Perl zuzuschreiben ist. Es ist möglich, die erzeugten Daten auf einen Datenträger zu schreiben, so daß bei einem erneuten Aufruf nur die Cache-Version geladen und die Berechnung nicht ausgeführt werden muß. Eine weitere Möglichkeit des Caching ist die Verwendung der Methoden load_net_mldbm und save_net_mldbm. Bei diesen Methoden wird der Graph in eine MLDBM-Datenbank geschrieben. Der Zugriff auf die Daten erfolgt direkt von der Datenbank; ein gesondertes Einlesen ist deshalb nicht notwendig. Der Vorteil hierbei ist, daß die Zeit zum Berechnen des Straßennetzes entfällt und daß der Speicherverbrauch niedriger ist. Nachteilig ist die geringere Geschwindigkeit als bei der Verwendung von Perl-Hashes (ca. 3x langsamer). Weiterhin nachteilig ist, daß mit add_net nachträglich hinzugefügte Punkte persistent bleiben. Als letzte Möglichkeit der Speicherung des Graphen wurde die Verwendung von IPC::Shareable ausgetestet. Dabei muß ein weiteres Server-Programm gestartet werden, bei dem das Straßennetz berechnet wird und dann per Shared Memory anderen Programmen zur Verfügung gestellt. Diese Vorgehensweise würde man bei der CGI-Anwendung wählen, wenn viele Zugriffe zu erwarten sind, da der Speicher für den Graphen nur einmal verbraucht wird.

Der erzeugte Graph kann mit der make_sperre-Methode so modifiziert werden, daß gesperrte Straßen und die gesperrte Richtung bei Einbahnstraßen aus dem Graphen gelöscht werden. add_net fügt hingegen weitere Punkte zum Graphen hinzu (siehe Abschnitt über das Setzen von Start- und Zielpunkten auf Seite 50). Wenn Fährstrecken bei der Suche berücksichtigt werden sollen und somit im Straßennetz-Graphen vorkommen müssen, muß zusätzlich add_faehre aufgerufen werden. Das Löschen von nachträglich hinzugefügten Punkten wird in del_add_net vorgenommen. reachable gibt an, ob ein Knoten im Graphen erreicht werden kann. Es liegt in der Verantwortung des Datenpflegers zuzusichern, daß der Graph zusammenhängend ist; in diesem Fall gibt es zwischen zwei erreichbaren Knoten immer eine Route. In der realen Welt ist die Forderung nach einem zusammenhängenden Graphen angemessen, wenn keine Inseln betrachtet werden. In diesem Fall könnte man während des Erzeugen des Graphes für einen Knoten eine Traversierung starten und sich die traversierten Knoten in einem

Hash merken. Nach der Traversierung würde man die restlichen Knoten durchgehen und könnte leicht feststellen, welche Knoten sich im selben Netz befinden. Alle anderen Knoten würden sich in einem anderen Netz befinden. Auch hier muß man eine Traversierung starten usw.

In der CGI-Anwendung existiert die Möglichkeit, URLs[44] zu erstellen, die einen festen Start- oder Zielpunkt haben. Es kann passieren, daß die Koordinaten für den Start- oder Zielpunkt wegen einer Korrektur der Datenbasis ungültig werden. Um das zu vermeiden, können mit der Methode `fix_coords` die nächstgelegenen Koordinaten festgestellt werden. Diese Methode funktioniert nur, wenn die Änderung im kleinen Rahmen bleibt — was in der Regel der Fall ist.

Normalerweise enthält der erzeugte Graph die Entfernung zwischen Knoten als Kantenwert. Mit `make_net_cat` wird stattdessen die Kategorie des Streckenstücks, mit `make_net_steigung` die Steigung zwischen den beiden Knoten zugewiesen. Bei der Berechnung des Steigungs-Netzes kann eine minimale Steigung übergeben werden, für die eine Kante im Netz erzeugt werden soll. Diese minimale Steigung beträgt in der Voreinstellung 1%.

Die Funktionen für die Suche sind in Abschnitt 3.6.2 (Seite 79) genauer beschrieben.

Eine erzeugte Route besteht nur aus Koordinaten. Um die entsprechenden Straßennamen zu finden, kann die Methode `net2name` aufgerufen werden. Die Methode `route_to_name` verwendet `net2name`, um zusätzlich zum Straßennamen auch die Entfernung eines Streckenstücks und den Winkel für den nächsten Abbiegevorgang zu berechnen. Die Berechnung des Winkels erfolgt über eine geometrische Betrachtung der Route und kann deshalb in der Realität ungenau sein. Eine bessere Datengrundlage kann dieses Problem lösen. Bei Landstraßen besteht der Name meist aus einer Zeichenkette mit dem Format „A - B“, wobei dies bedeutet, daß die Strecke von A nach B führt. In `route_to_name` wird festgestellt, ob diese Zeichenkette zu „B - A“ umgedreht werden muß, falls die Route in die entgegengesetzte Richtung führt. In der Methode `nearest_street` wird versucht, für Streckenstücke, die nicht im Straßennetz-Graphen vorkommen, eine Alternative in der Nähe zu finden. Solche Ungenauigkeiten können durch Anwendung von `add_net`, dem Abspeichern einer Route und späteren Laden vorkommen, oder auch durch Korrekturen in der Datenbasis.

`Strassen::Util`

In `Strassen::Util` befinden sich für alle anderen Klassen benutzbare Cache-Funktionen. `try_cache` überprüft, ob eine Cache-Datei bereits existiert. Mit `valid_cache` wird anhand von Zeitstempeln überprüft, ob die Cache-Datei mit Hinblick auf die Quelldateien aktuell und somit gültig ist. Die Funktion `get_from_cache` lädt eine Datenstruktur aus dem Cache und `write_cache`

[44]URL: Uniform Resource Locator, bezeichnet die eindeutige Adresse eines Dokuments innerhalb des WWWs

schreibt sie in den Cache. Je nachdem, welche Module im System installiert sind, wird für die persistente Speicherung der Perl-Datenstrukturen `Storable` (bevorzugt, weil am schnellsten), `Data::Dumper`[45] oder `VirtArray` (nur für eine eingeschränkte Art von Arrays) verwendet.

In `Strassen::Util` befinden sich auch Utility-Funktionen für das Berechnen einer Entfernung (`strecke`) und die Richtung für das Abbiegen (`abbiegen`, `get_direction` und `best_from_direction`).

3.6.2 Suche

Arten der Suchalgorithmen

Bei der Suche zwischen einem Start- und Zielpunkt in einem Straßennetz handelt es sich um eine Suche von einem Startknoten in einem gerichteten Graphen zu einem Zielknoten. Bei den Suchalgorithmen wird zwischen uninformierten und informierten Suchalgorithmen unterschieden.

Uninformierte Suchalgorithmen

Bei uninformierten Suchalgorithmen, auch „Brute-force search“ genannten, wird kein weiteres Wissen über die Position des Zielknotens verwendet; die Suche muß also „blind“ nach dem Ziel suchen. Bei diesen Suchalgorithmen gibt es folgende Vertreter:

Breitensuche (Breadth-first search): Bei der Breitensuche werden in jedem Iterationsschritt alle Nachfolgeknoten expandiert und dies so lange wiederholt, bis der Zielknoten gefunden wurde. Bei diesem Verfahren wird immer der kürzeste Weg gefunden, allerdings unter hohen Kosten, was Suchzeit und Speicherverbrauch betrifft. Der Aufwand für beides beträgt $O(b^d)$, wobei b der „node branching factor“, also die durchschnittliche Anzahl von Nachfolgeknoten, und d die Tiefe ist.

Tiefensuche (Depth-first search): Bei der Tiefensuche wird in jedem Iterationsschritt nur ein neuer Knoten expandiert und dies solange wiederholt, bis der Zielknoten gefunden wurde oder keine Nachfolgeknoten mehr existieren. Im letzteren Fall wird per Backtracking zum davor expandierten Knoten zurückgegangen und ein anderer Knoten expandiert. Bei unendlich großen Graphen oder wenn keine Liste der bereits untersuchten Knoten gehalten wird und der Graph Zyklen enthält, kann die Termination des Algorithmus' nicht zugesichert werden. Der zeitliche Suchaufwand beträgt $O(e^d)$, wobei e der „edge branching factor“ ist, also die durchschnittliche

[45] `Data::Dumper` ist seit der Perl-Version 5.005 in der Standard-Distribution enthalten. Dieses Modul ist allerdings langsamer als Storable, weil im Gegensatz dazu lesbarer Perl-Code und keine Binärdateien erzeugt werden.

Anzahl von Nachfolgekanten. Der Aufwand für den Speicherplatz hingegen ist nur linear.

Depth-first iterative-deepening: Eine Variante der Tiefensuche, die nicht die Nachteile der gewöhnlichen Tiefensuche hat, ist „depth-first iterative-deepening". Dabei wird eine Tiefensuche zunächst zu allen Knoten durchgeführt, die einen Knoten entfernt sind, dann eine Tiefensuche zu allen Knoten, die zwei Knoten entfernt sind usw. Dadurch, daß noch immer eine Tiefensuche durchgeführt wird, liegt der Speicheraufwand noch immer bei $O(d)$. Laut [Sti85] liegt auch der zeitliche Aufwand bei $O(b^d)$.

Bidirektionale Suche: Diese Suchart kann angewendet werden, wenn genau ein Ziel vorhanden ist. Dabei wird simultan sowohl vom Startknoten als auch vom Zielknoten aus gesucht, bis sich die beiden Untersuchungsräume schneiden und somit ein Pfad gefunden wurde.

Heuristische Suchalgorithmen

Bei den informierten Suchalgorithmen können Heuristiken angewandt werden, so daß die Zahl der untersuchten Knoten verringert und somit die Suchzeit verkürzt werden kann.

A*

Der Algorithmus A* hält während der Laufzeit eine Liste aller offenen Knoten (Knoten, die bereits erreicht, aber noch nicht expandiert wurden). Von diesen Knoten wird immer derjenige mit dem kleinsten Wert f ausgesucht, um weitere Knoten zu expandieren. f berechnet sich aus den bisher berechneten Kosten für den Weg vom Startknoten bis zum aktuellen Knoten und einer heuristischen Funktion, die die Kosten vom aktuellen Knoten bis zum Zielknoten abschätzt. Wenn ein Zielknoten gefunden wurde, wird der Algorithmus beendet und der gefundene Weg ausgegeben.

In [Pea85] wird gezeigt, daß dieser Suchalgorithmus optimal ist, wenn die verwendete heuristische Funktion die tatsächlichen Kosten niemals überschreitet. Optimal heißt in diesem Fall, daß immer der kürzeste Weg gefunden wird und daß die Zahl der untersuchten Knoten minimal ist. Als Problem des Algorithmus' wird bei [Kor88] der hohe Speicherverbrauch beschrieben. Als Alternative zu A* wird dort vorgeschlagen, den Algorithmus Iterative-Deepening-A* (IDA*) zu verwenden, der ähnlich optimal wie A* ist und nicht dessen Speicherverbrauch besitzt.

In den folgenden Abbildungen sich die verschiedenen Suchformen, die in BBBike durchgeführt werden können, als Menge der expandierten Knoten visualisiert. Bei der Tiefensuche ist hier eine einfache Variante ohne Backtracking verwendet worden, die teilweise informiert ist, d.h. der nächste expandierte Knoten ist immer der dem Ziel am nächsten gelegene. Die Breitensuche und die Suche mit A* entspricht den oben beschriebenen Algorithmen.

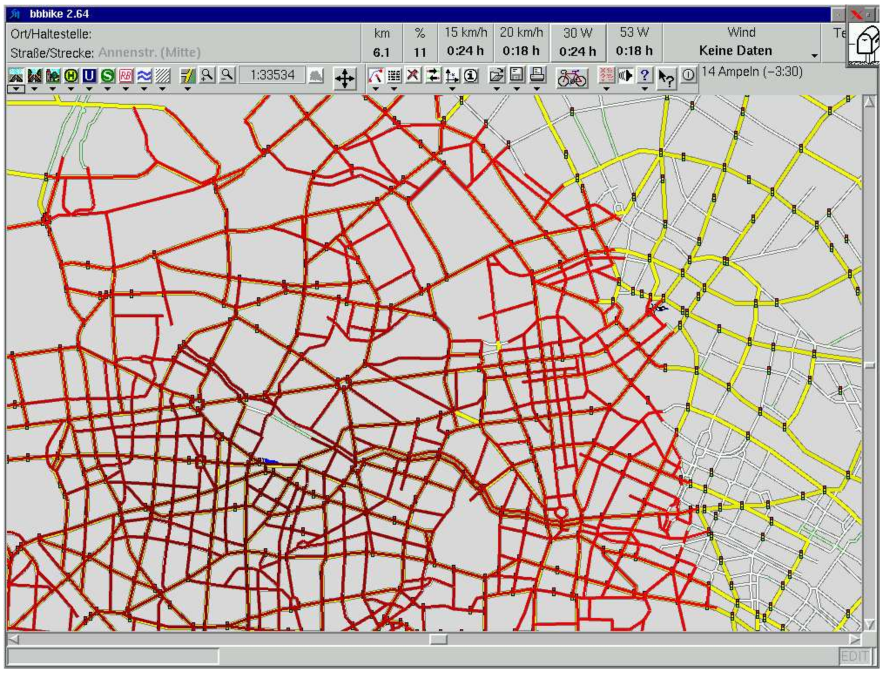

Abbildung 3.12: Visualisierung der expandierten Knoten bei einer Breitensuche

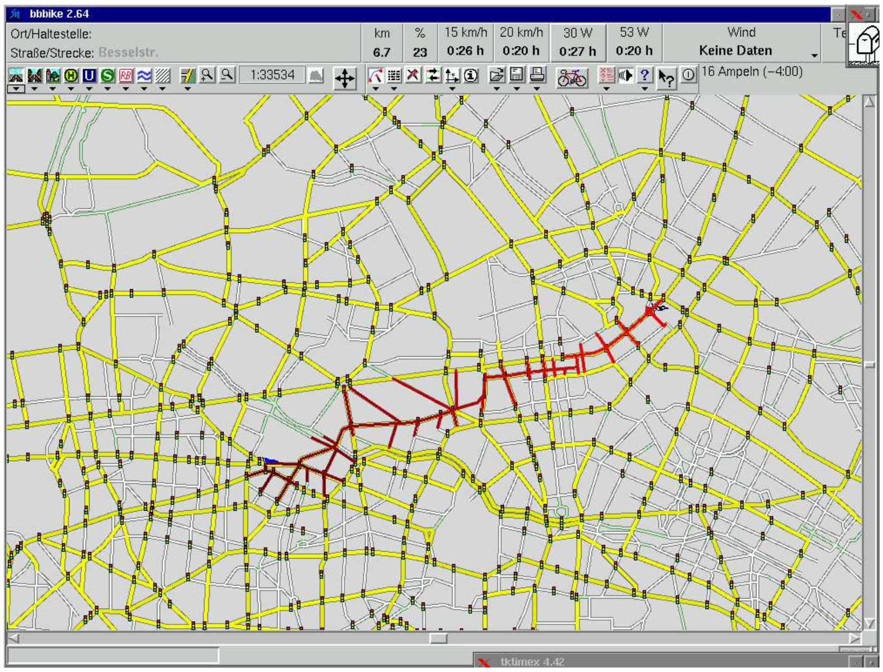

Abbildung 3.13: Visualisierung der expandierten Knoten bei einer Tiefensuchvariante

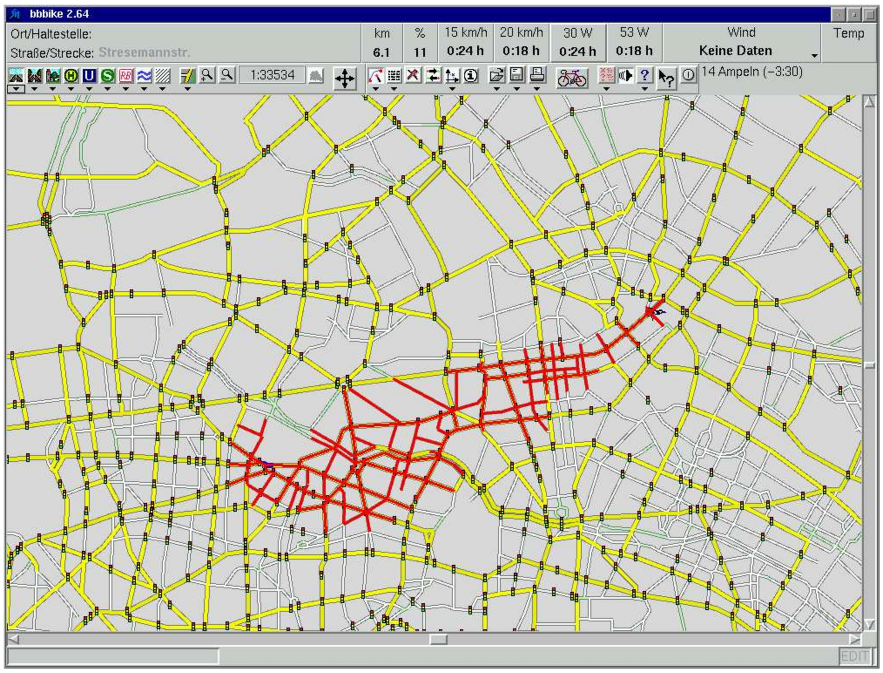

Abbildung 3.14: Visualisierung der expandierten Knoten bei A*

Besondere Betrachtungen für BBBike

Bei der Suche in einem Straßennetz kann die Luftlinienentfernung als heuristische Funktion verwendet werden. Somit bietet sich A* als Suchalgorithmus an. Auf die Verwendung von IDA* wurde verzichtet, weil der Speicherverbrauch kein vordringliches Problem ist. Die Zahl der Knoten ist endlich und relativ klein (abhängig von der räumlichen Ausdehnung des Straßennetzes ca. 15000 bis 25000 Knoten), so daß sich die maximale Größe für die Liste der geöffneten Knoten auch in diesem Bereich befindet.

Im einfachen Fall, also wenn keine weiteren Parameter für die Routenoptimierung gegeben sind, wird als heuristische Funktion die Entfernung der geradlinigen Verbindung zwischen dem untersuchten Knoten und dem Endknoten angenommen. Diese Entfernung ist auch die minimal mögliche Entfernung, deshalb findet der Algorithmus immer die optimale (kürzeste) Route (die Forderung lautet, daß das Ergebnis der heuristischen Funktion immer kleiner oder gleich der Länge des optimalen Pfades sein muß).

Wenn Parameter für die Routenoptimierung gegeben sind, muß die Funktion zur Berechnung der Kosten modifiziert werden. Bei einer Parametrisierung wird nicht mehr ausschließlich mit Streckenlängen gearbeitet, sondern mit einer Mischung aus Fahrzeiten und Streckenentfernungen. Für die heuristische Funktion wird weiterhin die Luftlinienentfernung verwendet. Für die Berechnung der Kosten einer Route wird bei nicht parametrisierten Strecken weiterhin die Entfernung verwendet. Bei parametrisierten Strecken wird die Entfernung mit dem Quotienten aus Wunschgeschwindigkeit und möglicher Geschwindigkeit modifiziert. Beispiel: als Wunschgeschwindigkeit wurde 25 km/h angegeben, ein Streckenabschnitt erlaubt aber (z.B. wegen schlechter Fahrbahnbeschaffenheit) nur 18 km/h als Geschwindigkeit. Die Streckenlänge für dieses Straßenstück wird für die Suche um 25/18 erhöht.

Dieses Verfahren ist nicht perfekt, weil unter ungünstigen Umständen die heuristische Funktion nicht immer minimal bleibt. Dies kann bei Gefällen auftreten, wo sich bei der Modifizierung der Streckenlänge eine kürzere Länge ergibt. Für Berlin wurde wegen der Seltenheit und Kürze von Steigungen auf eine gesonderte Behandlung verzichtet. Für hügelige Orte kann sich dieser Umstand auf die Optimalität der Route auswirken; dort müßte man gesonderte Mechanismen anwenden, z.B. eine Analyse der Höhenpunkte von Start- und Zielknoten und eine entsprechende Modifikation der heuristischen Funktion.

Eine Problematik bei der Parametrisierung besteht darin, daß nicht mehr ausschließlich objektive Werte (die Entfernung) bei der Suche verwendet werden, sondern zusätzlich Werte, die subjektiv begründet sind. Einerseits gilt das für die Attribute des Straßennetzes, für die es oft keine objektiven Grundlagen gibt und die von einem Experten abgeschätzt werden müssen, anderseits für die Einschätzung der gefahrenen Geschwindigkeiten vom Benutzer („Wie schnell fahre ich auf Kopfsteinpflaster? Wie schnell fahre ich normalerweise?"). Weiterhin wird bei der Parametrisierung auch mit statistischen Werten gearbeitet, die nur eine Annäherung an die reale Welt darstellen[46].

Implementierung

Die Suche ist in der Datei `Strassen.pm` implementiert. Dabei wird die Fähigkeit von Perl verwendet, dynamisch Perl-Code zu erzeugen und während der Laufzeit in die interne Struktur zu kompilieren und ablaufen zu lassen. Dadurch, daß nicht alle Parameter bei der Suche angegeben werden müssen, kann ein Performancegewinn erzielt werden, weil nicht benötigte Code-Stücke nicht durchlaufen werden müssen und einige Bedingungen bereits in die Erzeugung des Codes verlagert werden können. Zusätzlich können Optimierungsvariablen als Konstanten in den Perl-Code geschrieben werden; auch dadurch verbessert sich die Performance, da auf Perl-Konstanten schneller zugegriffen werden kann als auf Variablen. Durch die dynamische Erzeugung des Codes können auch verschiedene Algorithmen für die Suche erzeugt werden.

Die Methode für die Suche ist `new_search` im Modul `StrassenNetz`. Als Argumente werden die Start- und Zielkoordinate eingegeben. Weitere Argumente werden in Hash-Schreibweise (-Option `=>` Wert) übergeben. Die Vorgehensweise in dieser Methode ist wie folgt:

[46] Dies ist der Fall bei Ampelschaltungen, wo aus der durchschnittlichen Rot- und Grünphase der Zeitverlust und damit die in dieser Zeit verlorene Strecke berechnet wird.

- Abfragen und Definieren der Optimierungsparameter sowie statistischer und ähnlicher Optionen
- Erzeugen des Suchcodes als Zeichenkette mit der Methode `build_search_code`
- Erzeugen einer anonymen Funktion aus dem Suchcode mit der `eval`-Funktion von Perl
- Ausführen der anonymen Funktion und Aufbereiten des Ergebnisses
- Ausgabe des Ergebnisses

Zunächst wird überprüft, ob Optimierungsparameter vorhanden sind. Optimierungsparameter existieren für Ampeln (Routen mit vielen Ampeln vermeiden), Straßenqualität (schlechten Straßenbelag vermeiden), Straßenkategorie (Nebenstraßen gegenüber Hauptstraßen bevorzugen), Radwege (je nach Eingabe der Optimierungsparameter Bevorzugung oder Vermeidung von Radwegen) und Steigungen (Vermeiden von Steigungen). Die Optimierungsparameter werden in lexikalische Variablen geschrieben, die in der anonymen Suchfunktion sichtbar sind. Es wäre auch möglich gewesen, direkt auf den `%args`-Hash zuzugreifen, dies wäre aber etwas langsamer als der Zugriff auf die entsprechenden Skalare.

Weiterhin werden am Anfang lexikalische Variablen für die Statistik deklariert und weitere Argumente für die visuelle Suche und die Kontrolle des Rückgabewertes ausgewertet.

Mit den Argumenten wird dann über `build_search_code` der Perl-Code für die anonyme Suchfunktion erzeugt. `build_search_code` wird weiter unten genauer beschrieben. Der erzeugte Code kann im Debug-Modus (wenn die Variable `$VERBOSE` gesetzt ist) ausgegeben werden. Dies hat den Vorteil, daß bei Fehlermeldungen im später evaluierten Code über die Zeilennummer die Position des Fehlers festgestellt werden kann.

Der erzeugte Code wird evaluiert und die erzeugte Suchfunktion in die Variable `$search_sub` geschrieben. Wenn ohne Via-Punkte gesucht werden soll, wird die Suchfunktion nur einmal aufgerufen und das Suchergebnis in `@res` gespeichert. Mit Via-Punkten wird die Suchfunktion in einer Schleife ausgeführt und die Route schrittweise zusammengebaut.

Falls während der Suche statistischen Daten gesammelt wurden, werden diese am Ende der Suchfunktion ausgegeben. Die Statistik wird nur bei Bedarf vorgenommen, weil das Mitführen und Berechnen der statistischen Daten während der Suche einen weiteren Performanceverlust bedeutet.

Zum Schluß wird das Resultat ausgegeben. Wenn der Parameter `AsObj` gesetzt ist, wird das Ergebnis als Liste von Route-Objekten zurückgegeben, ansonsten als Liste von Pfadlisten.

Erzeugen der Suchfunktion In `build_search_code` wird eine Perl-Funktion als Zeichenkette erzeugt und zurückgegeben. Für den Aufbau der Suchfunktion gibt es zwei Typen:

- einfache Suche über die Länge ohne weitere Routenoptimierungen
- erweiterte Suche mit der Verwendung von „Penalties“[47] anstelle der Länge für Routenoptimierungen

Bei der erweiterten Suche wird die Länge durch die Optimierungsparameter modifiziert, wie auf Seite 78 beschrieben.

Der Ablauf in `build_search_code` sieht wie folgt aus: Zunächst wird festgestellt, ob mit einfacher oder erweiterter Suche gearbeitet werden soll. Entsprechend wird die Variable `$has_penalty` gesetzt und der zusätzliche Code in der Variable `$penalty_code` erzeugt. Je nachdem, welcher Algorithmus gewählt wurde (A*, Tiefen- oder Breitensuche) wird der entsprechende Code erzeugt und in der Variablen `$code` abgelegt. Zusätzlich gibt es noch die Möglichkeit, zwischen den beiden Datenstrukturformen des Straßennetz-Objekts zu wählen, also zwischen dem Perl-Hash und der komprimierten binären Form (Variable `$use_2`). Für die Statistik und das Feature „Visual Search“ wird je nach Wunsch ebenfalls Code erzeugt.

Schließlich wird das Ergebnis als Liste mit zwei Elementen zurückgegeben: das erste Element enthält die gefundene Pfadliste und das zweite Element enthält die Länge des Pfades.

3.6.3 `BikePower`

Für die Berechnung der aufgewendeten Leistung beim Fahrradfahren bzw. umgekehrt für die Berechnung der möglichen Geschwindigkeiten bei konstanten Leistungen existieren Formeln, in die Größen wie Steigung, Windgeschwindigkeit und -richtung, Masse von Fahrer und Rad und andere eingehen. Zwar ist diese Berechnung mittels einfacher mechanischer Formeln eine grobe Vereinfachung der Realität, da chaotische Störungen wie kurzzeitige Änderungen des Windes und meßtechnische Ungenauigkeiten nicht erfaßt werden können. Eine weitere Quelle der Ungenauigkeit ist eine lückenhafte Erfassung von grundlegenden Daten wie z.B. beim Höhenprofil. Außerdem spielen besonders im Stadtverkehr andere Faktoren eine Rolle bei der Fahrzeitberechnung, z.B. die Dichte des Verkehrs (auch Stau betrifft Radfahrer!) oder Ampelschaltungen. Trotzdem sollte der Vergleich zwischen der reinen Fahrtzeit berechnet aus der Durchschnittsgeschwindigkeit und der Fahrzeit, die sich aus der Leistungsabgabe berechnet, dem Radfahrer ein Gefühl geben, wo er nun mehr „strampeln“ muß, um zum Ziel zu kommen.

[47]„Penalty“ bezeichnet hier Parameter, mit dem ein Streckenstück gewichtet wird.

Es existiert bereits ein C-Programm von Ken Roberts namens `bike_pwr`, das die mechanischen Berechnungen durchführt. Die Formeln in diesem Programm ([Bpw99], [Str99]) sind hauptsächlich aus [Whi82] übernommen worden.

Um das Programm in akzeptabler Geschwindigkeit aus BBBike verwenden zu können[48], war eine Portierung nach Perl, und, als zweiter Schritt, eine Portierung nach Perl/XS notwendig. Die Portierung hatte ein abgeschlossenes Perl5-Modul zum Ergebnis: BikePower. Dieses Modul wurde bereits im Perl-Netzwerk CPAN (Comprehensive Perl Archive Network, [CPA99]), veröffentlicht[49].

Das Modul besitzt zwei Modi:

1. Eingabe einer Geschwindigkeit und Berechnung der Leistung

2. Eingabe einer Leistung und Berechnung der Geschwindigkeit

Die Kalorienaufnahme kann zusätzlich berechnet werden; die Berechnung erfolgt direkt aus der Leistung.

Folgende Eingabewerte sind möglich:

Temperatur: Die Temperatur wirkt sich leicht auf die Dichte der Luft und somit auf den Luftwiderstand aus. Die Unterschiede sind vernachlässigbar (ca. 5 W Unterschied bei 20 km/h zwischen 0°C und 30°C); die Berechnung war im Originalprogramm bereits vorhanden und wurde für den Port übernommen. Da die Wetterdaten für die Windberechnung bereits aus dem WWW geholt werden, ist es kein weiteres Problem, auch die Temperatur dabei mit einzulesen. Die Temperatur wird in °C angegeben.

Windgeschwindigkeit: Die Windgeschwindigkeit wirkt sich maßgeblich auf die Leistung aus. Dabei kann angegeben werden, ob es sich um Rückenwind (negative Werte) oder Wind von vorne (positive Werte) handelt. Bei Winden von der Seite kann eine Variable `cross_wind` gesetzt werden. Damit wird eine Windgeschwindigkeit von ca. 60 % des Wertes angenommen. Die Windgeschwindigkeit wird in m/s angegeben.

Luftwiderstandsbeiwert: Der Luftwiderstand ist um so größer, je größer die Vorderfläche des Objekts ist. Als weiterer Faktor kommt die Objektform hinzu, die für zusätzliche Verwirbelungen sorgen kann, die entsprechend hemmend wirken. Eine genaue Messung des Luftwiderstandsbeiwerts kann nur im Windkanal erfolgen. Da dies sehr aufwendig ist, ist im Modul eine Liste mit Standardwerten für den c_w-Wert enthalten . Der c_w-Wert hängt von der Haltung des Fahrers (stehend bis gebückt) ab. Bei Fahrten im Verbund sinkt der Luftwiderstand erheblich.

[48] Mit Perl ist es möglich, externe Programme mit dem „Backticks"-Operator zu starten und die Ausgabe einzulesen. Allerdings ist diese Methode ineffizient, weil für jeden Aufruf ein Prozeß gestartet werden muß.

[49] Die genaue URL lautet `<URL:http://www.cpan.org/modules/by-module/BikePower>`

Masse des Fahrers: Die Masse wirkt sich auf den Rollwiderstand und besonders auf Fahrten mit Steigungen aus. Da mit der Körpermasse auch die Kalorienaufnahme berechnet wird, muß sie von der sonstigen Masse (Rad, Kleidung und Gepäck) getrennt eingegeben werden. Die Massen werden in kg angegeben.

Masse des Rades: Hier wird die Masse eingegeben, die sich nicht auf die Kalorienberechnung bezieht, also die Masse des Rades, der Kleidung und sonstiges.

Grundumsatz (basal metabolism): Der Grundumsatz ist die Leistung, die ein Mensch im Ruhezustand verbraucht. Diese Leistung wird bei der Kalorienberechnung berücksichtigt.

Mechanischer Wirkungsgrad (human inefficiency): Die tatsächliche geleistete mechanische Arbeit ist nur ein Bruchteil der Gesamtleistung eines Menschen (siehe [Lei97]).

Rollwiderstandsbeiwert: Je nach Beschaffenheit des Reifens (Unterschiede im Profil, Breite des Reifens und Druck) variiert der Rollwiderstand. Im Modul sind Beispielwerte für verschiedene Reifentypen enthalten.

Effizienz der Übertragung: Fahrräder mit Kettenschaltungen besitzen eine Übertragungseffizienz von 95%. Die Effizienz bei Fahrrädern mit Nabenschaltungen beträgt je nach eingelegten Gang 80% bis 95% ([Piv90]). Verdreckte und rostige Ketten besitzen ebenfalls eine geringere Effizienz. Die Einheit für diese Eingabe ist %.

Steigung: Mit positiven Werten werden Steigungen gekennzeichnet, mit negativen Werten Gefälle. Die Einheit ist m/m, also die gleiche Einheit, die auf Straßenschildern (hier in Prozent-Angabe) verwendet werden.

Folgende Werte können, je nach Modus, als Ein- oder Ausgabewerte behandelt werden:

Geschwindigkeit: Wenn die Geschwindigkeit (in m/s) angegeben ist, kann anhand der anderen Werte die benötigte Leistungsaufnahme (in W) berechnet werden.

Leistung: Wenn die Leistung angegeben ist, wird die Geschwindigkeit dazu ausgegeben. Für die Berechnung wird eine Intervallsuche nach Newton verwendet; deshalb ist dieser Modus langsamer als der Geschwindigkeitsmodus.

Verbrauch: Hiermit kann der Kalorienverbrauch ausgegeben werden.

Das Modul ist objekt-orientiert geschrieben. Es gibt einen Konstruktur `new`, der die Standardwerte aus der persönlichen Datei des Benutzers (in Unix aus der Datei `~/.bikepowerrc`), oder, falls nicht vorhanden, aus Standardwerten, die im Modul definiert wurden, setzt.

Um die Variablen zu setzen, können dynamisch erzeugte Methoden verwendet werden. Z.B. muß zum Setzen der Geschwindigkeit folgende Methode verwendet werden:

```
$bp->velocity(20/3.6);
```

während zum Auslesen die gleiche Methode ohne Funktionswert verwendet wird. Um anzugeben, welche Variable gegeben sein soll, kann die Methode `given` mit den Argumenten `v` für Geschwindigkeit oder `P` für Leistung aufgerufen werden. Mit der Methode `calc` wird schließlich die Berechnung durchgeführt. Danach kann der berechnete Wert, also z.B. die Leistung, wieder ausgelesen werden:

```
$power = $bp->power;
```

Um neben der reinen Perl-Version des Moduls noch bessere Performance zu erreichen, wurde eine zusätzliche XS-Version geschrieben, die die `calc`-Methode um eine schnellere C-Version ersetzt. Die `calc`-Methode ist die Methode, die in Schleifen permanent aufgerufen wird und deshalb als erster Kandidat für die Performance-Verbesserung gewählt wurde.

Die Entsprechung der `calc`-Methode heißt in der XS-Datei `calcXS`. In der XS-Funktion müssen zuerst die Variablen, die im Perl-Objekt als Hashwerte enthalten sind, ausgelesen und normalen C-Variablen zugewiesen werden. Nach der Berechnung müssen die berechneten Werte für die Einzelleistungen wieder in das Perl-Objekt geschrieben werden (mittels der Perl-Funktion `hv_store`).

Die Entscheidung, ob `calc` oder `calcXS` verwendet wird, erfolgt in der Datei `BikePower.pm`. Je nachdem, ob der „bootstrap“ (das Einbinden der dynamischen Library, die aus der XS-Datei nach dem Übersetzen hervorgegangen ist) erfolgreich war, wird als Alias für die `calc`-Methode `calcXS` oder `calc_slow` (die langsamere Perl-Variante) zugewiesen. Zusätzlich wird die Variable `$has_xs` gesetzt.

Zusätzlich zum Kern-Modul existieren zwei weitere Module, `BikePower::Tk` (siehe Abbildung 3.15) und `BikePower::HTML`. Ersteres ist eine grafische Oberfläche für das interaktive Eingeben von Werten und kann mit dem Perl-Skript `tkbikepwr` aufgerufen werden. `BikePower::HTML` ist ein einfaches HTML-Formular, das für die Eingabe und zum Abspeichern der eingegeben Daten in einen HTTP-Cookie dient. Hierbei erfolgt keine weitere Berechnung. Dieses Modul wird in erster Linie von der WWW-Version von BBBike verwendet.

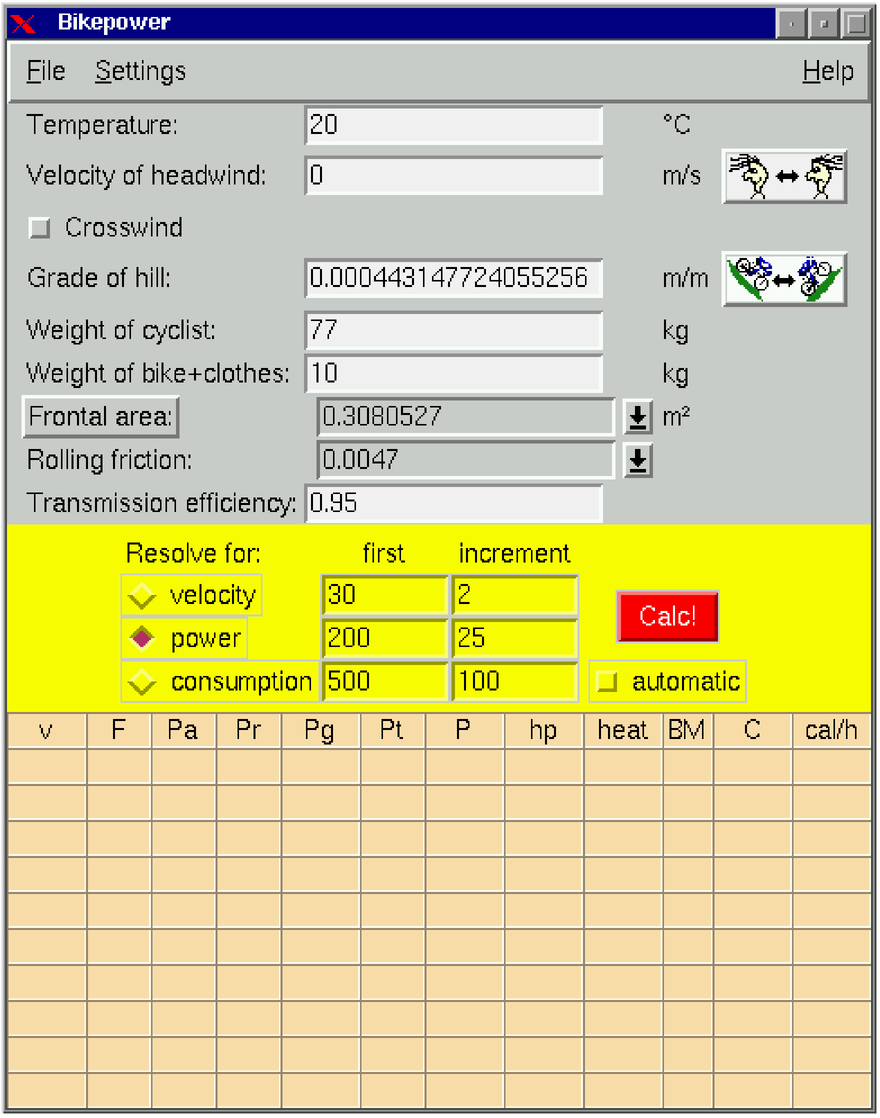

Abbildung 3.15: Tk-Interface des BikePower-Moduls

3.6.4 `wettermeldung2`

`wettermeldung2` ist ein hybrides Perl-Skript/Modul[50], mit dem Wetterinformationen aus dem WWW ausgewertet werden können. Zur Zeit können folgende URLs behandelt werden:

- `<URL:http://www.met.fu-berlin.de/deutsch/Wetter/beobachtung.html>`
- `<URL:http://www.met.fu-berlin.de/deutsch/Wetter/meldungen.html>`
- `<URL:http://www.hundert6.de/b_wett.htm>`

[50] Je nach Aufrufart kann diese Datei als Skript oder Modul verwendet werden.

Die Information der ersten beiden URLs stammt vom Meteorologischen Institut der FU Berlin mit dem Standort Berlin-Dahlem. Die Datei `beobachtung.html` wird halbstündlich aktualisiert, enthält aber nur die maximale Windstärke, während `meldungen.html` nur alle drei Stunden aktualisiert wird, dafür aber die mittlere Windgeschwindigkeit enthält, die für die Benutzung in BBBike nützlicher ist.

Die Wetterinformationen der dritten URL werden vom Deutschen Wetterdienst (DWD) geliefert. Der Standort ist Berlin-Tempelhof, die Informationen werden stündlich aktualisiert und es ist die mittlere Windstärke angegeben.

Neben der Benutzung aus BBBike heraus kann das Skript auch als eigenständiges Programm benutzt werden. Ein Anwendungsbeispiel wäre das schnelle Holen der Wetterinformationen, ohne einen WWW-Browser verwenden zu müssen. Ein anderes Beispiel wäre der Aufbau einer Wetterdatenbank, indem das Skript periodisch aus einem Cron-Job heraus aufgerufen wird. Im Fachbereichsnetz Informatik der TU Berlin wird dies praktiziert; die Wetterdatenbanken (aufgeteilt nach Jahren und Quellen der Information) können im Verzeichnis `/home/e/eserte/doc/met` gefunden werden.

Um höchste Kompatibilität zu wahren, kann das Skript sowohl mit Perl4 als auch mit Perl5 verwendet werden. Getestete Plattformen sind Solaris 2.6, FreeBSD 2.2.8/3.3 und Windows NT 4.0.

Aufruf In der Kommandozeile muß beim Aufruf des Skriptes genau eine der Optionen `-dahlem1` (für `beobachtung.html` der FU Berlin), `-dahlem2` (für `meldungen.html` der FU) oder `-tempelhof` (für die Informationen des DWD) angegeben werden.

Bei dieser Aufrufart wird versucht, eine Verbindung ins Internet aufzubauen. Um unnötig wartende Prozesse zu verhindern, wird zunächst in der Funktion `site_reachable` überprüft, ob der angesprochene Server auf `ping` reagiert, d.h. ob der Server läuft und über das Netz erreicht werden kann. Falls das Skript mit Perl5 läuft, wird das Standardmodul `Net::Ping` verwendet, ansonsten wird das Programm `ping`[51] aufgerufen.

Falls `ping` erfolgreich war, wird die Seite vom Server geladen. Das geschieht mit der Funktion `fetch_cmd`. Auch hier werden mehrere Alternativen probiert. Mit Perl5 und einem installierten libwww-Paket wird das Modul `LWP::UserAgent` zum Holen der Seite verwendet. Die Seite wird in eine temporäre Datei geschrieben. Falls Perl5 und/oder libwww nicht verfügbar ist, wird untersucht, ob das Programm `socket`[52] vorhanden ist. Falls `socket` ebenfalls nicht vorhanden ist,

[51] Für `ping` muß berücksichtigt werden, daß die Kommandozeilenoptionen von Betriebssystem zu Betriebssystem unterschiedlich sind. Im Skript stehen die Optionen in den Feldern `@pingopt` und `@pingpost`, die beim Programmstart abhängig vom Betriebssystem initialisiert werden.

[52] `socket` ist ein Programm von Jürgen Nickelsen und unter `<URL:http://www.freebsd.org/~wosch/src/socket-1.1.tar.gz>` zu finden.

wird das Modul `Http.pm`, das mit BBBike mitgeliefert wird, verwendet (nur für Perl5).

Der Rückgabewert ist entweder der Dateiname der temporären Datei oder ein Pipe-Kommando für den `socket`-Aufruf.

Abhängig von der gewählten URL wird eine der Parser-Routinen `parse_dahlem1`, `parse_dahlem2` oder `parse_tempelhof` aufgerufen.

Die Parser-Routinen sind für das Holen der Datei aus dem Internet (mittels der Routine `fetch_cmd`) verantwortlich. Danach wird, wie bei Perl üblich, eine `open - while(<...>) - close`-Schleife durchgeführt, bei der mittels regulären Ausdrücken versucht wird, die erforderlichen Daten wie Datum, Uhrzeit, Temperatur, Windgeschwindigkeit etc. zu erkennen. Durch die Verwendung von regulären Ausdrücken (es wird sonst kein weiteres Wissen über die Struktur der Datei benötigt) ist das Parsen relativ fehlertolerant. Da aber Strukturänderungen an den Dateien, die der Parser doch nicht mehr erkennt, nicht ausgeschlossen werden können, wird als Minimallösung am Ende des Parsens überprüft, ob alle Daten ausgelesen werden konnten. Außerdem wird während des Parsens überprüft, ob Daten doppelt ausgelesen werden, sowie eine einfache Plausibiltätsprüfung vorgenommen. So gibt es für die Windrichtungen nur acht verschiedene Möglichkeiten. Beim Luftdruck wird nachgeprüft, ob der Wert größer als 970 hPa ist. Beim Einsatz des Parsers wurden einige oft auftretende Probleme erkannt und mit zusätzlichen Abfragen behoben[53]. Bei Problemen wird eine entsprechende Meldung (ggfs. als E-Mail) ausgegeben.

Beim erfolgreichen Parsen wird eine Zeile mit den Daten ausgegeben. Das Format lautet:

```
   Datum (dd.mm.yyyy) | Uhrzeit (hh.mm) | Temperatur
(°C) | Luftdruck (hPa) | Windrichtung (N, E, W ...) |
Spitzenwindstärke/-geschwindigkeit (in Beaufort oder m/s) |
Luftfeuchtigkeit (%) | mittlere Windstärke/Geschwindigkeit
(in Beaufort oder m/s) | Niederschlag (in mm seit 0 Uhr) |
Wetterzustand (Fließtext)
```

3.6.5 BBBikeXS

Zu den nachträglich geladenen Modulen gehört `BBBikeXS`. Die Dateien dazu befinden sich im Verzeichnis `ext/BBBikeXS`. Hierbei handelt sich um eine XS-Erweiterung. XS-Erweiterungen sind in C oder einer anderen kompilierten Sprache geschriebene Bibliotheken für die Verwendung in Perl-Programmen. XS-Dateien müssen kompiliert werden.

Da nicht überall die Existenz von Compilern vorausgesetzt werden kann (insbesondere bei Windows!), befinden sich in dieser Datei nur Funktionen, für die

[53] So kommt es bei der Wettermeldung aus Dahlem häufiger vor, daß offensichtlich falsche Werte für Niederschlag, Luftfeuchtigkeit und Windgeschwindigkeit eingegeben wurden.

es eine Entsprechung in Perl gibt, d.h. die XS-Erweiterung ist komplett optional. Die Verwendung der XS-Erweiterung gibt einen Geschwindigkeitsvorteil bei einigen Funktionen um den Faktor 2–3 (siehe Tabelle weiter unten). Die Kompilation wird im Installationsprogramm gestartet, wenn sie überhaupt möglich ist.

Folgende Funktionen werden in BBBikeXS bereitgestellt (Datei `BBBikeXS.xs`):

- `Strassen::to_koord1_XS` für das Umwandeln eines Strings im „x,y“-Format in ein zwei-elementiges Feld.
- `Strassen::to_koord_XS` für das Umwandeln einer Liste von „x,y“-Strings in ein Feld im Format [[x1,y1],[x2,y2],...].
- `StrassenNetz::make_net` für das Erstellen des internen Straßennetzes.
- `main::set_canvas_scale_XS`: Beim Ändern der Vergrößerung muß die Skalierungsvariable mit dieser Funktion gesetzt werden, damit `transpose_ls` korrekt funktioniert.
- `main::transpose_ls` für die Umberechnung der Koordinaten in das aktuelle Canvas-Format.
- `BBBike::fast_plot_str`: Diese Funktion implementiert das schnelle Zeichnen von Strecken. Diese Funktion ist nicht so mächtig wie die entsprechende Perl-Funktion `plotstr` und kann nicht unter allen Umständen verwendet werden (z.B. im Editiermodus oder beim Zeichnen im Portrait-Mode).
- `BBBike::fast_plot_point`: Diese Funktion implementiert das schnelle Zeichnen von Punkten. Diese Funktion beschränkt sich auf das Zeichnen von Ampeln und ist somit wesentlich weniger mächtig als die entsprechende Perl-Funktion `plotp`.

Weil in dieser Datei auch Konstanten verwendet werden, die auch im Perl-Teil gesetzt werden, und diese Konstanten gleich sein müssen, kann beim Kompilieren mit der gesetzten Präprozessor-Variablen `MYDEBUG` überprüft werden, ob die Konstanten in beiden Teilen übereinstimmen. Diese Überprüfung befindet sich im BOOT-Teil des XS-Moduls, das jedesmal beim dynamischen Laden des Moduls ausgeführt wird.

In der folgenden Tabelle sind die Ergebnisse eines Benchmarks beschrieben, der auf einem PC mit AMD K6-III-Prozessor, 400 MHz mit FreeBSD 3.3, perl5.005_03 und Tk800.015 ausgeführt wurde (Datei `diplom/code/bench_bbbikexs.pl`):

Funktion	Loops pro CPU-Sekunde	
	Perl	XS
transpose_ls	49876	78817
to_koord	4412	13371
make_net	0,69	1,29
plotstr	0,63	1,76

3.6.6 Kurze Übersicht über weitere Module

Hier folgt eine kurze Übersicht über weitere Module, die für das Informationssystem geschrieben wurden.

`Ampelschaltung.pm`: Modul zum Verwalten von Ampelschaltungsphasen. Zur Zeit wird in der Anwendung nur die durchschnittlich verlorene Zeit an Ampeln berechnet. In Zukunft soll die verlorene Zeit für jede einzelne Ampel abhängig von der Tageszeit zurückgegeben werden können.

`BBBikeAdvanced.pm`: Hier sind Funktionen enthalten, die für Entwickler gedacht sind und die nicht für normale Benutzer zugänglich sind. Das sind in erster Linie Funktionen, die das Editieren der Datenbasis und das Debuggen des Programms erleichtern.

`BBBikeCalc.pm`: Berechnungen (z.B. Windberechnungen), die in mehreren Anwendungen verwendet werden, sind hier enthalten.

`BBBikeDraw`: Hiermit werden mittels des Moduls `GD` GIF- oder PNG-Grafiken der Straßenkarte für die Anzeige in `bbbike.cgi` erzeugt. Die Straßenkarten sind entweder statisch erzeugte Karten für die Eingabe des Start- und Zielpunktes oder dynamisch erzeugte, in denen eine Route eingezeichnet ist. Bei den dynamisch erzeugten Karten ist der Maßstab an die Route angepaßt.

`BBBikeEdit.pm`: Dieses Modul enthält weitere Funktionalität für das Editieren der Datenbasis, und zwar speziell zum Editieren von Radwegen und Ampelschaltungen.

`BBBikeExp.pm`: Hier ist das experimentelle dynamische Zeichnen des Straßennetzes implementiert.

`BBBikeMail.pm`: Enthält Funktionen für das Versenden von Text als E-Mail und Fax.

`BBBikePrint.pm`: Das Drucken ist auf den verschiedenen Plattformen unter Perl/Tk nicht standardisiert. Dieses Modul implementiert das gemeinsame Interface für das Erzeugen und Drucken von Postscript, sowohl von einem Tk-Canvas aus als auch für normalen Text.

`BBBikeProfil.pm`: Ausgabe des Streckenprofils für die angegebene Route.

`BBBikeTrans.pm`: Erzeugen der Umwandlungsfunktionen zwischen dem internen Koordinatensystem und den Canvas-Koordinaten.

`BBBikeUtil.pm`: Dieses Modul enthält eine Reihe weiterer Utility-Funktionen, die sich nirgendwo sonst einordnen lassen (Dateifunktionen, einfache mathematische/trigonometrische Funktionen und Zeichensatz-Konvertierungsfunktionen).

`GfxConvert.pm`: Hier sind Konvertierungsfunktionen für einige Bildformate (PPM, GIF, JPEG, Postscript) definiert. Diese Funktionen werden beim Speichern der Karte verwendet. Da hier auf externe Programme zurückgegriffen wird, die möglicherweise nicht überall installiert sind, funktionieren nicht alle Konvertierungsrichtungen.

`Karte.pm`, `Karte/*.pm`: Diese Module enthalten Konvertierungsfunktionen für verschiedene Koordinatensysteme.

`PLZ.pm`: Dieses Modul stellt Methoden zum Zugriff auf die STRADA-Datei mit den Berliner Strassen und den dazugehörigen Postleitzahlen. Unter anderem verwendet die Methode `look` approximative Suchmethoden für das Auffinden eines Straßennamens.

`Radwege.pm`: Hier sind Kategorien für Radwege aufgelistet.

`Route.pm`: Eine Klasse für das Kapseln von Routenbeschreibungen.

`Telefonbuch.pm`, `Telefonbuch98.pm`, `Telefonbuch99.pm`: Diese Module erzeugen einen Dialog für den Zugriff auf die Straßen-Datenbank der Telefonbuch-CD-ROMs der Deutschen Telekom.

`Update.pm`: Mit diesem Modul kann ein Update der Datenbasis über das Internet erfolgen.

`Wizards.pm`: Dieses experimentelle Modul stellt eine alternative Konfigurationsmethode mit Hilfe von „Wizards“ zur Verfügung.

`lib/BrowserInfo.pm`: Hiermit werden Browser-spezifische Parameter für die CGI-Anwendung gesetzt.

`lib/EasySound.pm`: Dieses Modul stellt ein Interface für verschiedene Audio-Module her. Bei der Überprüfung der installierten Module und Programme werden folgende Möglichkeiten getestet: das Vorhandensein der Unix-Devices `/dev/audio`, `/dev/pcaudio` oder `/dev/speaker`; die Existenz der Perl-Module `Audio::Play` und `Win32::Sound` sowie das externe „Network Audio System“.

`lib/Http.pm`: Ein sehr einfaches Modul für die Kommunikation mit einem WWW-Server. Dieses Modul wird nur verwendet, wenn das libwww-Paket nicht installiert ist.

`lib/MyFile.pm`: Die Funktionen in diesem Modul öffnen aus einer Liste von Dateien die erste mögliche.

`lib/TkChange.pm`: Hier sind einige Standard-Methoden aus Tk durch eigene Methoden überschrieben.

`lib/TkCompat.pm`: Kompatibilitätsmodul für ältere Tk-Versionen.

`lib/UnixUtil.pm`: Dieses Modul enthält Unix-spezifische Hilfsfunktionen.

`lib/WWWBrowser.pm`: Mit der Funktion `start_browser` wird eine plattformunabhängige Methode zum Starten eines WWW-Browsers mit der angegeben URL angeboten.

`lib/Win32Util.pm`: Das Modul stellt Windows-spezifische Funktionen zur Verfügung: Funktionen zum Auslesen der Registry, das Ausgeben der Zuordung Dateityp – ausführendes Programm oder um das Heimatverzeichnis des aktuellen Benutzers zurückzugeben.

`lib/WinCompat.pm`: Kompatibilitätsmodul für die Tk-Version unter Windows.

3.7 Hilfsprogramme

3.7.1 Konsistenzsicherung der Datenbasis

Um die Konsistenz der Datenbasis zu sichern, existieren einige Hilfsprogramme:

`check2`: Dieses Programm überprüft für die übergebenen Dateien, ob die Knoten der einen Datei eine Untermenge der Vereinigung der anderen Dateien bilden. Z.B. müssen alle Ampeln in Straßenknoten vorhanden sein.

`check_str_plz`: Damit verhindert wird, daß Straßennamen falsch geschrieben werden, wird hier überprüft, ob sich alle Straßennamen auch in der STRADA-Datei befinden. Außerdem wird bei diesem Check erzwungen, daß unterschiedliche Straßen mit gleichen Namen durch Angabe der Bezirke näher beschrieben werden. Ausnahmen können in den Dateien `add_plaetze` und `add_str` aufgeführt werden.

`check_cont`: Hier wird überprüft, ob in der Datenbank unterbrochene Straßen einen einzigen Streckenzug darstellen. Ausnahmen können in der Datei `str_cont_ausnahme` aufgeführt werden.

`check_double`: Dieses Programm überprüft, ob Ortsbezeichnungen doppelt vorkommen. Da im Auswahldialog Orte mit gleichen Namen nicht unterschieden werden können, müssen solche Fälle korrigiert werden. Gewöhnlich wird der kleinere Ort mit einer Zusatzbezeichnung versehen (z.B. „bei Bernau“ oder „Spreewald“).

`check_neighbour`: Hiermit wird überprüft, ob benachbarte Knoten in einer Attributdatei auch benachbarte Knoten in Straßennetzdatei sind. Fehler können auftreten, wenn nur in einer der beiden Dateien Knoten entfernt oder hinzugefügt wurden.

`check_nearest`: Nah beieinanderliegende Knoten stellen eine potentielle Fehlerquelle dar, weil sie möglicherweise eine Kreuzung darstellen sollten. Eine Liste von solchen Knotenpaaren wird mit diesem Programm ausgegeben; der Benutzer muß dann manuell feststellen, ob es sich tatsächlich um einen Fehler handelt.

`insert_points, change_points`: Wenn ein Knoten eingefügt oder gelöscht werden muß, sollten diese beiden Programme verwendet werden, damit alle abhängigen Attributdateien entsprechend geändert werden.

3.7.2 Erstellung von Distributionen

Es werden zwei Formen für die Erstellung von Distributionen unterstützt:

FreeBSD-Port: Das System der FreeBSD-Ports ([FBp99]) wird für die Erstellung von Softwaredistributionen für das Unix-Betriebssystem FreeBSD ([FBh99]) verwendet. Es besteht aus einem Makefile, das das Holen des Quellcodes aus dem Internet, das Kompilieren und Installieren übernimmt, sowie aus Patches, einigen Beschreibungsdateien und der Liste der installierten Dateien. Mit dem Makefile kann man entweder den Quellcode selbst kompilieren oder ein binäres Paket erzeugen und weiterverbreiten. Die Port-Beschreibungsdateien befinden sich im Unterverzeichnis `port/freebsd`.

Linux-RPM-Paket: RPM besteht aus einer Spezifikationsdatei, die die Informationen zum Softwarepaket enthält, sowie Build-Anweisungen und eine Liste der installierten Dateien enthält. Die Spezifikationsdatei kann verwendet werden, um ein binäres Paket (die sogenannte rpm-Datei) zu erstellen. Die Spezifikationsdatei befindet sich im Unterverzeichnis `port/rpm`.

Um die Liste der installierten Dateien nicht dreifach halten zu müssen[54], werden diese dynamisch erzeugt. Die zugehörigen Programme heißen `mkport.pl` und `mkrpm.pl`. Beide verwenden für gemeinsame Definitionen das Modul `MetaPort.pm`.

[54] Für die Erstellung einer normalen Perl-Distribution existiert noch die Datei `MANIFEST`.

3.7.3 Testprogramme

Um Probleme beim CGI-Programm leichter finden zu können, wurde ein Testprogramm entwickelt, das mit den Daten aus dem WWW-Logfile arbeitet. Mit diesem Programm können Benutzungsabläufe von Benutzern nachgespielt und damit evtl. Probleme erkannt werden. Bei den Problemen kann es sich sowohl um Bugs im Programmcode als auch um Fehler in der Datenbasis handelt. Weiterhin können Benutzungsmuster erkannt werden, die vom Programm nicht abgefangen werden. Bei der Analyse mit dem Testprogramm wurden folgende Probleme erkannt:

- Benutzer geben häufig Platznamen oder die Bezeichnungen von wichtigen Gebäuden (z.B. Flughäfen) ein. Aufgrund dieser Tatsache wurden häufig vorkommende Plätze in die Datenbasis aufgenommen.

- Häufig wurde von den Benutzern „straße“ ausgeschrieben statt die abkürzende Form „str.“ zu benutzen. Um die Trefferwahrscheinlichkeit bei der approximativen Suche nach den Straßennamen zu verbessern, werden nun beide Schreibweisen (auch die Schreibweise mit „ss“ statt „ß“) äquivalent behandelt.

Das Testprogramm `cgi2.t` befindet sich im Verzeichnis `t`. Zunächst startet das Programm den Netscape-Browser. Dann wird die `AccessLog`-Datei auf Zugriffe auf `bbbike.cgi` untersucht. Die gefundenen Zugriffe werden per Option `-remote` nacheinander an den Browser geschickt.

3.8 Optimierung

3.8.1 Auslagerung von zeitkritischen Abläufen nach C

Bereits auf Seite 87 beschrieben ist das Modul `BBBikeXS`, welches zeitkritische Funktionen wie das Berechnen des Straßennetzes und das Zeichnen von Straßen und Punkten schneller durchführt.

Weiterhin existiert das Modul `VirtArray.xs`, das eine virtuelle Array-Implementierung auf der Basis von `mmap` bietet. Diese Array-Implementierung ist zwar langsamer als gewöhnliche Perl-Arrays[55]. Allerdings wird durch die Verwendung von `mmap` der Hauptspeicher geschont. Das Array steht sofort zur Verfügung und muß nicht wie bei `Storable` oder `Data::Dumper` zeitaufwendig in den Speicher geladen werden. `VirtArray` wird für das Caching der Kreuzungsdaten in `cbbbike` verwendet.

[55] Der Geschwindigkeitsnachteil rührt aus der Ineffizienz von Funktionsaufrufen in Perl her.

3.8.2 Tk-Optimierungen

createLine Die ersten Versionen von BBBike haben für das Zeichnen von Straßenzügen für jedes einzelne Straßenstück einen Aufruf von `createLine` verwendet. Das hat den Vorteil, daß für jedes Straßenstück ein eigener *Tag* (Canvas-Markierung) zugewiesen werden kann, was die Erkennung der aktuellen Straßenkoordinaten erleichtert. Die massive Verwendung von `createLine` führt jedoch zu Effizienzeinbußen. Als Lösung wird jetzt für jede Straße nur ein `createLine`-Aufruf durchgeführt, wodurch das Zeichnen schneller ist. Da für eine komplette Strecke nur ein *Tag* vergeben werden kann, ist das Auffinden der aktuellen Streckenkoordinaten schwieriger. Die Lösung, die in `nearest_line_points` implementiert ist, geht die Koordinaten der gesamten Strecke durch und überprüft, ob sich der angeklickte Punkt innerhalb eines Streckenstücks befindet.

Ein Benchmark (`miscsrc/createline_performance.pl`) unter AMD-K6-166MHz, Windows NT 4.0, perl5.00503 und Tk800.011, hat ergeben:

- Schnelle Lösung mit einem `createLine` für die gesamte Strecke: 6,9 Sekunden

- Langsame Lösung mit einem `createLine` für ein Streckenstück: 14,5 Sekunden

Von diesen Zeiten müssen ungefähr 3,8 Sekunden für die Schleife und den Aufruf der `transpose`-Funktion abgezogen werden.

Autoloading Ein Problem ist der hohe Speicherverbrauch des Programms. Bei Perl gibt es die Möglichkeit, Module als Autoload-Module zu kennzeichnen. Dabei werden alle Funktionen, die nach der Markierung `__END__` definiert werden, in eigene Dateien abgespeichert[56]. Diese Dateien werden bei Bedarf, also beim ersten Aufruf der jeweiligen Funktion, nachgeladen und stehen dann dem Programm zur Verfügung. Für das Erzeugen der einzelnen `.al`-Dateien existiert das Standard-Modul `AutoSplit`.

Für BBBike können diese standardisierten Möglichkeiten nicht direkt genutzt werden. `AutoSplit` funktioniert genaugenommen nur für Module, nicht aber für Skripte. Weiterhin gibt es Probleme, wenn sich mehrere Module (Packages) innerhalb einer Datei befinden, was z.B. bei `Strassen.pm` der Fall ist. Um diese Probleme zu umgehen, wurde das Programm `make_autoload.pl` geschrieben.

`make_autoload.pl` kopiert zunächst das gesamte BBBike-Verzeichnis in ein Arbeitsverzeichnis. Dann werden die Dateien `bbbike` und `Strassen.pm` so bearbeitet, daß sie für den Autosplitter verwendet werden können. Funktionen, die ausgelagert werden sollen, sind im Code mit `### AutoLoad Sub` gekennzeichnet und werden während der Bearbeitung an das Ende hinter die Markierung `__END__` verschoben. Für alle Packages wird der auskommentierte String `use AutoLoader`

[56] Diese Dateien befinden sich in den `auto`-Unterverzeichnissen und haben die Endung `.al`.

aktiviert. Jede Package aus `Strassen.pm` wird in eine gesonderte Datei geschrieben, damit der Autosplit-Mechanismus funktioniert. Ausnahmen sind Packages, die mit `AUTOLOAD: ignore` gekennzeichnet sind. Weiterhin sind teilweise zusätzliche Anweisungen in den gesplitteten Dateien notwendig (in diesem Fall sind es einige `requires` in Modul `Strassen`). Diese werden mit `AUTOLOAD: activate` aktiviert.

Nach dieser Vorbereitung wird `autosplit` aufgerufen. Danach werden die erzeugten Dateien nochmals aufbereitet: für Dateien des Hauptprogramms wird die Package-Bezeichnung `bbbike` durch `main` ersetzt. Alle Dateien geben zu Debug-Zwecken eine Meldung aus, wenn sie nachgeladen werden.

Bei Autoload-Dateien ist es problematisch, daß dateiglobale lexikalische Variablen in den nachgeladenen Funktionen nicht sichtbar sind. Deshalb müssen solche Variablen vermieden und stattdessen das Pragma `use vars` verwendet werden.

Ein großer Teil des Programmcodes (ca. 4000 der 12000 Zeilen des Hauptprogramms) besteht aus Initialisierungen sowie Widget- und Menü-Definitionen. Da diese nicht in Funktionen enthalten sind, können sie nicht ausgelagert werden.

Bei Tests ist der Effizienzgewinn eher bescheiden geblieben. Durch das Auslagern ergibt sich eine Einsparung beim Speicher und bei der Startzeit um ungefähr 5%[57].

lowmem Für Rechner mit sehr knappen RAM-Speicher kann die Option `-lowmem` beim Start angegeben werden. Dadurch werden einige Features ausgeschaltet:

- Beim Start wird das Straßennetz nicht gezeichnet, sondern das Zeichnen muß manuell angestoßen werden.
- Die Berechnung des Straßennetzes muß ebenfalls manuell gestartet werden.
- Tooltips (Balloons) werden nicht verwendet.
- Die Berechnung mit `BikePower` wird ausgeschaltet, somit ist eine leistungsbezogene Berechnung (Wind, Steigungen) nicht möglich.
- Audio-Funktionen werden ausgeschaltet.
- Die Icons werden nicht mit Bildern, sondern mit Text beschriftet.

Durch diese Featurebegrenzung kann man den Speicherverbrauch leicht senken[58].

[57] Die genauen Ergebnisse befinden sich in der Datei `diplom/code/bench_autoload.txt`.

[58] Nach dem Start belegt BBBike damit statt 14360 KB nur noch 12212 KB virtuellen Speicher. Die Zahlen wurden unter Windows NT 4.0 ermittelt.

Nachladen und Auslagern von Modulen Als weitere Optimierung kann man das Laden von Modulen verzögern. Statt des normalen Ladens mit `use` wird `require` verwendet, und zwar an der Stelle, an der die Funktion aus dem Modul verwendet wird. Da mit `require` Module erst zur Laufzeit geladen werden, können damit zusammenhängende Fehler ebenfalls erst während der Laufzeit erkannt werden. Ein weiterer Schwachpunkt ist, daß Funktionen aus Modulen in das Hauptprogramm nur importiert werden können, wenn `use` verwendet wird. Andernfalls muß der vollständig qualifizierte Name (also z.B. `File::Basename::basename` statt `basename`) angegeben werden.

Um diese Unbequemlichkeit zu ersparen, kann das Pragma `autouse` verwendet werden. Mit diesem Pragma kann angegeben werden, welche Funktionen aus welchen Modulen verwendet werden sollen. Das Laden des Moduls wird dann so lange verzögert, bis die Funktion erstmals ausgeführt wird. `autouse` funktioniert nicht bei allen Modulen. Die Probleme sind in der dazugehörigen Manpage beschrieben. Weiterhin verlangt `autouse`, daß der Name des nachzulandenden Moduls und der Package-Name identisch sind, was bei vielen Modulen von BBBike nicht der Fall ist.

Betriebssystemspezifische Funktion bzw. Funktionen, die nur für eine bestimmte Tk-Version notwendig sind, sind ebenfalls aus Gründen der Effizienz ausgelagert.

Dynamisches Zeichnen des Straßennetzes In einem weiteren Experiment wird dynamisches Zeichnen für das Straßennetz verwendet. Dieses Feature wird mit der Option `-turbo` aktiviert. Der Code dazu befindet sich in `BBBikeExp.pm`. Das Prinzip ist folgendes: zu Beginn wird für alle zu zeichnenden Kartenobjekte festgestellt, in welchen Quadranten eines gedachten Gitters sie sich befinden. Bei Veränderungen des Kartenausschnitts, sei es durch Scrolling oder Zooming, werden für die aktuell angezeigten Quadranten die Kartenobjekte gezeichnet. Es gibt eine Reihe von Vor- und Nachteilen bei dieser Vorgehensweise:

Vorteile

- Da nur die angezeigten Kartenobjekte gezeichnet werden müssen, können ohne größere Performance-Verluste alle Kartenobjekttypen (Straßen, Landstraßen, Orte, Gewässer usw.) beim Start aktiviert werden.

Nachteile

- Das Scrolling ist etwas schwerfälliger, weil die Kartenobjekte dynamisch gezeichnet werden. Sobald allerdings ein Quadrant gezeichnet wurde, ist ein erneutes Zeichnen nicht mehr notwendig.

- Das Ein- und Ausschalten des Zeichnens von bestimmten Kartenobjekttypen und weiteren Attributierungen (Ausgabe von Straßennamen etc.) ist noch nicht implementiert.

- Es werden nur Streckenteile gezeichnet, bei denen wenigstens ein Punkt innerhalb eines zu zeichnenden Quadranten liegt. Unter Umständen kann es vorkommen, daß beide Endpunkte außerhalb von zu zeichnenden Quadranten liegen, die Strecke dazwischen aber zu zeichnende Quadranten durchquert.

Verzögertes Abarbeiten Teilweise wird im Programm mit kurzen Verzögerungen für die Abarbeitung von Aktionen gearbeitet. Ein Beispiel hierfür ist das „Restacking". Hierbei wird die korrekte Schichtung der Kartenobjekte neu erzeugt. Nach jedem Zeichenvorgang wird diese Funktion aufgerufen. Da mehrere Zeichenvorgänge hintereinander gestartet werden können, ist es sinnvoll (und auch so implementiert), daß das Restacking solange verzögert wird, bis der letzte Zeichenvorgang abgeschlossen ist.

Weiterhin kann mit Verzögerungen gearbeitet werden, wenn man Attribute von Kartenobjekten ändert. In diesem Fall wird die Änderung mit der Funktion `pending` registriert. Falls eine automatische Aktualisierung gewünscht wurde (mit dem Menüpunkt „Ausgabe sofort aktualisieren"), wird die Änderung sofort durchgeführt, ansonsten erst, wenn „Alles aktualisieren" aufgerufen wurde (dieser Menüpunkt ruft die Funktion `update` auf).

Caching von Datenstrukturen In `Strassen.pm` ist das Caching von Datenstrukturen realisiert. Das Caching wird im Unterverzeichnis `cache` durchgeführt. Falls aus dem Programm heraus versucht wird, eine Cache-Datei zu laden, wird zunächst anhand der Änderungszeit überprüft, ob sich die Quelldateien geändert haben. Wenn das der Fall ist, wird die normale Berechnung der Datenstruktur durchgeführt und dann die neu berechnete Datenstruktur als Cache-Datei gespeichert. Für das Format der Cache-Dateien stehen drei Module zur Verfügung: `VirtArray`, `Data::Dumper` und `Storable`.

- `VirtArray` kann nur für flache Arrays (d.h. Arrays ohne weitere Referenzen) verwendet werden.

- `Storable` ist ein Zusatzmodul aus dem CPAN[59] und erlaubt das effiziente Speichern von Perl-Datenstrukturen in einem binären Format.

- `Data::Dumper` ist ein Standardmodul und speichert Perl-Datenstrukturen in eine Textdatei, die mit dem Perl-Parser analysiert werden und somit mit `do` und `require` geladen werden kann.

[59] `<URL:http://www.cpan.org/authors/id/RAM>`

In `try_cache` wird überprüft, welche Cache-Methoden überhaupt zur Verfügung stehen (`VirtArray` ist evtl. nicht kompiliert und `Storable` standardmäßig nicht mit Perl ausgeliefert). Im Dateinamen der Cachedatei sind neben dem Namen der Datenstruktur auch weitere Parameter abespeichert, die die eindeutige Identifizierung ermöglichen. So können beispielsweise unterschiedliche `StrassenNetz`-Strukturen gleichzeitig gespeichert werden.

Folgende Datenstrukturen werden im Cache abgespeichert:

- Die Liste aller Kreuzungen (das Ergebnis aus `all_crossings`).
- Die Zugehörigkeit von Strecken und Punkten zu Gitter-Quadranten (das Ergebnis aus `make_grid`).
- Die `StrassenNetz`-Datenstruktur, falls die entsprechende XS-Funktion nicht aufgerufen wurde.

Ausstehende Optimierungen Ein weiterer Optimierungsansatz, der noch nicht implementiert ist, wäre die strikte Objektorientierung und Modularisierung des Programms. Dadurch könnte der recht große Initialisierungscode in Module ausgelagert werden, der nur geladen wird, wenn das entsprechende Feature wirklich benötigt wird.

Bei der Modularisierung würde man ein Hauptobjekt `BBBikeContext` halten, das Zugriff auf die Subsysteme (Kartenobjekte, BikePower, verschiedene Teile der GUI) besitzt. Die Subsysteme würden nur bei Bedarf initialisiert werden. Durch eine solche Modularisierung könnte man auch mehrere Kartenfenster parallel halten und teilweise Menüstrukturen ausblenden.

3.9 Entwicklung

Bei der Entwicklung des Systems wurden die gewöhnlichen Programmierwerkzeuge wie Editoren und Unix-Tools verwendet. Zusätzlich konnten durch den Einsatz einer hochdynamischen Sprache wie Perl auch andere Techniken verwendet werden.

Verwendung des dynamischen Nachladens Perl bietet die Möglichkeit, Funktionen zu überschreiben. Dabei wird nur eine Warnung ausgegeben. Nachfolgende Aufrufe der Funktion verwenden den neuen Programmcode. Dieses Feature kann dazu verwendet werden, um Codestücke im laufenden Programm auszutauschen und neu auszutesten. Im Advanced-Modus von BBBike wurden dafür zwei Menüpunkte eingerichtet: „Eval /tmp/add.pl“ und „Reload modules“.

Der erste dieser beiden Menüpunkte evaluiert den Inhalt der Datei `/tmp/add.pl` (oder es wird, falls diese Datei nicht existiert, nach einer anderen Perl-Datei gefragt) und überschreibt evtl. bereits existierende gleichnamige Funktionen.

Der zweite Menüpunkt dient zum automatischen Nachladen von geänderten Modulen. In regelmäßigen Abständen (jede Minute) wird in der Funktion `check_new_modules` überprüft, ob neue Module in das System geladen wurden. Die Zeit der letzten Änderung wird im Hash `%module_time` gespeichert. Somit kann in `reload_new_modules` festgestellt werden, welche Module sich seit ihrem ersten Laden geändert haben. Diese Module werden mittels `do` nachgeladen. Diese Methode funktioniert allerdings nur bei Funktionen, die nicht als Referenz im aufrufenden Objekt gebunden sind, wie das bei Menüeinträgen der Fall ist.

Es gibt weitere Probleme, die mit diesen Methoden auftreten. So gehen dateiglobale lexikalische Variablen beim Nachladen einer einzelnen Funktion verloren, da sich die neu geladene Funktion in einer anderen Datei befindet und damit außerhalb des Gültigkeitsbereichs der lexikalischen Variablen. Weiterhin gibt es offenbar Memory Leaks, die den BBBike-Prozeß bei häufigen Nachladen größer werden lassen.

Trotz aller Probleme lohnt sich die Verwendung dieses Features, da dadurch die Startzeit des Programms eingespart wird und man bereits die gewünschten Einstellungen am Programm durchgenommen hat.

Normaler Perl-Debugger Eine weitere Möglichkeit ist die Verwendung des standardmäßigen Perl-Debuggers. Allerdings wird die Ausführgeschwindigkeit des Programms bei Verwendung des Debuggers gebremst. Beispielzeiten für den Start des Programms auf einem System mit Pentium-kompatiblen Prozessor mit 166 MHz sind: ca. 26 Sekunden im Debug-Modus und ca. 15 Sekunden im Normal-Modus. Durch diesen Zeitverlust ist es oft effizienter, mit geschickten Ausgabefunktionen für Variablen zu arbeiten oder `ptksh` (siehe weiter unten) zu verwenden.

Tk-basierter Perl-Debugger ptkdb Es existiert ein weiterer Perl-Debugger, der im Gegensatz zum Standard-Debugger eine auf Tk basierende grafische Benutzeroberfläche besitzt[60]. Allerdings gibt beim Verwenden dieses Debuggers bei Tk-Programmen es noch Probleme, weil es zu Blockierungen (z.B. durch gegenseitige „Grabs“[61]) kommen kann. Außerdem leidet auch dieser Debugger unter dem gleichen Performance-Verlust wie der Standard-Debugger.

„printf-Debugging“ Eine primitive Art des Debuggings ist das „printf-Debugging“. Dabei werden einfach an markanten Stellen des Programms Werte von Variablen ausgegeben. Natürlich ist diese Art des Debuggens auch unter

[60] http://www.cpan.org/authors/id/A/AE/AEPAGE

[61] Bei einem „Grab“ wird die gesamte Ein- und Ausgabe auf ein bestimmtes Fenster gelenkt. Mit der Maus ist somit ein Ändern des Fokus (des aktiven Fensters) nicht möglich.

Perl möglich. Für die Ausgabe von komplexen Datenstrukturen (Hashes und Arrays sowie Kombinationen aus beiden) kann das Modul `Data::Dumper` verwendet werden.

Interaktive Perl-Shell ptksh In der Standard-Perl/Tk-Distribution ist auch die interaktive Perl-Shell `ptksh` enthalten. Diese Shell kann sowohl von der Kommandozeile als auch aus Programmen heraus aufgerufen werden. Im letzteren Fall läuft die Shell im Prozeßraum des Programms. Dies kann dazu verwendet werden, um die Belegungen von globalen Variablen zu überprüfen. Lexikalische Variable (sowohl dateiglobale als auch funktionslokale) können hiermit leider nicht beobachtet werden, aus den gleichen Gründen wie beim oben beschriebenen dynamischen Nachladen. Die Shell kann auch gebraucht werden, um nachträglich Widgets einzufügen oder Funktionen manuell aufzurufen.

Tk::WidgetDump Um die hierarchische Struktur der grafischen Benutzeroberfläche („Widget-Baum") ausgeben zu können, wurde während der Entwicklung von BBBike das Modul `Tk::WidgetDump` geschrieben. Dieses Modul zeigt für alle Widgets in der Hierarchie den Tk-Namen, die Klasse (sowohl die interne Tk-Klasse als auch die entsprechende Perl-Klasse) und einen charakteristischen Wert des Widgets (z.B. den Text eines Labels oder den Namen eines Bildes). Die Spalte „Size" bezieht sich nur auf die Größe des am Widget gekoppelten Perl-Hashs; die interne Größe des Widgets (also den von Tk verwendeten Speicher) wird dabei nicht betrachtet.

Hybride Perl-Skripte/Module Mit Perl ist es leicht möglich, Skripte zu schreiben, die gleichzeitig als Perl-Module verwendet werden können. Grundsätzlich sieht der Aufbau dafür so aus: die Datei enthält zwei Packages, einmal die Package des eigentlichen Moduls und weiterhin die Package `main`. Am Ende des Modul-Packages wird die Zeile

```
return 1 if caller;
```

eingefügt, mit der die weitere Abarbeitung abgebrochen wird, wenn aus einem `use`- oder `require`-Kontext aufgerufen wird. Beim Aufruf als Skript wird der Rest (das `main`-Modul) abgearbeitet, in dem die Kommandozeile (z.B. per `Getopt::Long`) analysiert und eine Funktion des eigentlichen Moduls mit diesen Parametern aufgerufen wird.

3.10 Aufgetauchte Probleme

Ein Problem ist die fehlende Datenbasis. Als Zwischenlösung wurden für das Programm die Daten wie Straßennetz, Qualität des Belags, Position von Ampeln etc. manuell erfaßt. Eine genauere Recherche bei Tiefbauämtern und kartographischen Instituten sollte Informationen darüber liefern, ob solche Daten bereits vorhanden sind.

Subjektivität von Beobachtungen Teilweise stützen sich Berechnungen auf subjektive Beobachtungen. Beispielsweise wurden bei der Kategorisierung der Straßenqualität keine objektiven Kriterien verwendet. Dadurch verliert die Anwendung von zusätzlichen Parametern bei der Routensuche an Aussagekraft. Allerdings wäre eine Untersuchung der Attributierung nach objektiven Kriterien wegen des benötigten Aufwands nicht machbar.

Kapitel 4

Abschließende Betrachtungen

4.1 Grenzen des Systems

Das System ist als Informationssystem angelegt. Bei der Routensuche wäre die Verwendung von mehr Intelligenz wünschenswert. Weitere Ideen hierzu finden sich im Abschnitt 4.3.3 auf Seite 104.

Das Zusammenstellen der Datenbasis ist schwierig. Zur Zeit müssen die Koordinaten der Straßen aus gewöhnlichen Karten ausgemessen und in die Datenbasis übertragen werden. Weiterhin müssen zusätzliche Daten wie Straßenqualität und Vorhandensein von Radwegen, für die meines Wissens keine Daten in digitaler Form vorliegen, erfaßt werden. Diese Probleme sind die größten Hürden bei einer Portierung des Systems für andere Städte.

Dies gilt auch für die Zeiten der Ampelphasen, sowohl für die Länge der Phasen als auch für die Verschiebung der Ampelphasen zwischen benachbarten Ampeln. Mit dieser Information könnte man den Verlust durch Ampelschaltungen wesentlich genauer erfassen und (in günstigen Fällen) die Fahrzeit exakt voraussagen.

Trotz aller Optimierungen ist das Stand-Alone-Programm recht groß und benötigt einen schnellen Rechner. Dies ist allerdings bei heute handelsüblichen Rechnern mit Intel-Prozessoren mit über 300 MHz Prozessortakt kaum ein Problem.

Ein Fahrradinformationssystem auf einem Desktop-Rechner befindet sich eigentlich an der falschen Stelle. Eine Portierung auf Handheld-Computer, die direkt auf dem Fahrrad montiert werden könnten, am besten mit GPS-Anbindung, wäre wünschenswert.

4.2 Analyse der WWW-Zugriffe

Auf die WWW-Version kann bereits öffentlich über die URL
`<URL:http://user.cs.tu-berlin.de/~eserte/bbbike/cgi/bbbike.cgi>` zu-

gegriffen werden. Die Seite wurde in das deutsche Yahoo-Verzeichnis (`<URL:http://www.yahoo.de>`) eingetragen.

Um die Akzeptanz der Anwendung zu ermitteln, wurde die `AccessLog`-Datei für den Monat August 1999 analysiert. In diesem Zeitraum wurde insgesamt 6424 Mal auf das CGI-Programm `bbbike.cgi` zugegriffen. Diese Zugriffe können 528 Sitzungen (Sessions) zugeordnet werden. Dabei wurde 221 Mal nur die Startseite aufgerufen, bei den restlichen 307 Sitzungen wurde anscheinend mit dem Programm richtig gearbeitet. Bei einer richtigen Sitzung wurde das CGI-Programm demnach durchschnittlich ca. 20 Mal aufgerufen.

Bei der Analyse der *Referer*, also der Seiten, von denen die CGI-Anwendung aufgerufen wurde, kann man feststellen, daß 31% der Besucher von `<URL:http://www.yahoo.de>` aus die Seite gefunden haben. Weitere 19% verteilen sich auf andere Seiten, von denen ein Link auf `bbbike.cgi` existiert[1]. Die restlichen 40% haben keinen Referer, es kann also angenommen werden, daß diese Zugriffe gezielt (z.B. aus Bookmarks heraus) vorgenommen wurden.

Weitere Ergebnisse der Analyse: den Microsoft Internet Explorer haben 49% der Anwender verwendet, den Netscape Navigator 44%. Der Anteil an modernen WWW-Browsern, also diejenigen, die auch DHTML[2] unterstützen, beträgt 80%. Bei den Betriebssystemen sieht die Verteilung wie folgt aus: Windows 85%, Macintosh 4% und Unix 3%. Bei den restlichen Zugriffen konnten entweder Betriebssystem oder verwendeter Browser nicht festgestellt werden.

4.3 Ausblick

4.3.1 Externe Datenpflege

Wünschenswert wäre die durch Benutzer durchgeführte Pflege der Datenbasis über das WWW[3]. Das System würde eine Registrierung der Benutzer erfordern, um eine eindeutige Zuordnung der Daten in der Datenbasis erzeugen zu können.

Für die Pflege der Datenbasis sind verschiedene Ausbaustufen denkbar. In einer einfacheren Version könnte der Benutzer in einem existierenden Straßennetz den Streckenabschnitten Attribute zuweisen. Attribute wären hierbei Straßenqualität, Vorhandensein von Radwegen, Ampeln und weitere in Fließtext geschriebene Notizen. Die Daten müssen einen Timestamp erhalten, damit die Aktualität

---

[1] Unter anderem sind das die URLs: `<URL:http://www.infobike.de>`, `<URL:http://verkehr.epilog.de>`, `<URL:http://www.de.freebsd.org/~wosch>` und `<URL:http://www.bikesport.de>`.

[2] DHTML wird vom MSIE und Netscape ab Version 4 unterstützt.

[3] Eine bereits existierende Datenbank, die auf Daten von Benutzern aufgebaut ist, ist die Compact-Disk-Datenbank CDDB ([Cdd99]). Neue Einträge werden per E-Mail abgeschickt, die in einem speziellen Format abgeschickt werden (dieses Format wird von CDDB-fähigen CD-Playern automatisch erzeugt). Der Vorschlag für den neuen Eintrag wird manuell überprüft, bevor er in die Datenbank eingefügt wird ([Cdf99]).

des Attributs abgeschätzt werden kann. Bei mehreren, sich evtl. widersprechenden Attributen müßte entweder ein Administrator eingreifen oder die aktuelleren Daten würden automatisch die Priorität in der Datenbasis erhalten. Unter Umständen kann die Attributierung vom verwendeten Fahrrad des Benutzers oder von seinen Präferenzen abhängen. Es könnte versucht werden, dies in eine Bewertung der Attribute mit einfließen zu lassen.

In einer späteren Version könnte man dem Benutzer auch neue Straßen einfügen lassen. Dafür wären digitalisierte Karten als Vorlage erforderlich.

In [Anz98] wird von der australisch-neuseeländischen Behörde ANZLIC ein Vorschlag für die Behandlung von Metadaten im geografischen Bereich geboten. Die Verwendung von Metadaten nach diesem Schema wäre für die aufzubauende Datenbank wünschenswert.

4.3.2 Java-Version

Eine höhere Interaktivität bei der Web-Version kann durch die Entwicklung eines Java-Applets erreicht werden. Alternativ könnte auch eine Plugin-Lösung auf Tcl/Tk- oder Perl/Tk-Basis entwickelt werden, wobei der Aufwand für letzteres am geringsten ist.

4.3.3 Weiterentwicklung zu einem Expertensystem

Ein Expertensystem sollte alle Attribute, die bei der Suche verwendet werden, z.B. die Werte für die Optimierungen, automatisch setzen können. Die Präferenzen sollten vom Benutzer in weitgehend natürlicher Sprache eingegeben werden können („Ich fahre am liebsten im Grünen und fahre niemals auf Hauptstraßen auf der Fahrbahn“ oder „Ich bin meist schnell unterwegs und möchte auf meinem Rennrad nur gut ausgebaute Straßen benutzen“). Das Expertensystem sollte daraus die geeigneten Optimierungsparameter errechnen können. Nach Benutzung der Route könnte der Benutzer die vorgeschlagene Strecke bewerten, indem er einzelne Routenteile als geeignet oder ungeeignet festlegt. Im Dialog müßte das Expertensystem feststellen können, warum der Benutzer die Strecke als ungeeignet empfindet. Das Ergebnis könnte einerseits in eine Korrektur der Präferenzen des Benutzers münden (möglicherweise hat der Benutzer angegeben, daß er Hauptstraßen benutzen würde, aber beim Befahren einer Hauptstraße festgestellt, daß er diese Aussage korrigieren möchte) oder anderseits die Datenbasis korrigieren (z.B. eine nicht bekannte Kopfsteinpflasterstrecke). Im letzteren Fall wäre eine Angleichung mit einer zentralen, sich im Internet befindenden Datenbasis wünschenswert.

Bei einer Kopplung des Systems mit einem GPS-Empfänger könnte eine automatische Aufzeichnung einer gefahrenen Route erfolgen. Später kann anhand der gefahrenen Fahrzeit und der Kenntnis von vorhandenen Ampeln und Vorfahrtsregelungen die tatsächlich gefahrene Geschwindigkeit und Verzögerungen für Teilstrecken abgeschätzt werden. Beim häufigen Befahren einer Strecke könnten statistische Mittelwerte errechnet werden. Mit solchen gesammelten Werten für mehrere Strecken können Methoden der KI verwendet werden, um Regeln aufstellen zu können (z.B. „auf Hauptstraßen fährt man meistens schneller als auf Nebenstraßen", „bei Gegenwind einer bestimmten Stärke fahren Radfahrer um diesen Prozentsatz langsamer").

4.3.4 Profile

Bislang sind Optionen nur für einen Benutzer und ein Fahrrad festgeschrieben. Ein Ausweiten auf mehrere Räder oder vielleicht sogar Präferenzen („ich fahre manchmal sportlich und manchmal langsam") wäre wünschenswert. Dieses Merkmal würde Änderungen sowohl am Hauptprogramm als auch am Modul `BikePower` erfordern.

4.3.5 Ausweitung auf andere Städte und Sprachen

Für eine Portierung des Programms auf andere Städte müßten einige Änderungen vorgenommen werden. Die Dateien der Datenbasis, die sich bislang im `data`-Verzeichnis befinden, müßten eine weitere Hierarchiestufe nach unten verschoben werden, wobei für jede Stadt ein Verzeichnis erstellt wird (z.B. `berlin`, `hamburg`...).

Andere Städte brauchen andere Koordinatensysteme. Deshalb werden für jede Stadt separate `transpose`-Funktionen und spezifische `Karte/*`-Module benötigt.

Weiterhin gibt es in der Perl/Tk-Anwendung und im CGI-Programm textuelle und grafische Elemente, die sich nicht auf alle Städte anwenden lassen. So gibt es nicht in allen Städten U- und S-Bahnen; in anderen Ländern sehen die entsprechenden Symbole völlig anders aus.

Schließlich muß bei einer Ausweitung auf andere Länder das Programm mehrsprachig gemacht werden. Da die API[4] für andere Programmierer interessant sein könnte, wäre eine Umsetzung aller Modul- und Funktionsnamen in englische Bezeichnungen wünschenswert.

4.4 Bestehende Fahrradinformationssysteme

4.4.1 Allgemeine Informationssysteme

Mit `<URL:http://www.infobike.de>` und `<URL:http://www.bikesport.de>` existieren klassische Informationssysteme, die allgemeine Informationen zum Thema Radfahren geben. Die Informationen enthalten u.a. Linklisten, Tourenbe-

[4] Application Programming Interface

schreibungen und Rennveranstaltungen. Im Gegensatz zu BBBike gibt es keine Möglichkeiten zum individuellen Suchen und Darstellen von Routen.

4.4.2 Cycle Mapping Project Sydney

In Sydney hat eine Gruppe engagierter Radfahrer (Cameron Shorter et al.) ein Projekt zur Darstellung von Fahrradrouten in Sydney ins Leben gerufen ([Syd99]). Bei diesem Projekt liegt der Schwerpunkt in der Visualisierung von Stadtplänen und Fahrradrouten und weniger auf der automatisierten Suche. Ziel ist es, eine WWW-basierte Sammlung von Karten zu haben, auf die jeder zugreifen und auch modifizieren (erweitern) kann. Die Karten sollen für Radfahrer interessante Information darstellen: Sicherheitsgrad, Straßentyp, Qualität der Straßenoberfläche, „Direktheit" einer Route und textuelle Beschreibungen von Besonderheiten (z.B. Schlaglöcher).

Es sollen verschiedene Ebenen (Layer) dargestellt werden können: Fahrradrouten, geografische „Features" (Flüsse, Hügel), Straßen, ein Kartengitter sowie als Hintergrundbild eine reale Karte von Sydney.

Es existiert bereits ein Prototyp mit einer WWW-Schnittstelle. Als Kartenserver wird `mapserv` verwendet.

Geplant ist eine Schnittstelle, mit der jeder seine eigenen Routen und dazugehörige Kommentare eingeben kann.

4.4.3 BikeMap.com

In [Bkm99] befindet sich eine Webseite im Aufbau, die Fahrrad-Karten in den USA zeigen soll. Zum Service werden Karten von vorgefertigten Routen, Städten, Regionen und Staaten angezeigt. Weiterhin werden zusätzliche Informationen wie Fahhradmitnahme auf Überlandverbindungen mit öffentlichen Verkehrsmitteln angeboten.

Literaturverzeichnis

[Anz98] *ANZLIC, Metadata Guidelines*, Juli 1998.
`<URL:http://www.anzlic.org.au/metaelem.htm>`.

[Apa99] *Apache Server Project*, Oktober 1999.
<URL:http://www.apache.org>.

[ARE99] *AutoRoute Express Europa 2000* von Microsoft, September 1999.
<URL:http://www.microsoft.com/GERMANY/homeoffice/autoroute/default.htm>.

[BDB99] *Berkeley Database*, Sleepycat Software, September 1999.
<URL:http://www.sleepycat.com>.

[Bkm99] *BikeMap.com*, September 1999.
<URL:http://www.bikemap.com>.

[Bpw99] bike_power - the program, September 1999.
<URL:http://okcforum.org/markg/mark/bikepower.html>.

[Cdd99] *CDDB*, September 1999.
<URL:http://www.cddb.org>.

[Cdf99] *CDDB-FAQ*, September 1999.
<URL:http://www.cddb.org/userfaq.html/page=7>.

[CGI98] *CGI — Common Gateway Interface*, W3C — The World Wide Web Consortium, März 1998.
<URL:http://www.w3.org/CGI/>.

[CPA99] *Comprehensive Perl Archive Network (CPAN)*, Oktober 1999.
<URL:http://www.cpan.org/>.

[CSS99] *Cascading Style Sheets*, W3C — The World Wide Web Consortium, September 1999.
<URL:http://www.w3.org/Style/css/>.

[DHT97] *DevEdge Online — Dynamic HTML Manual*, Netscape, 1997.
`<URL:http://developer.netscape.com /docs/manuals/communicator/dynhtml/index.htm>`.

[DWD99] *Deutscher Wetterdienst*, August 1999.
`<URL:http://www.dwd.de/>`.

[FaB99] *Fahrinfo Berlin*, September 1999.
`<URL:http://www.fahrinfo-berlin.de>`.

[FBh99] Homepage von *FreeBSD Inc.*, September 1999.
`<URL:http://www.FreeBSD.org>`.

[FBp99] *Verzeichnis der FreeBSD-Ports*, September 1999.
`<URL:ftp://ftp.FreeBSD.org/pub/FreeBSD/ports>`.

[Fcg99] *The Official FastCGI Home Page*, September 1999.
`<URL:http://www.fastcgi.com>`.

[FDB99] *Fahrplanauskunft der Deutschen Bahn AG*, September 1999.
`<URL:http://bahn.hafas.de>`.

[Fla98] David Flanagan, *Java in a Nutshell*, O'Reilly Verlag, 1998.

[Flk99] *Routenplaner von Falkplan*, September 1999.
`<URL:http://www.falk-online.de/go_routing.html>`.

[Flo94] Jim Flowers, *X Logical Font Description Conventions*, X Consortium Draft Standard, Februar 1994. ISBN 1-56592-071-6.

[Fre96] N. Freed, N. Borenstein, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types.* Request for Comments (2046): Network Working Group, November 1996.
`<URL:ftp://ftp.cs.tu-berlin.de/pub/doc/rfc/rfc2046.gz>`.

[GDB99] *GDBM*, Mai 1999.
`<URL:http://www.gnu.org/software/gdbm/gdbm.html>`.

[Gim99] *GIMP — GNU Image Manipulation Program*, September 1999.
`<URL:http://www.gimp.org/>`.

[Hac99] Homepage der *HaCon Ingenieurgesellschaft mbH*, September 1999.
`<URL:http://www.hacon.de>`.

[HTM99] *HTML Home Page.* W3C — The World Wide Web Consortium, Juni 1999.
`<URL:http://www.w3.org/MarkUp/>`.

[HTT96] *Hypertext Transfer Protocol — HTTP/1.0.* Request for Comments: Network Working Group, Mai 1996.
`<URL:ftp://ftp.cs.tu-berlin.de/pub/doc/rfc/rfc1945.gz>`.

[HTT99] *HTTP — Hypertext Transfer Protocol Overview.* W3C — The World Wide Web Consortium, Juni 1999.
`<URL:http://www.w3.org/Protocols/>`.

[IVU99] Homepage der *IVU Traffic Technologies AG*, Juli 1999.
`<URL:http://www.ivu.de>`.

[Jas99] Homepage von *Javasoft*, September 1999.
`<URL:http://www.javasoft.com>`.

[Jon91] Oliver Jones, *Einführung in das X-Window-System*, Hanser, Wien, 1991.

[Kan88] Kanal, L.; and Kumar, V., *Search in Artificial Intelligence*, Springer-Verlag New York, 1988.

[Kor88] Korf, Richard E., *Optimal Path-Finding Algorithms*, in Search in Artificial Intelligence, Springer-Verlag New York, 1988, Seiten 233-241.

[Kuk89] Milan Kukrika, *Programski jezik C*, Školska knjiga Zagreb, 1989.

[Lei97] *Leistung und Energie*, Juli 1997.
`<URL:http://www.asn-linz.ac.at/iaac/ptuj/leistung.htm>`.

[lgd99] Homepage der *GD-Bibliothek* (gd 1.7.2), September 1999.
`<URL:http://www.boutell.com/gd/>`.

[Lin94] H. Linde, *Einführung in das Verkehrswesen* (Skript zur Vorlesung), Technische Universität Berlin, 1994, Seite 0.3.

[Lnx99] *Linux*, Dr. K.-W. Schulte, September 1999.
`<URL:http://www.hrz.uni-wuppertal.de /dienste/software/os/linux>`.

[Lxa99] *Warum Linux?*, Peter Bingel, April 1999.
`<URL:http://www.lxschule.de/txt/warumli.html>`.

[Met99] *Meteorologisches Institut der Freien Universität Berlin*, März 1999.
`<URL:http://www.met.fu-berlin.de/>`.

[MMb73] Müller-Merbach, *Operations Research*, 3. Auflage, Verlag Vahlen, 1973

[Mpl99] *Apache/Perl Integration Project*, September 1999.
`<URL:http://perl.apache.org/>`.

[MSI99] *Microsoft Internet Explorer*, Juli 1999.
`<URL:http://www.microsoft.com/windows/ie>`.

[Net99] *DevEdge Online - DevEdge Home Page*, Netscape Communications Corporation, September 1999.
`<URL:http://developer.netscape.com>`.

[NCS99] *CGI: Common Gateway Interface*, National Center for Supercomputing Applications at the University of Illinois at Urbana - Champaign, IL, USA, Dezember 1995.
`<URL:http://hoohoo.ncsa.uiuc.edu/cgi/intro.html>`.

[Pal99] *3Com, Palm Computing*, September 1999.
`<URL:http://www.palm.com>`.

[Pea85] R. Dechter and J. Pearl, *Generalized best-first search strategies and the optimality of* A^*, Journal of the Association for Computing Machinery, Vol. 32, No. 3, July 1985, Seiten 505–536.

[Per99] Homepage von *Perl*, September 1999.
`<URL:http://www.perl.org>`.

[Pex99] *Perl2Exe Home Page*, Oktober 1999.
`<URL:http://www.demobuilder.com/perl2exe.htm>`.

[Piv90] Rainer Pivit, *Die Fahrwiderstände in Formeln* in Radfahren 2/1990.
`<URL:http://www.bielefeld.netsurf.de:8080`
`/~r_pivit/aero/formel.html>`.

[PLZ99] *xplz: Postleitzahldateien*, Juni 1993.
`<URL:ftp://ftp.cs.tu-berlin.de/pub/doc/PLZ/xplz>`.

[PMp99] *Perl Manpages*, September 1999.
`<URL:http://www.perl.com/pub/v/documentation>`.

[PRK97] Nate Patwardham et al, *Perl Resource Kit — Unix Edition*, O'Reilly & Associates, Inc., 1997.

[Pup88] Frank Puppe, *Einführung in Expertensysteme*, Springer, Berlin, 1988.

[Rai98] Paul Raines, *Tcl/Tk kurz & gut*, O'Reilly Verlag, 1998.

[Sim92] K. Simonsen, *Character Mnemonics and Character Sets.* Request for Comments (1345): Network Working Group, Juni 1992.
`<URL:ftp://ftp.cs.tu-berlin.de/pub/doc/rfc/rfc1345.gz>`.

[Scr99] Homepage von *Scriptics*, September 1999.
`<URL:http://www.scriptics.com>`.

[Sri98] Sriram Srinivasan, *Fortgeschrittene Perl-Programmierung*, O'Reilly Verlag, 1998.

[Sti85] Stickel, M.E., and Tyson, W.M., *An analysis of consecutively bounded depth-first search with applications in automated deduction*, in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-85), Los Angeles, Ca., August, 1985.

[Str99] `bike_power` - the program, Wolfgang Strobl, September 1999.
`<URL:http://www.geocities.com /SiliconValley/Ridge/3933/fahrrad/bikepwr.htm>`.

[Syd99] *Cycle Mapping Project Sydney*, September 1999.
`<URL:http://www.nccnsw.org.au/member/cmass/maps>`.

[Tw99] Homepage von *Talkingweb*, September 1999.
`<URL:http://www.talkingweb.de>`.

[Uni99] *LZW information*, UNISYS, September 1999.
`<URL:http://corp2.unisys.com/LeadStory/lzwfaq.html>`.

[Vtk99] Homepage des *V Toolkits*, September 1999.
`<URL:http://www.objectcentral.com>`.

[Whi82] Frank R. Whitt and David G. Wilson, *Bicycling Science*, (Second Edition), MIT Press, 1982.

[Wu91] Sun Wu and Udi Manber, *Fast Text Searching With Errors*, Technical report #91-11, Department of Computer Science, University of Arizona, June 1991 (auch verfügbar unter
`<URL:ftp://ftp.cs.arizona.edu/agrep/agrep.ps.1>`).

[Wu92] Sun Wu and Udi Manber, *Agrep — A Fast Approximate Pattern Searching Tool*, 1992 (verfügbar unter
`<URL:ftp://ftp.cs.arizona.edu/agrep/agrep.ps.2>`).

Danksagung

An dieser Stelle möchte ich allen danken, die mir bei der Realisierung dieser Arbeit Unterstützung geleistet haben, sei es in fachlicher oder in moralischer Hinsicht, als Programmtester oder Ideengeber. Besonders hervorheben möchte ich dabei meine Eltern, meine Schwestern Helena und Marija, sowie Gordana, Kati, Michael, Nicole und Wolfram.

Die selbständige und eigenhändige Anfertigung versichere ich an Eides statt.
Berlin, den 15. Oktober 1999

Slaven Rezić